

Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and IP Routers
H. Jonathan Chao, Cheuk H. Lam, Eiji Oki
Copyright © 2001 John Wiley & Sons, Inc.
ISBNs: 0-471-00454-5 (Hardback); 0-471-22440-5 (Electronic)

BROADBAND PACKET SWITCHING TECHNOLOGIES

BROADBAND PACKET SWITCHING TECHNOLOGIES

A Practical Guide to ATM Switches and IP Routers

H. JONATHAN CHAO

CHEUK H. LAM

EIJI OKI



A Wiley-Interscience Publication

JOHN WILEY & SONS, INC.

New York / Chichester / Weinheim / Brisbane / Singapore / Toronto

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in initial capital or ALL CAPITAL LETTERS. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Copyright © 2001 by John Wiley & Sons, Inc. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic or mechanical, including uploading, downloading, printing, decompiling, recording or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ & WILEY.COM.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

ISBN 0-471-22440-5

This title is also available in print as ISBN 0-471-00454-5

For more information about Wiley products, visit our web site at www.Wiley.com.

CONTENTS

PREFACE	xiii
1 INTRODUCTION	1
1.1 ATM Switch Systems / 3	
1.1.1 Basics of ATM networks / 3	
1.1.2 ATM switch structure / 5	
1.2 IP Router Systems / 8	
1.2.1 Functions of IP routers / 8	
1.2.2 Architectures of IP routers / 9	
1.3 Design Criteria and Performance Requirements / 13	
References / 14	
2 BASICS OF PACKET SWITCHING	15
2.1 Switching Concepts / 17	
2.1.1 Internal link blocking / 17	
2.1.2 Output port contention / 18	
2.1.3 Head-of-line blocking / 19	
2.1.4 Multicasting / 19	
2.1.5 Call splitting / 20	
2.2 Switch Architecture Classification / 21	
2.2.1 Time division switching / 22	

2.2.2 Space division switching / 24	
2.2.3 Buffering strategies / 34	
2.3 Performance of Basic Switches / 37	
2.3.1 Input-buffered switches / 37	
2.3.2 Output-buffered switches / 40	
2.3.3 Completely shared-buffer switches / 44	
References / 46	
3 INPUT-BUFFERED SWITCHES	49
3.1 A Simple Switch Model / 50	
3.1.1 Head-of-line blocking phenomenon / 51	
3.1.2 Traffic models and related throughput results / 52	
3.2 Methods for Improving Performance / 53	
3.2.1 Increasing internal capacity / 53	
3.2.2 Increasing scheduling efficiency / 54	
3.3 Scheduling Algorithms / 57	
3.3.1 Parallel iterative matching (PIM) / 58	
3.3.2 Iterative round-robin matching (<i>i</i> RRM) / 60	
3.3.3 Iterative round-robin with SLIP (<i>i</i> SLIP) / 60	
3.3.4 Dual round-robin matching (DRRM) / 62	
3.3.5 Round-robin greedy scheduling / 65	
3.3.6 Design of round-robin arbiters/selectors / 67	
3.4 Output-Queuing Emulation / 72	
3.4.1 Most-Urgent-Cell-First-Algorithm (MUCFA) / 72	
3.4.2 Chuang et al.'s results / 73	
3.5 Lowest-Output-Occupancy-Cell-First Algorithm (LOOFA) / 78	
References / 80	
4 SHARED-MEMORY SWITCHES	83
4.1 Linked-List Approach / 84	
4.2 Content-Addressable Memory Approach / 91	
4.3 Space-Time-Space Approach / 93	
4.4 Multistage Shared-Memory Switches / 94	
4.4.1 Washington University gigabit switch / 95	
4.4.2 Concentrator-based growable switch architecture / 96	
4.5 Multicast Shared-Memory Switches / 97	

4.5.1	Shared-memory switch with a multicast logical queue / 97
4.5.2	Shared-memory switch with cell copy / 98
4.5.3	Shared-memory switch with address copy / 99
References	/ 101

5 BANYAN-BASED SWITCHES **103**

5.1	Banyan Networks / 103
5.2	Batcher-Sorting Network / 106
5.3	Output Contention Resolution Algorithms / 110
5.3.1	Three-phase implementation / 110
5.3.2	Ring reservation / 110
5.4	The Sunshine Switch / 112
5.5	Deflection Routing / 114
5.5.1	Tandem banyan switch / 114
5.5.2	Shuffle-exchange network with deflection routing / 117
5.5.3	Dual shuffle-exchange network with error-correcting routing / 118
5.6	Multicast Copy Networks / 125
5.6.1	Broadcast banyan network / 127
5.6.2	Encoding process / 129
5.6.3	Concentration / 132
5.6.4	Decoding process / 133
5.6.5	Overflow and call splitting / 133
5.6.6	Overflow and input fairness / 134
References	/ 138

6 KNOCKOUT-BASED SWITCHES **141**

6.1	Single-Stage Knockout Switch / 142
6.1.1	Basic architecture / 142
6.1.2	Knockout concentration principle / 144
6.1.3	Construction of the concentrator / 146
6.2	Channel Grouping Principle / 150
6.2.1	Maximum throughput / 150
6.2.2	Generalized knockout principle / 152
6.3	A Two-Stage Multicast Output-Buffered ATM Switch / 154
6.3.1	Two-stage configuration / 154

6.3.2 Multicast grouping network / 157
6.3.3 Translation tables / 160
6.3.4 Multicast knockout principle / 163
6.4 A Fault-Tolerant Multicast Output-Buffered ATM Switch / 169
6.4.1 Fault model of switch element / 169
6.4.2 Fault detection / 172
6.4.3 Fault location and reconfiguration / 174
6.4.4 Performance analysis of reconfigured switch module / 181
6.5 Appendix / 185
References / 187

7 THE ABACUS SWITCH 189

7.1 Basic Architecture / 190
7.2 Multicast Contention Resolution Algorithm / 193
7.3 Implementation of Input Port Controller / 197
7.4 Performance / 198
7.4.1 Maximum throughput / 199
7.4.2 Average delay / 203
7.4.3 Cell loss probability / 206
7.5 ATM Routing and Concentration Chip / 208
7.6 Enhanced Abacus Switch / 211
7.6.1 Memoryless multistage concentration network / 212
7.6.2 Buffered multistage concentration network / 214
7.6.3 Resequencing cells / 217
7.6.4 Complexity comparison / 219
7.7 Abacus Switch for Packet Switching / 220
7.7.1 Packet interleaving / 220
7.7.2 Cell interleaving / 222
References / 224

8 CROSSPOINT-BUFFERED SWITCHES 227

8.1 Overview of Crosspoint-Buffered Switches / 228
8.2 Scalable Distributed Arbitration Switch / 229
8.2.1 SDA structure / 229
8.2.2 Performance of SDA switch / 231
8.3 Multiple-QoS SDA Switch / 234
8.3.1 MSDA structure / 234

8.3.2 Performance of MSDA switch / 236	
References / 238	
9 THE TANDEM-CROSSPOINT SWITCH	239
9.1 Overview of Input–Output–Buffered Switches / 239	
9.2 TDXP Structure / 241	
9.2.1 Basic architecture / 241	
9.2.2 Unicasting operation / 242	
9.2.3 Multicasting operation / 246	
9.3 Performance of TDXP Switch / 246	
References / 252	
10 CLOS-NETWORK SWITCHES	253
10.1 Routing Properties and Scheduling Methods / 255	
10.2 A Suboptimal Straight Matching Method for Dynamic Routing / 258	
10.3 The ATLANTA Switch / 259	
10.3.1 Basic architecture / 261	
10.3.2 Distributed and random arbitration / 261	
10.3.3 Multicasting / 262	
10.4 The Continuous Round-Robin Dispatching Switch / 263	
10.4.1 Basic architecture / 264	
10.4.2 Concurrent round-robin dispatching (CRRD) scheme / 265	
10.4.3 Desynchronization effect of CRRD / 267	
10.5 The Path Switch / 268	
10.5.1 Homogeneous capacity and route assignment / 272	
10.5.2 Heterogeneous capacity assignment / 274	
References / 277	
11 OPTICAL PACKET SWITCHES	279
11.1 All-Optical Packet Switches / 281	
11.1.1 The staggering switch / 281	
11.1.2 ATMOS / 282	
11.1.3 Duan’s switch / 283	
11.2 Optoelectronic Packet Switches / 284	
11.2.1 HYPASS / 284	
11.2.2 STAR-TRACK / 286	

11.2.3	Cisneros and Brackett's Architecture / 287
11.2.4	BNR switch / 289
11.2.5	Wave-mux switch / 290
11.3	The 3M Switch / 291
11.3.1	Basic architecture / 291
11.3.2	Cell delineation unit / 294
11.3.3	VCI-overwrite unit / 296
11.3.4	Cell synchronization unit / 297
11.4	Optical Interconnection Network for Terabit IP Routers / 301
11.4.1	Introduction / 301
11.4.2	A terabit IP router architecture / 303
11.4.3	Router module and route controller / 306
11.4.4	Optical interconnection network / 309
11.4.5	Ping-pong arbitration unit / 315
11.4.6	OIN complexity / 324
11.4.7	Power budget analysis / 326
11.4.8	Crosstalk analysis / 328
	References / 331

12 WIRELESS ATM SWITCHES 337

12.1	Wireless ATM Structure Overviews / 338
12.1.1	System considerations / 338
12.1.2	Wireless ATM protocol / 349
12.2	Wireless ATM Systems / 341
12.2.1	NEC's WATMnet prototype system / 341
12.2.2	Olivetti's radio ATM LAN / 342
12.2.3	Virtual connection tree / 342
12.2.4	BAHAMA wireless ATM LAN / 343
12.2.5	NTT's wireless ATM Access / 343
12.2.6	Other European projects / 243
12.3	Radio Access Layers / 344
12.3.1	Radio physical layer / 344
12.3.2	Medium access control layer / 346
12.3.3	Data link control layer / 346
12.4	Handoff in Wireless ATM / 347
12.4.1	Connection rerouting / 348
12.4.2	Buffering / 340

12.4.3	Cell routing in a COS / 351
12.5	Mobility-Support ATM Switch / 352
12.5.1	Design of a mobility-support switch / 353
12.5.2	Performance / 358
	References / 362

13 IP ROUTE LOOKUPS **365**

13.1	IP Router Design / 366
13.1.1	Architectures of generic routers / 366
13.1.2	IP route lookup design / 368
13.2	IP Route Lookup Based on Caching Technique / 369
13.3	IP Route Lookup Based on Standard Trie Structure / 369
13.4	Patricia Tree / 372
13.5	Small Forwarding Tables for Fast Route Lookups / 373
13.5.1	Level 1 of data structure / 374
13.5.2	Levels 2 and 3 of data structure / 376
13.5.3	Performance / 377
13.6	Route Lookups in Hardware at Memory Access Speeds / 377
13.6.1	The DIR-24-8-BASIC scheme / 378
13.6.2	Performance / 381
13.7	IP Lookups Using Multiway Search / 381
13.7.1	Adapting binary search for best matching prefix / 381
13.7.2	Precomputed 16-bit prefix table / 384
13.7.3	Multiway binary search: exploiting the cache line / 385
13.7.4	Performance / 388
13.8	IP Route Lookups for Gigabit Switch Routers / 388
13.8.1	Lookup algorithms and data structure construction / 388
13.8.2	Performance / 395
13.9	IP Route Lookups Using Two-Trie Structure / 396
13.9.1	IP route lookup algorithm / 397
13.9.2	Prefix update algorithms / 398
13.9.3	Performance / 403
	References / 404

APPENDIX SONET AND ATM PROTOCOLS	407
A.1 ATM Protocol Reference Model / 409	
A.2 Synchronous Optical Network (SONET) / 410	
A.2.1 SONET sublayers / 410	
A.2.2 STS-N signals / 412	
A.2.3 SONET overhead bytes / 414	
A.2.4 Scrambling and descrambling / 417	
A.2.5 Frequency justification / 418	
A.2.6 Automatic protection switching (APS) / 419	
A.2.7 STS-3 versus STS-3c / 421	
A.2.8 OC-N multiplexer / 422	
A.3 Sub-Layer Functions in Reference Model / 423	
A.4 Asynchronous Transfer Mode (ATM) / 425	
A.4.1 Virtual path/virtual channel identifier (VPI/VCI) / 426	
A.4.2 Payload type identifier (PTI) / 427	
A.4.3 Cell loss priority (CLP) / 428	
A.4.4 Pre-defined header field values / 428	
A.5 ATM Adaptation Layer (AAL) / 429	
A.5.1 AAL type 1 (AAL1) / 431	
A.5.2 AAL type 2 (AAL2) / 433	
A.5.3 AAL types 3/4 (AAL3/4) / 434	
A.5.4 AAL type 5 (AAL5) / 436	
References / 438	

INDEX	439
--------------	------------

PREFACE

This packet switching book mainly targets high-speed packet networking. As Internet traffic grows exponentially, there is a great need to build multi-terabit Internet protocol (IP) routers, asynchronous transfer mode (ATM) switches, multiprotocol label switch (MPLS) switches, and optical switches.

Packet switching technologies have been investigated and researched intensively for almost two decades, but there are very few appropriate textbooks describing it. Many engineers and students have to search for technical papers and read them in an ad hoc manner. This book is the first that explains packet switching concepts and implementation technologies in broad scope and great depth.

This book addresses the basics, theory, architectures, and technologies to implement ATM switches, IP routers, and optical switches. The book is based on the material that Jonathan has been teaching to the industry and universities for the past decade. He taught a graduate course “Broadband Packet Switching Systems” at Polytechnic University, New York, and used the draft of the book as the text. The book has incorporated feedback from both industry people and college students.

The fundamental concepts and technologies of packet switching described in the book are useful and practical when designing IP routers, packet switches, and optical switches. The basic concepts can also stand by themselves and are independent of the emerging network platform, for instance, IP, ATM, MPLS, and IP over wavelength-division multiplexing (WDM).

ATM switching technologies have been widely used to achieve high speed and high capacity. This is because ATM uses fixed-length cells and the switching can be implemented at high speed with synchronous hardware

logics. Although most of low-end to medium-size IP routers do not use the same hardware-based technologies as those of ATM switches, next-generation backbone IP routers will use the ATM switching technologies (although the cell size in the switch core of IP routers may be different from that of ATM cells). The switching technologies described in this book are common to both ATM switches and IP routers. We believe that the book will be a practical guide to understand ATM switches and IP routers.

AUDIENCE

This book can be used as a reference book for industry people whose job is related to ATM/IP/MPLS networks. Engineers from network equipment and service providers can benefit from the book by understanding the key concepts of packet switching systems and key techniques of building a high-speed and high-capacity packet switch. This book is also a good text for senior and graduate students in electrical engineering, computer engineering, and compute science. Using it, students will understand the technology trend in packet networks so that they can better position themselves when they graduate and look for jobs in the high-speed networking field.

ORGANIZATION OF THE BOOK

The book is organized as follows.

- Chapter 1 introduces the basic structure of ATM switching systems and IP routers. It discusses the functions of both systems and their design criteria and performance requirements.
- Chapter 2 classifies packet switching architectures into different categories and compares them in performance and implementation complexity. It also covers terminologies, concepts, issues, solutions, and approaches of designing packet switches at a high level so that readers can grasp the basics before getting into the details in the following chapters.
- Chapter 3 discusses the fundamentals of input-buffered switches. Switches with input and output buffering are also described in this chapter. We show the problems of input-buffered switches, and present the techniques and algorithms that have been proposed to tackle the problems.
- Chapter 4 discusses the shared-memory switches, which have been widely used in industry because of their high performance and small buffers. We describe the operation principles of the shared-memory switches in detail.

- Chapter 5 discusses banyan-family switches, which have attracted many researchers for more than two decades as components of interconnection networks. We discuss the theory of the nonblocking property of Batcher–banyan switches and describe several example architectures in detail.
- Chapter 6 discusses several switches based on the knockout principle. Their implementation architectures are described in detail.
- Chapter 7 describes a scalable multicasting switch architecture and a fault-tolerant switch. The latter is very important for a reliable network but has not been received much attention. We discuss the architectures and algorithms for building such switches.
- Chapter 8 discusses a scalable crosspoint-buffered switch architecture with a distributed-contention control scheme. We also describe how to support multiple quality-of-service (QoS) classes in the switch.
- Chapter 9 discusses an input–output-buffered switch, called the tandem-crosspoint switch, that fully utilizes current CMOS technologies.
- Chapter 10 discusses multi-stage Clos-network switches, which are attractive because of their scalability. It presents the properties of Clos networks and introduces several routing algorithms in the Clos network.
- Chapter 11 describes optical switch architectures in both all-optical and optoelectronic approaches. Several design examples are described.
- Chapter 12 introduces mobility-support ATM switches. It also discusses wireless ATM protocols and surveys several proposed wireless ATM systems.
- Chapter 13 discusses fast IP route lookup approaches, which have been proposed over the past few years. Their performance and implementation complexity are described.

ACKNOWLEDGMENTS

This book could not have been published without the help of many people. We thank them for their efforts in improving the quality of the book. We have done our best to accurately describe broadband packet switching technologies. If any errors are found, please send an email to chao@poly.edu. We will correct them in future editions.

The entire manuscript draft was reviewed by Dr. Aleksandra Smiljanic (AT & T Laboratories), Dr. Li-Sheng Chen (Alcatel), Dr. Kurimoto Takashi (NTT), Dr. Soung-Yue Liew, and Dr. Zhigang Jing (Polytechnic University). We are immensely grateful for their critiques and suggestions.

Several chapters of the book are based on research work that was done at Polytechnic University, Chinese University of Hong Kong, and NTT. We

would like to thank several persons who contributed material to some chapters. Especially, we thank Professor Tony Lee (Chinese University of Hong Kong), Dr. Necdet Uzun (Aurora Netics, Inc.), Professor Byeong-Seog Choe (Dong Guk University), Dr. Jin-Soo Park (Coree Networks), Dr. Ti-Shiang Wang (Nokia), Dr. Heechang Kim (Telecordia), Roberto Rojas-Cessa (Coree Networks), Taweesak Kijkanjanarat (Polytechnic University), and Dr. Naoaki Yamanaka (NTT).

Jonathan wants to thank his wife, Ammie, and his children, Jessica, Roger, and Joshua, for their love, support, encouragement, patience and perseverance. He also thanks his parents for their encouragement. Cheuk would like to thank his wife, Lili, and his parents for their love and support. Eiji wishes to thank his wife, Noako, and his daughter, Kanako, for their love.

H. JONATHAN CHAO

CHEUK H. LAM

EIJI OKI

July 2001

Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and IP Routers
H. Jonathan Chao, Cheuk H. Lam, Eiji Oki
Copyright © 2001 John Wiley & Sons, Inc.
ISBNs: 0-471-00454-5 (Hardback); 0-471-22440-5 (Electronic)

BROADBAND PACKET SWITCHING TECHNOLOGIES

CHAPTER 1

INTRODUCTION

The scalable and distributed nature of the Internet continuously contributes to a wild and rapid growth of its population, including the number of users, hosts, links, and emerging applications. The great success of the Internet thus leads to exponential increases in traffic volumes, stimulating an unprecedented demand for the capacity of the core network.

Network providers therefore face the need of providing a new network infrastructure that can support the growth of traffic in the core network. Advances in fiber throughput and optical transmission technologies have enabled operators to deploy capacity in a dramatic fashion. However, the advancement in packet switch/router technologies is rather slow, so that it is still not able to keep pace with the increase in link transmission speed.

Dense-wavelength-division-multiplexing (DWDM) equipment is installed on each end of the optical fiber to multiplex wavelengths (i.e., channels) over a single fiber. For example, a 128-channel OC-192 (10 Gbit/s) DWDM system can multiplex the signals to achieve a total capacity of 1.2 Tbit/s. Several vendors are expected to enter trials for wide area DWDM networks that support OC-768 (40 Gbit/s) for each channel in the near future.

Another advanced optical technology that is being deployed in the optical network is the optical cross connect (OXC) system. Since the optical-to-electrical-to-optical conversions do not occur within the system, transmission interfaces are transparent. The OXC System is based on the microelectromechanical systems (MEMS) technology, where an array of hundreds or thousands of electrically configurable microscopic mirrors is fabricated on a

single substrate to direct light. The switching scheme is based on freely moving mirrors being rotated around micromachined hinges with submillisecond switching speed. It is rate- and format-independent.

As carriers deploy fiber and DWDM equipment to expand capacity, terabit packet switching technologies are required to aggregate high-bit-rate links while achieving higher utilization on the links. Although OXC systems have high-speed interfaces (e.g., 10 or 40 Gbit/s) and large switching capacity (e.g., 10–40 Tbit/s), the granularity of the switching is coarse, e.g., 10 or 40 Gbit/s. As a result, it is required to have high-speed and large-capacity packet switches/routers to aggregate lower-bit-rate traffic to 10 or 40 Gbit/s links. The aggregated traffic can be delivered to destinations through DWDM transmission equipment or OXC systems. The terabit packet switches that are critical elements of the Internet network infrastructure must have switch fabric capable of supporting terabit speeds to eliminate the network bottlenecks. Core terabit switches/routers must also deliver low latency and guaranteed delay variance to support real-time traffic. As a result, quality-of-service (QoS) control techniques, such as traffic shaping, packet scheduling, and buffer management, need to be incorporated into the switches/routers.

Asynchronous transfer mode (ATM) is revolutionizing the telecommunications infrastructure by transmitting integrated voice, data, and video at very high speed. The current backbone network mainly consists of ATM switches and IP routers, where ATM cells and IP packets are carried on an optical physical layer such as the Synchronous Optical Network (SONET). ATM also provides different QoS requirements for various multimedia services. Readers who are interested in knowing the SONET frame structure, the ATM cell format, and the functions associated with SONET/ATM layers are referred to the Appendix.

Along with the growth of the Internet, IP has become the dominant protocol for data traffic and is making inroads into voice transmission as well. Network providers recognize the cost savings and performance advantages of converging voice, data, and video services onto a common network infrastructure, instead of an overlayed structure. Multi-protocol label switching (MPLS) is a new technology combining the advantageous features of the ATM network, short labels and explicit routing, and the connectionless datagram of the IP network. The MPLS network also provides traffic engineering capability to achieve bandwidth provisioning, fast restoration, load balancing, and virtual private network (VPN) services. The so-called label switching routers (LSRs) that route packets can be either IP routers, ATM switches, or frame relay switches. In this book, we will address the issues and technologies of building a scalable switch/router with large capacity, e.g., several terabits per second.

In the rest of this chapter, we briefly describe the ATM network, ATM switch systems, IP router systems, and switch design criteria and performance requirements.

1.1 ATM SWITCH SYSTEMS

1.1.1 Basics of ATM Networks

ATM protocol corresponds to layer 2 as defined in the open systems interconnection (OSI) reference model. ATM is connection-oriented. That is, an end-to-end connection (or *virtual channel*) needs to be set up before routing ATM cells. Cells are routed based on two important values contained in the 5-byte cell header: the *virtual path identifier* (VPI) and *virtual channel identifier* (VCI), where a virtual path consists of a number of virtual channels. The number of bits allocated for a VPI depends on the type of interface. If it is the *user network interface* (UNI), between the user and the first ATM switch, 8 bits are provided for the VPI. This means that up to $2^8 = 256$ virtual paths are available at the user access point. On the other hand, if it is the *network node interface* (NNI), between the intermediate ATM switches, 12 bits are provided for the VPI. This indicates that there are $2^{12} = 4096$ possible virtual paths between ATM switches. In both UNI and NNI, there are 16 bits for the VCI. Thus, there are $2^{16} = 65,536$ virtual channels for each virtual path.

The combination of the VPI and the VCI determines a specific virtual connection between two ends. Instead of having the same VPI/VCI for the whole routing path, the VPI/VCI is determined on a per-link basis and changes at each ATM switch. Specifically, at each incoming link to a switch node, a VPI/VCI may be replaced with another VPI/VCI at the output link with reference to a table called a *routing information table* (RIT) in the ATM switch. This substantially increases the possible number of routing paths in the ATM network.

The operation of routing cells is as follows. Each ATM switch has its own RIT containing at least the following fields: old VPI/VCI, new VPI/VCI, *output port address*, and *priority* field (optional). When an ATM cell arrives at an input line of the switch, it is split into the 5-byte header and the 48-byte payload. By using the VPI/VCI contained in the header as the old VPI/VCI value, the switch looks in the RIT for the arriving cell's new VPI/VCI. Once the match is found, the old VPI/VCI value is replaced with the new VPI/VCI value. Moreover, the corresponding output port address and priority field are attached to the 48-byte payload of the cell, before it is sent to the switch fabric. The output port address indicates to which output port the cell should be routed. There are three modes of routing operations within the switch fabric: the *unicast mode* refers to the mode in which a cell is routed to a specific output port, the *multicast mode* refers to the mode in which a cell is routed to a number of output ports, and the *broadcast mode* refers to the mode in which a cell is routed to all output ports. In the unicast mode, $\log_2 N$ bits, where N is the number of input/output ports, are sufficient to indicate any possible output port. However, in the multicast/broadcast modes, N bits, each associated with a particular output

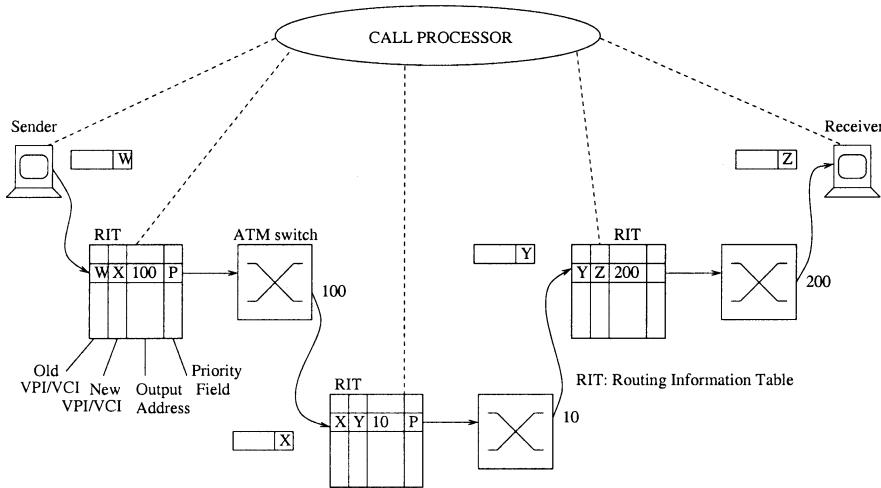


Fig. 1.1 VPI/VCI translation along the path.

port, are needed in a single-stage switch. The priority field enables the switch to selectively transmit cells to the output ports or discard them when the buffer is full, according to service requirements.

ATM connections either are preestablished through provisioning or are set up dynamically on demand using signaling, such as UNI signaling and private network–network interface (PNNI) routing signaling. The former is referred to permanent virtual connections (PVCs), while the latter is referred to switched virtual connections (SVCs). For SVCs, the RIT is updated by a *call processor* during the call setup, which finds an appropriate routing path between the source and the destination. The VPI/VCI of every link along the path, the output port addresses of the switches, and the priority field are determined and filled into the table by the call processor. The call processor has to ensure that at each switch, the VPI/VCI of the cells coming from different connections but going to the same output port are different. In practice, there is a call processor for every ATM switch. For simplicity, Figure 1.1 just shows a call processor to update the RIT of each switch in a conceptual way.

With respect to Figure 1.1, once a call setup is completed, the source starts to send a cell whose VPI/VCI is represented by W . As soon as this cell arrives at the first ATM switch, the entries of the table are searched. The matched entry is found with a new VPI/VCI X , which replaces the old VPI/VCI W . The corresponding output port address (whose value is 100) and the priority field are attached to the cell so that the cell can be routed to output port 100 of the first switch. At the second ATM switch, the VPI/VCI of the cell whose value is X is updated with a new value Y . Based on the output port address obtained from the table, the incoming cell is routed to

output port 10. This operation repeats in other switches along the path to the destination. Once the connection is terminated, the call processor deletes the associated entries of the routing tables along the path.

In the multicast case, a cell is replicated into multiple copies and each copy is routed to an output port. Since the VPI/VCI of each copy at the output port can be different, VPI/VCI replacement usually takes place at the output instead of the input. As a result, the routing table is usually split into two parts, one at the input and the other at the output. The former has two fields in the RIT: the old VPI/VCI and the N -bit routing information. The latter has three fields in the RIT: the input port number, the old VPI/VCI, and the new VPI/VCI. The combination of the input port number and the old VPI/VCI can uniquely identify the multicast connection and is used as an index to locate the new VPI/VCI at the output. Since multiple VPI/VCIs from different input ports can merge to the same output port and have the identical old VPI/VCI value, it thus has to use extra information as part of the index for the RIT. Using the input port number is a natural and easy way.

1.1.2 ATM Switch Structure

Figure 1.2(a) depicts a typical ATM switch system model, consisting of input port controllers (IPCs), a switch fabric, and output port controllers (OPCs). In practice, the IPC and the OPC are usually built on the same printed circuit board, called the line interface card (LIC). Multiple IPCs and OPCs can be built on the same LIC. The center switch fabric provides interconnections between the IPCs and the OPCs. Figure 1.2(b) shows a chassis containing a power supply card, a CPU card to perform operation, administration, and maintenance (OAM) functions for the switch system, a switch fabric card, and multiple LICs. Each LIC has a transmitter (XMT) and a receiver (RCV).

As shown in Figure 1.3(a), each IPC terminates an incoming line and extracts cell headers for processing. In this example, optical signals are first converted to electrical signals by an optical-to-electrical (O/E) converter and then terminated by an SONET framer. Cell payloads are stored in a first-in, first-out (FIFO) buffer, while headers are extracted for routing processing. Incoming cells are aligned before being routed in the switch fabric, which greatly simplifies the design of the switch fabric. The cell stream is slotted, and the time required to transmit a cell across to the network is a time slot.

In Figure 1.3(b), cells coming from the switch fabric are stored in a FIFO buffer.¹ Routing information (and other information such as a priority level,

¹For simplicity, here we use a FIFO for the buffer. To meet different QoS requirements for different connections, some kind of scheduling policies and buffer management schemes may be applied to the cells waiting to be transmitted to the output link. As a result, the buffer may not be as simple as a FIFO.

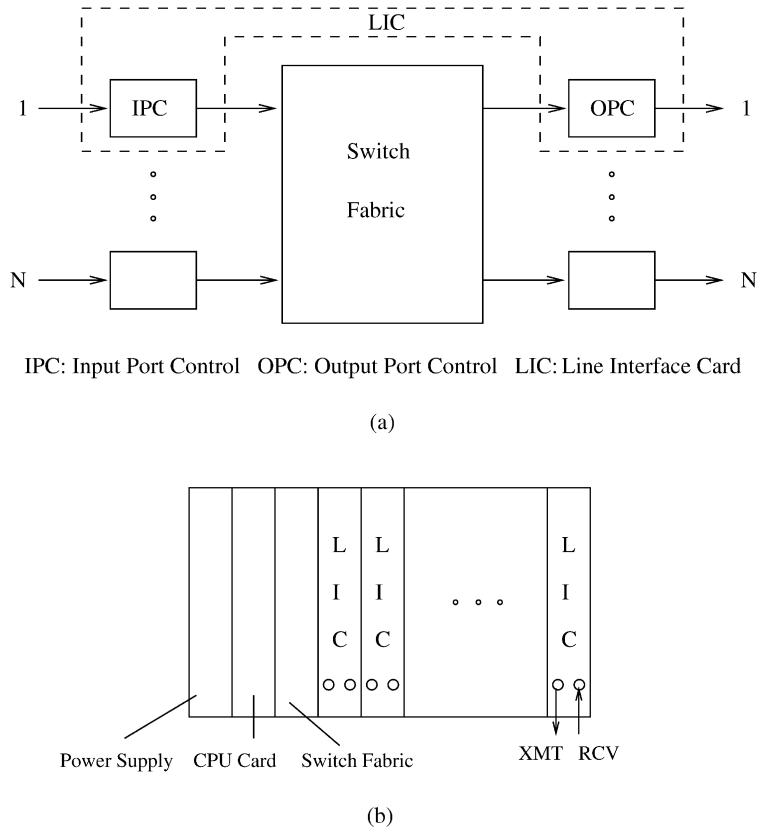


Fig. 1.2 An ATM switch system example.

if any) will be stripped off before cells are written to the FIFO. Cells are then carried in the payload of SONET frames, which are then converted to optical signals through an electrical-to-optical (E/O) converter.

The OPC can transmit at most one cell to the transmission link in each time slot. Because cells arrive randomly in the ATM network, it is likely that more than one cell is destined for the same output port. This event is called output port contention (or conflict). One cell will be accepted for transmission, and the others must be either discarded or buffered. The location of the buffers not only affects the switch performance significantly, but also affects the switch implementation complexity. The choice of the contention resolution techniques is also influenced by the location of the buffers.

There are two methods of routing cells through an ATM switch fabric: *self-routing* and *label routing*. In self-routing, an output port address field (A) is prepended to each cell at the input port before the cell enters to the switch

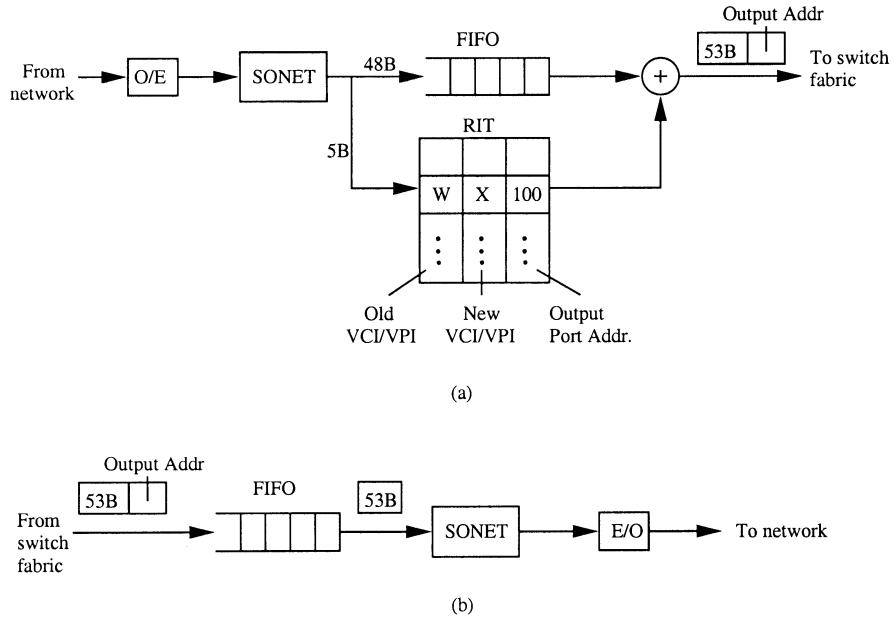


Fig. 1.3 Input and output port controller block diagram.

fabric. This field, which has $\log_2 N$ bits for unicast cells or N bits for multicast/broadcast cells, is used to navigate the cells to their destination output ports. Each bit of the output port address field is examined by each stage of the switch element. If the bit is 0, the cell is routed to the upper output of the switch element. If the bit is 1, it is routed to its lower output. As shown in Figure 1.4, a cell whose output port address is 5 (101) is routed to input port 2. The first bit of the output port address (1) is examined by the first stage of the switch element. The cell is routed to the lower output and goes to the second stage. The next bit (0) is examined by the second stage, and the cell is routed to the upper output of the switch element. At the last stage of the switch element, the last bit (1) is examined and the cell is routed to its lower output, corresponding to output port 5. Once the cell arrives at the output port, the output port address is removed.

In contrast, in label routing the VPI/VCI field within the header is used by each switch module to make the output link decision. That is, each switch module has a VPI/VCI lookup table and switches cells to an output link according to the mapping between VPI/VCI and input/output links in the table. Label routing does not depend on the regular interconnection of switching elements as self-routing does, and can be used arbitrarily wherever switch modules are interconnected.

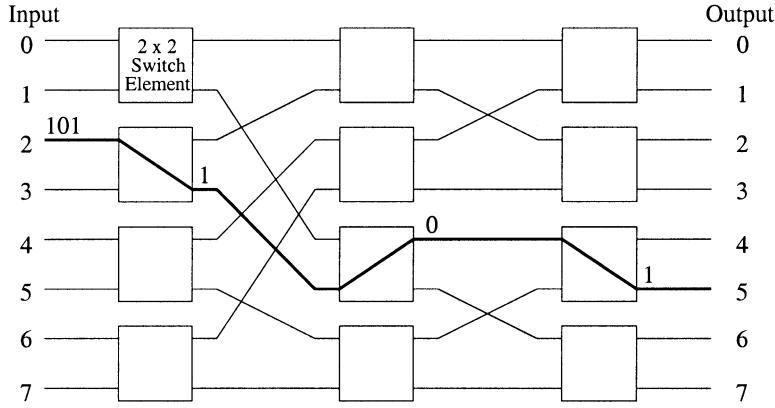


Fig. 1.4 An example of self-routing in a delta network.

1.2 IP ROUTER SYSTEMS

The Internet protocol corresponds to layer 3 as defined in the OSI reference model. The Internet, as originally conceived, offers best-effort data delivery. In contrast to ATM, which is a connection-oriented protocol, IP is a connectionless protocol. A user sends IP packets to IP networks without setting up any connection. As soon as a packet arrives at an IP router, the router decides to which output link the packet is to be routed, based on its IP address in the packet overhead, and transmits it to the output link.

To meet the demand for QoS for IP networks, the Internet Engineering Task Force (IETF) has proposed many service models and mechanisms, including the integrated services/resource reservation protocol (RSVP) model, the differentiated services (DS) model, MPLS, traffic engineering, and constraint-based routing [2]. These models will enhance today's IP networks to support a variety of service applications.

1.2.1 Functions of IP Routers

IP routers' functions can be classified into two categories, datapath functions and control functions [1].

The datapath functions are performed on every datagram that passes through the router. These include the forwarding decision, switching through the backplane, and output link scheduling. When a packet arrives at the forwarding engine, its destination IP address is first masked by the subnet mask (logical AND operation), and the resulted address is used to look up the forwarding table. A so-called longest-prefix matching method is used to find the output port. In some application, packets are classified based on the combined information of IP source/destination addresses, transport layer

port numbers (source and destination), and type of protocol: total 104 bits. Based on the result of classification, packets may be either discarded (firewall application) or handled at different priority levels. Then, the time-to-live (TTL) value is decremented and a new header checksum is calculated. Once the packet header is used to find an output port, the packet is delivered to the output port through a switch fabric. Because of contention by multiple packets destined for the same output port, packets are scheduled to be delivered to the output port in a fair manner or according to their priority levels.

The control functions include the system configuration, management, and exchange of routing table information. These are performed relatively infrequently. The route controller exchanges the topology information with other routers and constructs a routing table based on a routing protocol (e.g., RIP and OSPF). It can also create a forwarding table for the forwarding engine. Since the control function is not processed for each arriving packet, it does not have a speed constraint and is implemented in software.

1.2.2 Architectures of IP Routers

1.2.2.1 Low-End Routers In the architecture of the earliest routers, the forwarding decision and switching function were implemented in a central CPU with a shared central bus and memory, as shown in Figure 1.5. These functions are performed based on software. Since this software-based structure is cost-effective, it is mainly used by low-end routers. Although CPU performance has improved with time, it is still a bottleneck to handle all the packets transmitted through the router with one CPU.

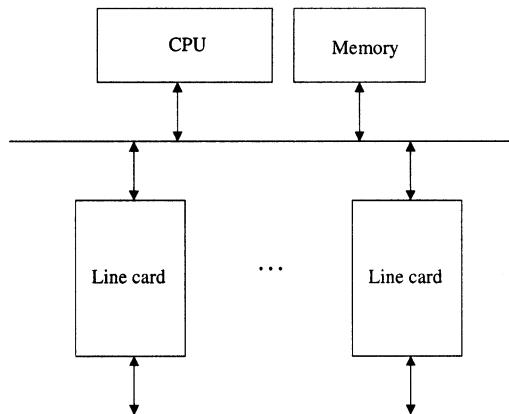


Fig. 1.5 Low-end router structure.

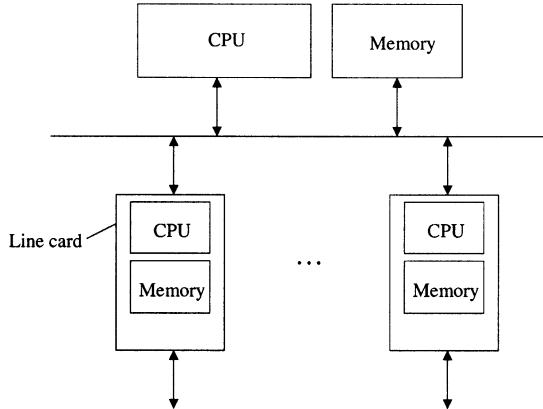


Fig. 1.6 Medium-size router structure.

1.2.2.2 Middle-Size Routers To overcome the limitation of one central CPU, medium-size routers use router structure where each line card has a CPU and memory for packet forwarding, as shown in Figure 1.6. This structure reduces the central CPU load, because incoming packets are processed in parallel by the CPU associated with each line card, before they are sent through the shared bus to the central CPU and finally to the memory of the destined output line card. However, when the number of ports and the port speed increase, this structure still has a bottleneck at the central CPU and shared bus.

1.2.2.3 High-End Routers In current high-performance routers, the forwarding engine is implemented in each associated ingress line card, and the switching function is implemented by using switch-fabric boards. As shown in Figure 1.7, a high-performance router consists of a route controller, forwarding engine, switch fabric, and output port scheduler. In this structure, multiple line cards can communicate with each other. Therefore, the router throughput is improved.

Once a packet is switched from an ingress line card to an egress line card, it is usually buffered at the output port that is implemented in the egress line card. This is because multiple packets may be destined for the same output port at the same time. Only one packet can be transmitted to the network at any time, and the rest of them must wait at the output buffer for the next transmission round. In order to provide differentiated service for different flows, it is necessary to schedule packets according to their priority levels or the allocated bandwidth. There may also be some buffer management, such as random early detection (RED), to selectively discard packets to achieve certain goals (e.g., desynchronizing the TCP flows). How to deliver packets from the forward engines to the output ports through the switch fabric is one of main topics to be discussed in this book.

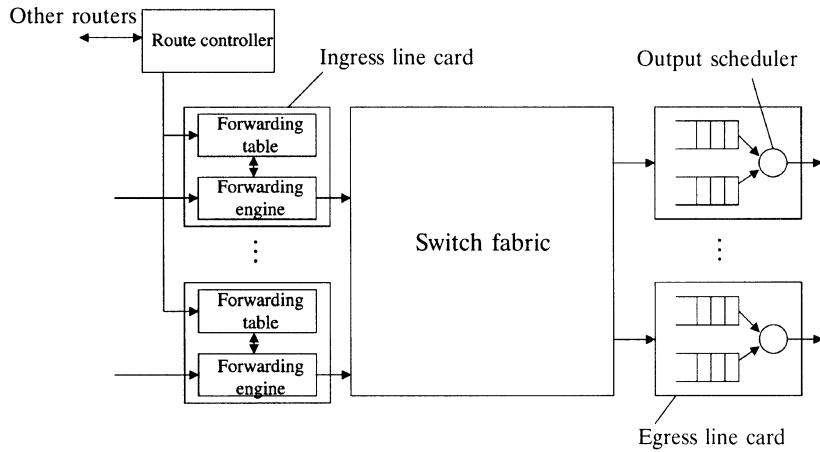


Fig. 1.7 High-end router structure.

For high-performance routers, datapath functions are most often implemented in hardware. If any datapath function cannot be completed in the interval of the minimum packet length, the router will not be able to operate at so-called wire speed, nor accommodate all arriving shortest packets at the same time.

Packets can be transferred across the switch fabric in small fixed-size data units, or as variable-length packets. The fixed-size data units are called cells (they do not have to be the same length as in ATM, 53 bytes). For the former case, each variable-length packet is segmented into cells at the input ports of the switch fabric and reassembled to that packet at the output ports. To design a high-capacity switch fabric, it is highly desirable to transfer packets in cells, for the following reasons. Due to output port contention, there is usually an arbiter to resolve contention among the input packets (except for output-buffered switches, since all packets can arrive at the same output port simultaneously—though, due to the memory speed constraint, the switch capacity is limited). For the case of delivering cells across the switch fabric, scheduling is based on the cell time slot. At the beginning of a time slot, the arbiter will determine which inputs' cells can be sent to the switch fabric. The arbitration process is usually overlapped (or pipelined) with the cell transmission. That is, while cells that are arbitrated to be winners in last time slot are being sent through the switch fabric in the current time slot, the arbiter is scheduling cells for the next time slot. From the point of view of hardware implementation, it is also easier if the decision making, data transferring, and other functions are controlled in a synchronized manner, with cell time slots.

For the case of variable-length packets, there are two alternatives. In the first, as soon as an output port completely receives a packet from an input port, it can immediately receive another packet from any input ports that

have packets destined for it. The second alternative is to transfer the packet in a cell-based fashion with the constraint that cells that belong to the same packet will be transmitted to the output port consecutively, and not interleaved with other cells. The differences between these two options have to do with event-driven vs. slot-driven arbitration. In the case of variable-length packets, the throughput is degraded, especially when supporting multicast services. Although the implementations of both alternatives are different, the basic cell transmission mechanisms are the same. Therefore, we describe an example of the variable-length-based operation by considering the first alternative for simplicity.

The first alternative starts the arbitration cycle as soon as a packet is completely transferred to the output port. It is difficult to have parallelism for arbitration and transmission, due to the lack of knowledge of when the arbitration can start. Furthermore, when a packet is destined for multiple output ports by taking advantage of the replication capability of the switch fabric (e.g., crossbar switch), the packet has to wait until all desired output ports become available. Consider a case where a packet is sent to output ports x and y . When output x is busy in accepting a packet from another input and output y is idle, the packet will not be able to send to output y until port x becomes available. As a result, the throughput for port y is degraded. However, if the packet is sent to port y without waiting for port x becomes available, as soon as port x becomes available, the packet will not be able to send to port x , since the remaining part of the packet is being sent to port y . One solution is to enable each input port to send multiple streams, which will increase the implementation complexity. However, if cell switching is used, the throughput degradation is only a cell slot time, as oppose to, when packet switching is used, the whole packet length, which can be a few tens or hundreds of cell slots.

1.2.2.4 Switch Fabric for High-End IP Routers Although most of today's low-end to medium-size routers do not use switch fabric boards, high-end backbone IP routers will have to use the switch fabric as described in Section 1.2.2.3 in order to achieve the desired capacity and speed as the Internet traffic grows. In addition, since fixed-size cell-switching schemes achieve higher throughput and simpler hardware design, we will focus on the switch architecture using cell switching rather than packet switching.

Therefore, the high-speed switching technologies described in this book are common to both ATM switches and IP routers. The terms of packet switches and ATM switches are interchangeable. All the switch architectures discussed in this book deal with fixed-length data units. Variable-length packets in IP routers, Ethernet switches, and frame relay switches, are usually segmented into fixed-length data units (not necessarily 53 bytes like ATM cells) at the inputs. These data units are routed through a switch fabric and reassembled back to the original packets at the outputs.

Differences between ATM switches and IP routers systems lie in their line cards. Therefore, both ATM switch systems and IP routers can be constructed by using a common switch fabric with appropriate line cards.

1.3 DESIGN CRITERIA AND PERFORMANCE REQUIREMENTS

Several design criteria need to be considered when designing a packet switch architecture. First, the switch should provide bounded delay and small cell loss probability while achieving a maximum throughput close to 100%. Capability of supporting high-speed input lines is also an important criterion for multimedia services, such as video conferencing and videophone. Self-routing and distributed control are essential to implement large-scale switches. Serving packets based on first come, first served provides correct packet sequence at the output ports. Packets from the same connection need to be served in sequence without causing out of order.

Bellcore has recommended performance requirements and objectives for broadband switching systems (BSSs) [3]. As shown in Table 1.1, three QoS classes and their associated performance objectives are defined: *QoS class 1*, *QoS class 3*, and *QoS class 4*. QoS class 1 is intended for stringent cell loss applications, including the circuit emulation of high-capacity facilities such as DS3. It corresponds to service class A, defined by ITU-T study group XIII. QoS class 3 is intended for low-latency, connection-oriented data transfer applications, corresponding to service class C in ITU-T study group XIII. QoS Class 4 is intended for low-latency, connectionless data transfer application, corresponding to service class D in ITU-T study group XIII.

The performance parameters used to define QoS classes 1, 3, and 4 are cell loss ratio, cell transfer delay, and two-point cell delay variation (CDV). The values of the performance objectives corresponding to a QoS class depend on the status of the cell loss priority (CLP) bit (CLP = 0 for high

TABLE 1.1 Performance Objective across BSS for ATM Connections Delivering Cells to an STS-3c or STS-12c Interface

Performance Parameter	CLP	QoS 1	QoS 3	QoS 4
Cell loss ratio	0	$< 10^{-10}$	$< 10^{-7}$	$< 10^{-7}$
Cell loss ratio	1	N/S ^a	N/S	N/S
Cell transfer delay (99th percentile) ^b	1/0	150 μ s	150 μ s	150 μ s
Cell delay variation (10^{-10} quantile)	1/0	250 μ s	N/S	N/S
Cell delay variation (10^{-7} quantile)	1/0	N/S	250 μ s	250 μ s

^aN/S not specified.

^bIncludes nonqueuing related delays, excluding propagation. Does not include delays due to processing above ATM layer.

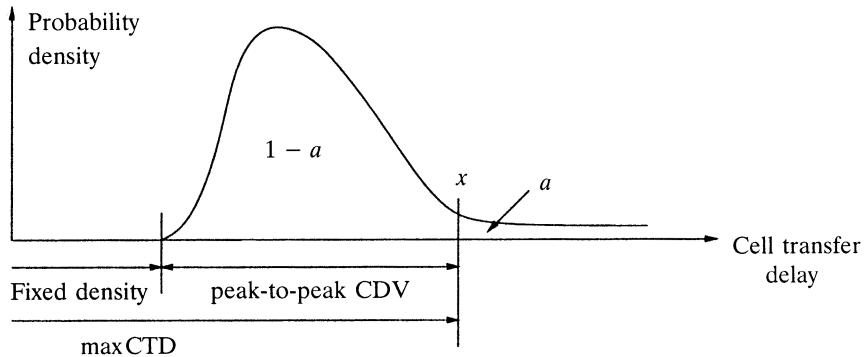


Fig. 1.8 Distribution of cell transfer delay.

priority, and CLP = 1 for low priority), which is initially set by the user and can be changed by a BSS within the connection's path.

Figure 1.8 shows a typical distribution of the cell transfer delay through a switch node. The fixed delay is attributed to the delay of table lookup and other cell header processing, such as header error control (HEC) byte examination and generation. For QoS classes 1, 3, and 4, the probability of cell transfer delay (CTD) greater than $150 \mu\text{s}$ is guaranteed to be less than $1 - 0.99$, that is, $\text{Prob}[\text{CTD} > 150 \mu\text{s}] < 1 - 99\%$. For this requirement, $a = 1\%$ and $x = 150 \mu\text{s}$ in Figure 1.8. The probability of CDV greater than $250 \mu\text{s}$ is required to be less than 10^{-10} for QoS class 1, that is, $\text{Prob}[\text{CDV} > 250 \mu\text{s}] < 10^{-10}$.

REFERENCES

1. N. McKeown, "A fast switched backplane for a gigabit switched router," *Business Commun. Rev.*, vol. 27, no. 12, Dec. 1997.
2. X. Xiao and L. M. Ni, "Internet QoS: a big picture," *IEEE Network*, pp. 8–18, March/April 1999.
3. Bellcore, "Broadband switching system (BSS) generic requirements, BSS performance," GR-110-CORE, Issue 1, Sep. 1994.

CHAPTER 2

BASICS OF PACKET SWITCHING

This chapter discusses the basic concepts in designing an ATM switch. An ATM switch has multiple input/output ports, each connecting to another port of an ATM switch or an ATM terminal. An ATM terminal can be a personal computer, a workstation, or any other equipment that has an ATM network interface card (NIC). The physical layer of interconnecting ATM switches or ATM terminals can be SONET (such as OC-3c, OC-12c, OC-48, or OC-192) or others (such as T1 or T3). Physical layer signals are first terminated and cells are extracted for further processing, such as table lookup, buffering, and switching. Cells from different places arrive at the input ports of the ATM switch. They are delivered to different output ports according to their labels, merged with other cell streams, and put into physical transmission frames. Due to the contention of multiple cells from different input ports destined for the same output port simultaneously, some cells must be buffered while the other cells are transmitted to the output ports. Thus, routing cells to the proper output ports and buffering them when they lose contention are the two major functions of an ATM switch.

Traditional telephone networks use circuit switching techniques to establish connections. In the circuit switching scheme, there is usually a centralized processor that determines all the connections between input and output ports. Each connection lasts on average 3 min. For an $N \times N$ switch, the time required to make each connection is 180 seconds divided by N . For instance, suppose N is 1000. The connection time is 180 ms, which is quite relaxed for most of switch fabrics using current technology, such as CMOS crosspoint switch chips. However, for ATM switches, the time needed to

configure the input/output connections is much more stringent, because it is based on the cell time slot. For instance, for the OC-12c interface, each cell slot time is about 700 ns. For an ATM switch with 1000 ports, the time to make the connection for each input/output is 700 ps, which is very challenging for existing technology. Furthermore, as the line bit rate increases (e.g., with OC-48c) and the number of the switch ports increases, the time needed for each connection is further reduced. As a result, it is too expensive to employ centralized processors for ATM switches.

Thus, for ATM switches we do not use a centralized connection processor to establish connections. Rather, a self-routing scheme is used to establish input/output connections in a distributed manner. In other words, the switch fabric has the ability to route cells to proper output ports, based on the physical output port addresses that are attached in the front of each cell. One switch fabric that has self-routing capability is called the banyan switch. It will be discussed in more detail in Chapter 5.

In addition to the routing function to the ATM switch, another important function is to resolve the output port contention when more than one cell is destined for the same output port at the same time. There are several contention-resolution schemes that have been proposed since the first ATM switch architecture was proposed in early 1980s (the original packet switch was called a high-speed packet switch; the term ATM was not used until late 1980). One way to resolve the contention is to allow all cells that are destined for the same output port to arrive at the output port simultaneously. Since only one cell can be transmitted via the output link at a time, most cells are queued at the output port. A switch with such architecture is called an *output-buffered* switch. The price to pay for such scheme is the need for operating the switch fabric and the memory at the output port at a rate N times the line speed. As the line speed or the switch port number increases, this scheme can have a bottleneck. However, if one does not allow all cells to go to the same output port at the same time, some kind of scheduling scheme, called an *arbitration scheme*, is required to arbitrate the cells that are destined for the same output port. Cells that lose contention will need to wait at the input buffer. A switch with such architecture is called an *input-buffered* switch.

The way of handling output contention will affect the switch performance, complexity, and implementation cost. As a result, most research on ATM switch design is devoted to output port contention resolution. Some schemes are implemented in a centralized manner, and some in a distributed manner. The latter usually allows the switch to be scaled in both line rate and switch size, while the former is for smaller switch sizes and is usually less complex and costly.

In addition to scheduling cells at the inputs to resolve the output port contention, there is another kind of scheduling that is usually implemented at the output of the switch. It schedules cell or packet transmission according to

the allocated bandwidth to meet each connection's delay/throughput requirements. Cells or packets are usually timestamped with values and transmitted with an ascending order of the timestamp values. We will not discuss such scheduling in this book, but rather focus on the scheduling to resolve output port contention.

In addition to scheduling cells to meet the delay-throughput requirement at the output ports, it is also required to implement buffer management at the input or output ports, depending on where the buffer is. The buffer management discards cells or packets when the buffer is full or exceeds some predetermined thresholds. The objectives of the buffer management are to meet the requirements on cell loss rate or fairness among the connections, which can have different or the same loss requirements. There is much research in this area to determine the discarding policies. Some of them will push out the cells or packets that have been stored in the buffer to meet the loss or fairness objectives, which is usually more complex than the scheme that blocks incoming cells when the buffer occupancy exceeds some thresholds. Again this book will focus on the switch fabric design and performance studies and will not discuss the subject of buffer management. However, when designing an ATM switch, we need to take into consideration the potential need for performing packet scheduling and buffer management. The fewer locations of the buffers on the data path, the less such control is required, and thus the smaller the implementation cost.

Section 2.1 presents some basic ATM switching concepts. Section 2.2 classifies ATM switch architectures. Section 2.3 describes performance of typical basic switches.

2.1 SWITCHING CONCEPTS

2.1.1 Internal Link Blocking

While a cell is being routed in a switch fabric, it can face a contention problem resulting from two or more cells competing for a single resource. Internal link blocking occurs when multiple cells contend for a link at the same time inside the switch fabric, as shown in Figure 2.1. This usually happens in a switch based on space-division multiplexing, where an internal physical link is shared by multiple connections among input/output ports. A *blocking* switch is a switch suffering from internal blocking. A switch that does not suffer from internal blocking is called *nonblocking*. In an internally buffered switch, contention is handled by placing buffers at the point of conflict. Placing internal buffers in the switch will increase the cell transfer delay and degrade the switch throughput.

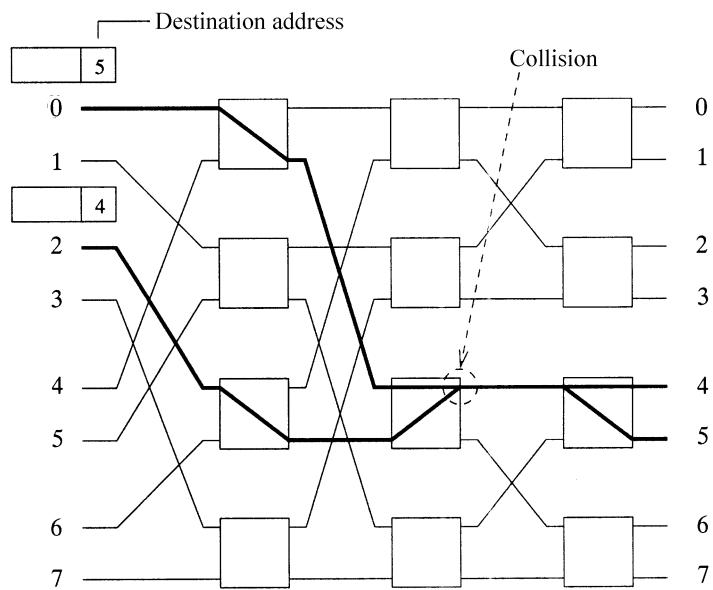


Fig. 2.1 Internal blocking in a delta network: two cells destined for output ports 4 and 5 collide.

2.1.2 Output Port Contention

Output port contention occurs when two or more cells arrive from different input ports and are destined for the same output port, as shown in Figure 2.2. A single output port can transmit only one cell in a time slot; thus the other cells must be either discarded or buffered. In the output-buffered switches, a buffer is placed at each output to store the multiple cells destined for that output port.

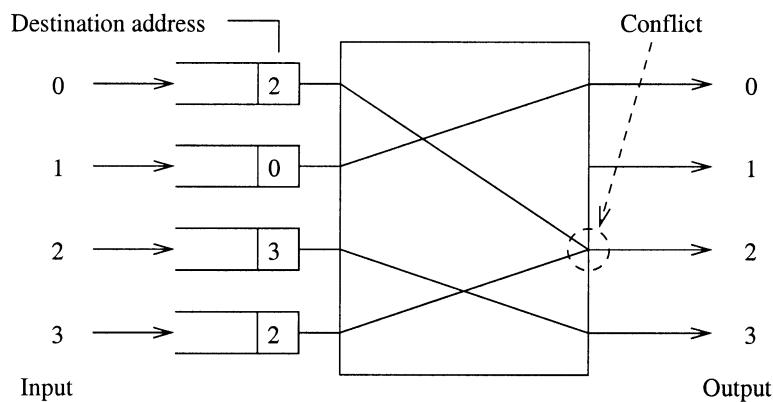


Fig. 2.2 Output port contention.

2.1.3 Head-of-Line Blocking

Another way to resolve output port contention is to place a buffer in each input port, and to select only one cell for each output port among the cells destined for that output port before transmitting the cell. This type of switch is called the input-buffered switch. An arbiter decides which cells should be chosen and which cells should be rejected. This decision can be based on cell priority or timestamp, or be random. A number of arbitration mechanisms have been proposed, such as ring reservation, sort-and-arbitrate, and route-and-arbitrate. In ring reservation, the input ports are interconnected via a ring, which is used to request access to the output ports. For switches that are based on a sorting mechanism in the switch fabric, all cells requesting the same output port will appear adjacent to each other after sorting. In the route-and-arbitrate approach, cells are routed through the switch fabric and arbiters detect contention at the point of conflict.

A well-known problem in a pure input-buffered switch with first-in-first-out (FIFO) input buffers is the *head-of-line* (HOL) blocking problem. This happens when cells are prevented from reaching a free output because of other cells that are ahead of it in the buffer and cannot be transmitted over the switch fabric. As shown in Figure 2.3, the cell behind the HOL cell at input port 0 is destined for an idle output port 1. But it is blocked by the HOL cell, which failed a transmission due to an output contention. Due to the HOL blocking, the throughput of the input buffered switch is at most 58.6% for random uniform traffic [5, 7].

2.1.4 Multicasting

To support video/audio conference and data broadcasting, ATM switches should have multicast and broadcast capability. Some switching fabrics achieve multicast by first replicating multiple copies of the cell and then routing each

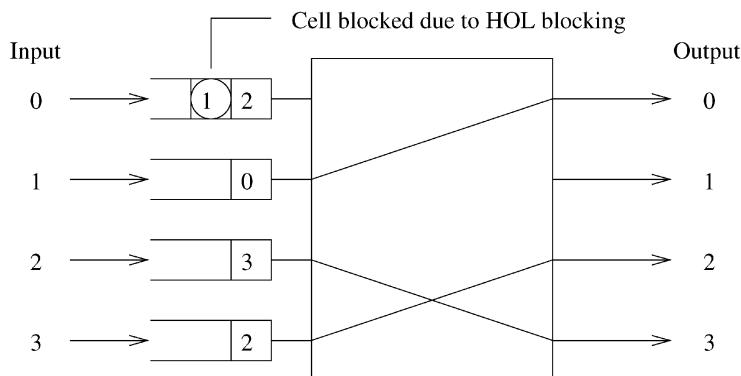


Fig. 2.3 Head-of-line blocking.

copy to their destination output ports. Other switches achieve multicast by utilizing the inherent broadcasting nature of the shared medium without generating any copies of ATM cells [3]. Details will be discussed later in the book.

2.1.5 Call Splitting

Several call scheduling disciplines have been proposed for the multicast function, such as one-shot scheduling, strict-sense call splitting, and wide-sense call splitting [1]. One-shot scheduling requires all the copies of the same cell to be transmitted in the same time slot. Strict-sense (SS) call splitting and wide-sense (WS) call splitting permit the transmission of the cell to be split over several time slots. For real-time applications, the delay difference among the receivers caused by call splitting should be bounded. A matrix is introduced to describe the operation of the scheduling algorithm, as shown in Figure 2.4, where each row (column) corresponds to an input line (output line) of the switch. With the multicast feature each row may contain two or more 1's rather than a single 1 in the unicast case. At each time slot only one cell can be selected from each column to establish a connection.

		Output				
		1	2	3	4	5
Input	1	1	1	0	0	1
	2	0	1	0	0	0
3	0	0	0	1	0	
4	0	1	1	0	0	
5	0	0	1	1	0	

Transmission requests matrix
(1: copy request, 0: no request)

X X 0 0 X	X 1 0 0 1	X 1 0 0 X
0 1 0 0 0	0 X 0 0 0	0 X 0 0 0
0 0 0 X 0	0 0 0 X 0	0 0 0 X 0
0 1 1 0 0	0 1 X 0 0	0 1 X 0 0
0 0 1 1 0	0 0 1 1 0	0 0 1 1 0

(a) One-shot (b) Strict-sense call splitting (c) Wide-sense call splitting
(X: accepted request, 1: rejected request)

Fig. 2.4 Call scheduling disciplines.

2.1.5.1 One-Shot Scheduling In a one-shot scheduling policy, all copies of the same cell must be switched successfully in one time slot. If even one copy loses the contention for an output port, the original cell waiting in the input queue must try again in the next time slot. It is obvious that this strategy favors cells with fewer copies (i.e., favors calls with fewer recipients), and more often blocks multicast cells with more copies.

2.1.5.2 Strict-Sense Call Splitting In one-shot discipline, if only one copy loses contention, the original cell (all its copies) must retry in the next time slot, thus degrading the switch's throughput. This drawback suggests transmitting copies of the multicast cell independently. In SS call splitting, at most one copy from the same cell can be transmitted in a time slot. That is, a multicast cell must wait in the input queue for several time slots until all of its copies have been transmitted. If a multicast cell has K copies to transmit, it needs at least K time slots to transmit them. Statistically this algorithm results in a low throughput when the switch is underloaded during light traffic, since the algorithm does not change dynamically with the traffic pattern.

2.1.5.3 Wide-Sense Call Splitting The case of light traffic should be considered when only one active input line has a multicast cell to be transmitted and all output ports are free. In this case, SS call splitting only allows one cell to be transmitted per time slot, resulting in a low utilization. WS call splitting was proposed to allow more than one copy from the same multicast cell to gain access to output ports simultaneously as long as these output ports are free.

It is clear that one-shot discipline has the lowest throughput performance among all three algorithms; however, it is easiest to implement. SS call splitting performs better than one-shot in the heavy traffic case, but it seems rigid in the light traffic case and results in a low utilization. WS call splitting has the advantage of the full use of output trunks, allowing the multicast cell to use all free output ports, which results in a higher throughput.

2.2 SWITCH ARCHITECTURE CLASSIFICATION

ATM switches can be classified based on their switching techniques into two groups: *time-division switching (TDS)* and *space-division switching (SDS)*. TDS is further divided into *shared-memory* type and *shared-medium* type. SDS is further divided into *single-path* switches and *multiple-path* switches, which are in turn further divided into several other types, as illustrated in Figure 2.5. In this section, we briefly describe the operation, advantages, disadvantages, and limitations inherent in each type of switch.

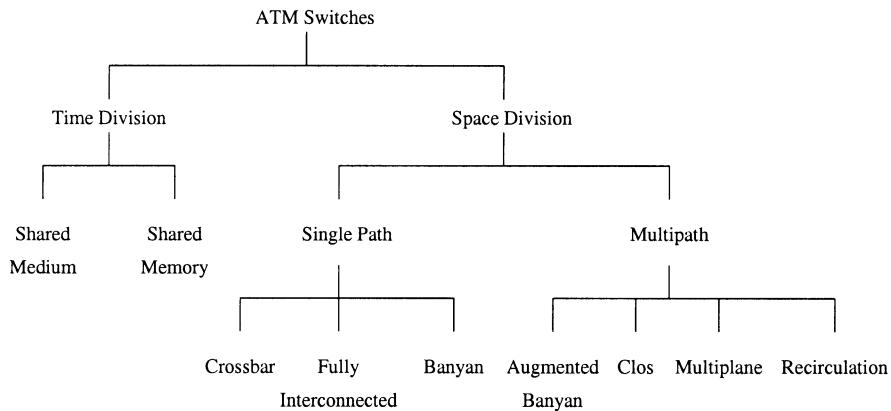


Fig. 2.5 Classification of ATM switching architectures.

2.2.1 Time-Division Switching

In TDS, there is a single internal communication structure, which is shared by all cells traveling from input ports to output ports through the switch. The internal communication structure can be a bus, a ring, or a memory. The main disadvantage of this technique is its strict capacity limitation of the internal communication structure. However, this class of switch provides an advantage in that, since every cell flows across the single communication structure, it can easily be extended to support multicast/broadcast operations.

2.2.1.1 Shared-Medium Switch In a shared-medium switch, cells arriving at input ports are time-division multiplexed into a common high-speed medium, such as a bus or a ring, of bandwidth equal to N times the input line rate. The throughput of this shared medium determines the capacity of the entire switch. As shown in Figure 2.6, each output line is connected to the shared high-speed medium via an interface consisting of an address filter (AF) and an output FIFO buffer. The AF examines the header part of the incoming cells, and then accepts only the cells destined for itself. This decentralized approach has an advantage in that each output port can operate independently and can be built separately. However, more hardware logic and more buffers are required to provide the separate interface for each output port.

A time slot is divided into N mini-slots. During each mini-slot, a cell from an input is broadcast to all output ports. This simplifies the multicasting process. A bit map of output ports with each bit indicating if the cell is routed to that output port can be attached to the front of the cell. Each AF will examine only the corresponding bit to decide if the cell should be stored

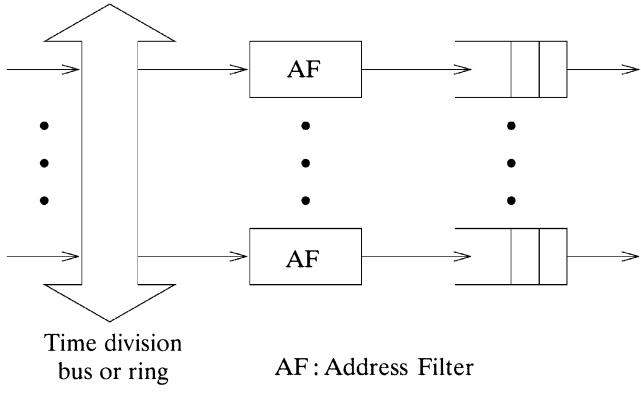


Fig. 2.6 Shared-medium switching architecture.

in the following FIFO. One disadvantage of this structure is that the switch size N is limited by the memory speed. In particular, when all N input cells are destined for the same output port, the FIFO may not be able to store all N cells in one time slot if the switch size is too large or the input line rate is too high. Another disadvantage is the lack of memory sharing among the FIFO buffers. When an output port is temporarily congested due to high traffic loading, its FIFO buffer is filled and starts to discard cells. Meanwhile, other FIFO buffers may have plenty of space but cannot be used by the congested port. As a result, a shared-memory switch, as described below, has a better buffer utilization.

Examples of shared-medium switches are NEC's ATOM (ATM output buffer modular) switch [13], IBM's PARIS (packetized automated routing integrated system) switch [2], and Fore System's ForeRunner ASX-100 switch [3].

2.2.1.2 Shared-Memory Switch In a shared-memory switch, as shown in Figure 2.7, incoming cells are time-division multiplexed into a single data stream and sequentially written to the shared memory. The routing of cells is accomplished by extracting stored cells to form a single output data stream, which is in turn demultiplexed into several outgoing lines. The memory addresses for both writing incoming cells and reading out stored cells are provided by a control module according to routing information extracted from the cell headers.

The advantage of the shared-memory switch type is that it provides the best memory utilization, since all input/output ports share the same memory. The memory size should be adjusted accordingly to keep the cell loss rate below a chosen value. There are two different approaches in sharing

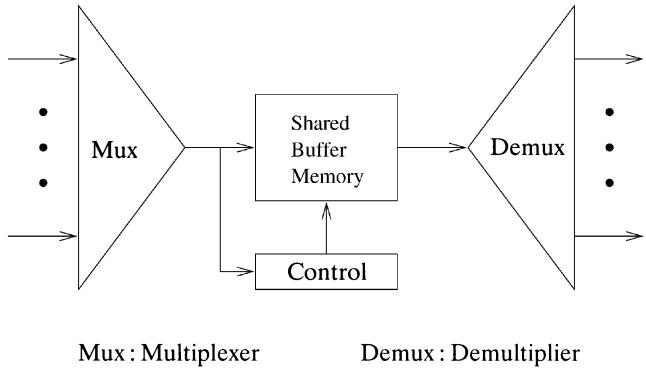


Fig. 2.7 Basic architecture of shared-memory switches.

memory among the ports: *complete partitioning* and *full sharing*. In complete partitioning, the entire memory is divided into N equal parts, where N is the number of input/output ports, and each part is assigned to a particular output port. In full sharing, the entire memory is shared by all output ports without any reservation. Some mechanisms, such as putting an upper and a lower bound on the memory space, are needed to prevent monopolization of the memory by some output ports.

Like shared-medium switches, shared-memory switches have the disadvantage that the memory access speed limits the switch size; furthermore, the control in the shared-memory switches is more complicated. Because of its better buffering utilization, however, the shared-memory type is more popular and has more variants than the shared-medium type. Detailed discussions about those variants and their implementations are given in Chapter 4.

Examples of shared-memory switches are Toshiba's 8×8 module on a single chip [12] and Hitachi's 32×32 module [8].

2.2.2 Space-Division Switching

In TDS, a single internal communication structure is shared by all input and output ports, while in SDS multiple physical paths are provided between the input and output ports. These paths operate concurrently so that multiple cells can be transmitted across the switch simultaneously. The total capacity of the switch is thus the product of the bandwidth of each path and the number of paths that can transmit cells concurrently. The total capacity of the switch is therefore theoretically unlimited. However, in practice, it is restricted by physical implementation constraints such as the device pin count, connection restrictions, and synchronization considerations.

SDS switches are classified based on the number of available paths between any input/output pair. In *single-path* switches, only one path exists for any input/output pair, while in *multiple-path* switches there is more than one. The former has simpler routing control than the latter, but the latter has higher fault tolerance.

2.2.2.1 Single-Path Switches Single path switches are classified into *crossbar-based* switches, *fully interconnected* switches, and *banyan-based* switches [10].

(a) Crossbar Switches A crossbar switch is schematically shown in Figure 2.8 for $N = 4$, where horizontal lines represent the inputs to the switch, and vertical lines represent the outputs. Basically, an $N \times N$ crossbar switch consists of a square array of N^2 individually operated crosspoints, one corresponding to each input–output pair. Each crosspoint has two possible states: cross (default) and bar. A connection between input port i and output port j is established by setting the (i, j) th crosspoint switch to the bar state while letting other crosspoints along the connection remain the cross state. The bar state of a crosspoint can be triggered individually by each incoming

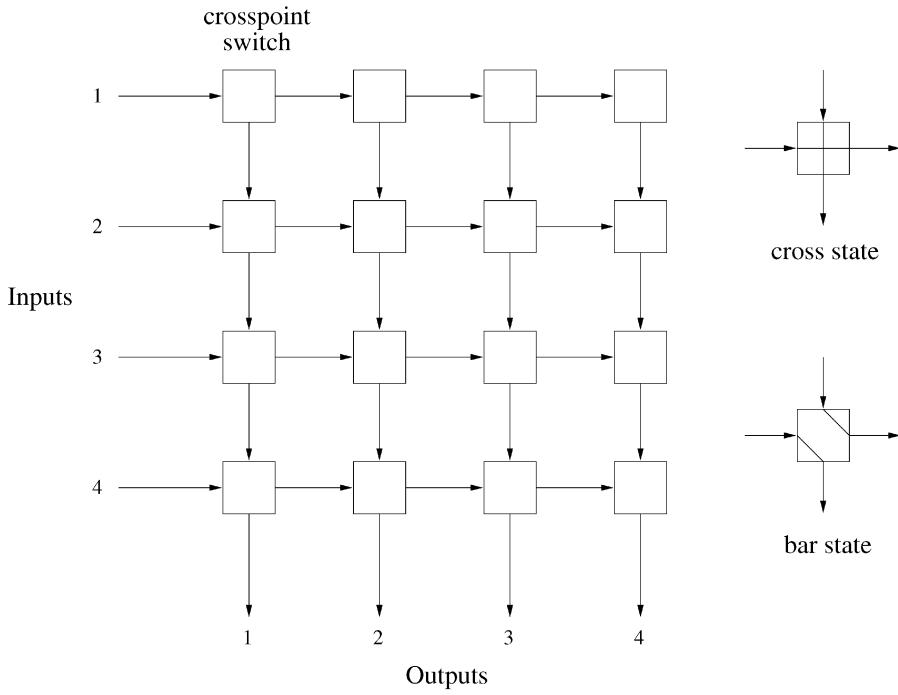


Fig. 2.8 A 4×4 crossbar switch.

cell when its destination matches with the output address. No global information about other cells and their destinations is required. This property is called the *self-routing* property; by it the control complexity is significantly reduced in the switching fabric, as the control function is distributed among all crosspoints.

Crossbar switches have three attractive properties: they are internally nonblocking, simple in architecture, and modular. However, they are complex in terms of the number of the crosspoints, which grows as N^2 . The arbitration that is to choose a winner for every output in each time slot can also become a system bottleneck as the switch size increases.

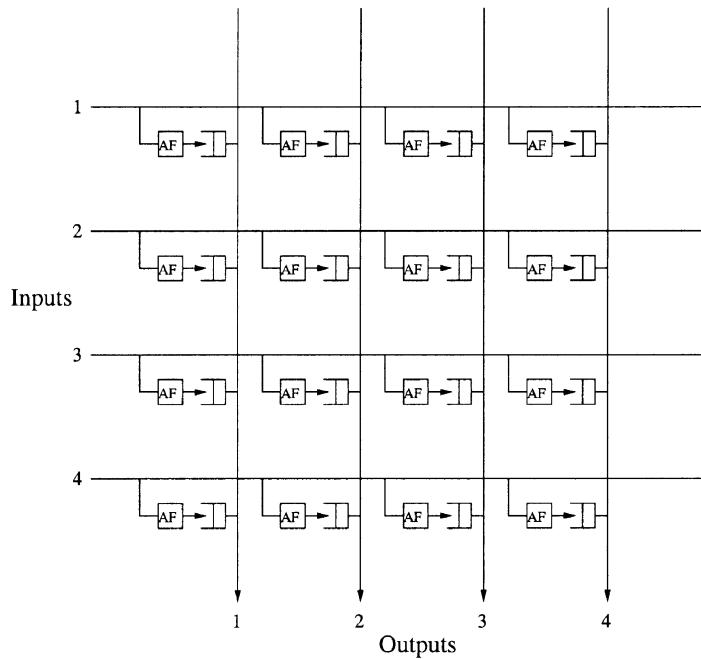
There are three possible locations for the buffers in a crossbar switch: (a) at the crosspoints in the switch fabric, (b) at the inputs of the switch, and (c) at the inputs and outputs of the switch. Each one has its advantages and disadvantages.

Figure 2.9(a) depicts the strategy of buffering cells at the crosspoints. The bus matrix switch (BMX) proposed by Fujitsu is an example of this type of switch [11]. There is an AF and a buffer at each crosspoint. The AF accepts the cells destined for the corresponding output port and stores them in the buffer. Cells waiting in the buffers on the same column are arbitrated to the output port with one cell per slot. The switch is work-conserving, and does not suffer the throughput limitation incurred with input buffering. In a sense, it is similar to achieving output queuing, with the difference that the queue for each output is distributed over N buffers. As there is no sharing among the N buffers, the total memory required for a given loss rate is greater than that required for output queuing (e.g., in the shared-medium case). As the buffer memory typically requires much more real estate in a chip than crosspoint logic, including the crosspoint buffers in the chip would severely limit the number of crosspoints in the chip.

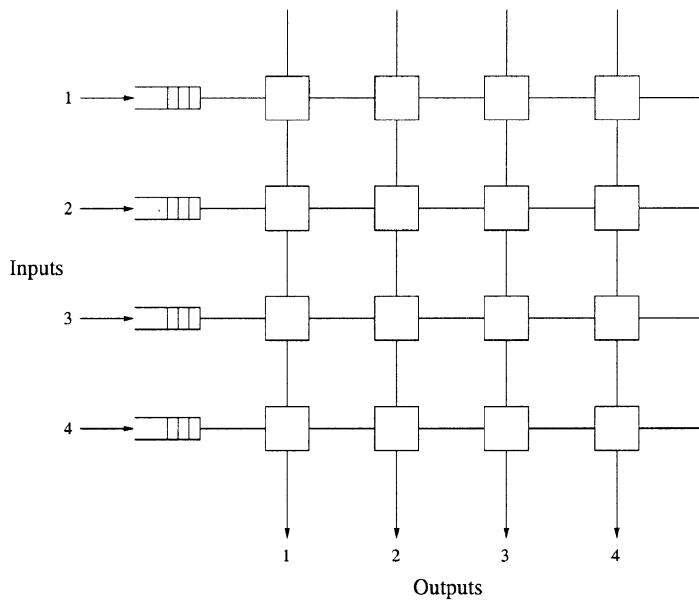
Figure 2.9(b) depicts the input queuing approach. Separating the buffers from the crosspoints is desirable from the viewpoint of layout and circuit compactness. A cell arriving at an input first enters the buffer, waiting its turn to be switched over the fabric. With distributed contention resolution, conflicts are resolved individually at crosspoints. When a cell reaches a crosspoint that has already been set by an earlier cell, or it loses contention to another contending cell, a *blocking* signal is generated and sent to the input port. This is to block the transmission of the cell and to keep the cell in the input buffer for later tries. With centralized contention resolution, alternatively, an arbiter is used for each output port to resolve contention, and only one cell destined for an output is allowed to be forwarded to the switch fabric.

The third approach combines the advantages of input buffering and output buffering. The detail is described in Section 2.2.3.

(b) Fully Interconnected Switches In a fully interconnected switch, the complete connectivity between inputs and outputs is usually accomplished by



(a) Buffering at crosspoints



(b) Buffering at the inputs

Fig. 2.9 Different buffering strategies for a crossbar switch.

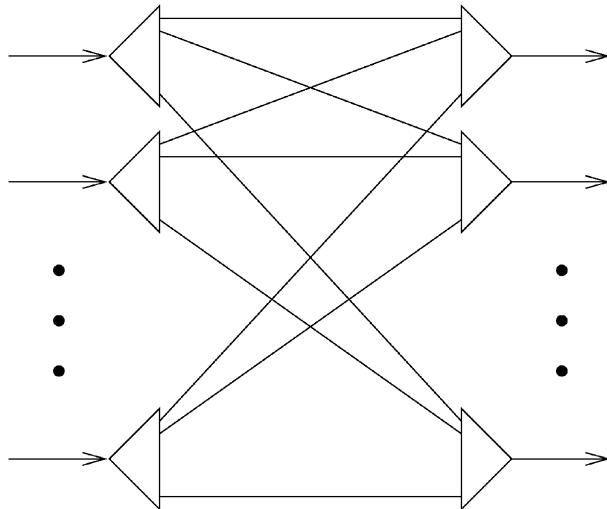


Fig. 2.10 A fully interconnected switch.

means of N separate broadcast buses from every input port to all output ports, as shown in Figure 2.10. N separate buffers are required in such a switch, one at each output port. However, if each of these N output buffers in the fully interconnected switch is partitioned and dedicated to each input line, yielding N^2 dedicated buffers, it becomes topologically identical with the crosspoint-buffered switch, and thus provides exactly the same performance and implementation complexity.

The fully interconnected switch operates in a similar manner to the shared-medium switch. A cell from any input port is broadcast to every output port. Thus, cells from several input ports can be simultaneously transmitted to the same output port. Therefore, cell filters and dedicated buffers, one for each output port, are required to filter out the misdelivered cells and to temporarily store the properly destined cells.

However, the fully interconnected switch is different from the shared-medium switch in that the speedup overhead requirement caused by sequential transmission over the shared medium is replaced by the space overhead requirement of the N^2 separate broadcast buses. This is considered a disadvantage of the switch type. The advantages of the fully interconnected switch lie in its simple and nonblocking structure, similar to the crossbar-based switch. The knockout switch is an example of this type of switch [15].

(c) *Banyan-Based Switches* Banyan-based switches are a family of self-routing switches constructed from 2×2 switching elements with a single path between any input-output pair. As shown in Figure 2.11, there are

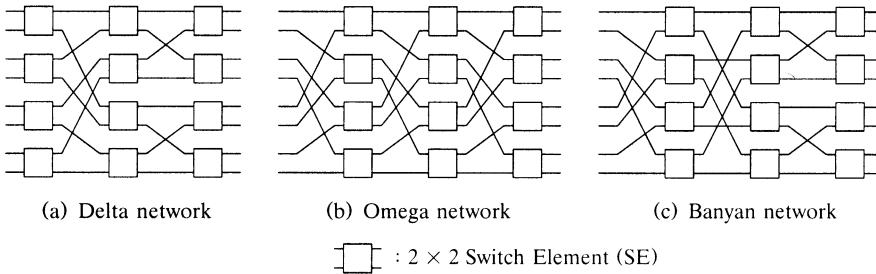


Fig. 2.11 Three different topologies of banyan-based switches.

three isomorphic topologies—*delta*, *omega*, and *banyan* networks—belonging to the banyan-based family. All of them offer equivalent performance and are discussed in detail in Chapter 5.

The banyan-based switch provides several advantages: First, it has a complexity of paths and switching elements of order $N \log N$, which makes it much more suitable than the crossbar-based and the fully interconnected switch, whose complexity is of order N^2 , for the construction of large switches. Self-routing is also an attractive feature in that no control mechanism is needed for routing cells. Routing information is contained within each cell, and it is used while the cell is routed along the path. Parallel structure of the switch provides a benefit in that several cells on different paths can be processed simultaneously. Due to their modular and recursive structure, large-scale switches can be built by using elementary switching elements without modifying their structures. This can be appropriately realized by VLSI implementation.

The main drawback of the banyan-based switch is that it is an internally blocking switch. Its performance degrades rapidly as the size of the switch increases. The performance may be improved if $M \times M$ ($M > 2$) switching elements are employed instead of 2×2 switching elements. This leads to the class of *delta-based* switches.

The delta-based switch is a family of self-routing switches constructed from $M \times M$ switching elements with a single path between any input and output port. While the performance of the delta-based switch can be significantly better than that of the banyan-based switch, it is still a blocking switch. The performance of the switch is reduced due to internal contention. This can be improved by increasing the speed of internal links within the switch with respect to that of input and output ports or by introducing buffers into the switching elements.

2.2.2.2 Multiple-Path Switches Multiple-path switches are classified as *augmented banyan* switches, *Clos* switches, *multiplane* switches, and *recirculation* switches, as shown in Figure 2.12.

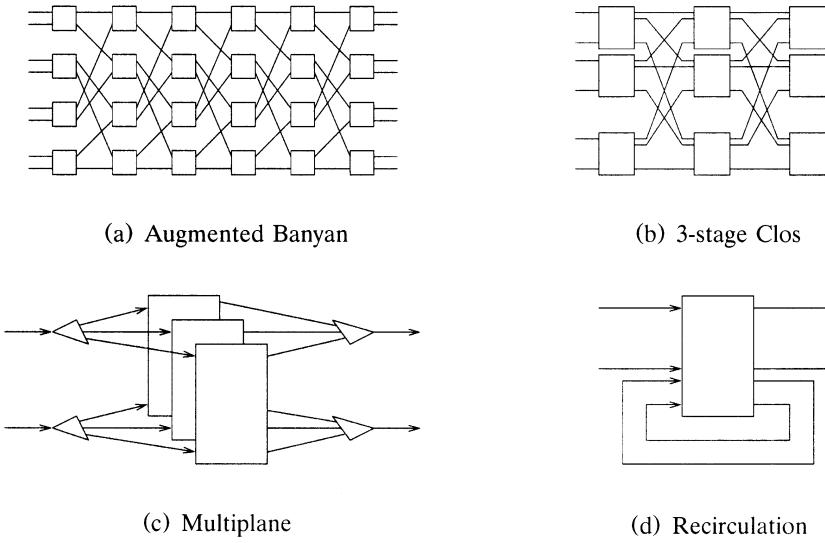


Fig. 2.12 Multiple-path space-division switches.

(a) Augmented Banyan Switches In a regular $N \times N$ banyan switch, cells pass through $\log N$ stages of switching elements before reaching their destinations. The augmented banyan switch, as illustrated in Figure 2.12(a), has more stages than the regular banyan switch. In the regular banyan switch, once a cell is deflected to an incorrect link and thus deviates from a predetermined unique path, the cell is not guaranteed to reach its requested output. Here, in the augmented banyan switch, deflected cells are provided more chances to be routed to their destinations again by using later augmented stages. When the deflected cells do not reach their destinations after the last stage, they are discarded.

The advantage of the augmented banyan switch is that by adding augmented stages, the cell loss rate is reduced. The performance of the switch is improved. The disadvantage of this switch type is its complicated routing scheme. Cells are examined at every augmented stage to determine whether they have arrived at their requested output ports. If so, they are sent to the output interface module. Otherwise, they are routed to the next stage and will be examined again. Another disadvantage is that the number of augmented stages needs to be sufficiently large. Adding each augmented stage to the switch causes increased hardware complexity. The tandem banyan switch [14] and dual shuffle exchange switch [9] are examples of the augmented banyan switches.

(b) Three-Stage Clos Switches The structure of three-stage Clos switches, as shown in Figure 2.12(b), consists of three stages of switch modules. At the

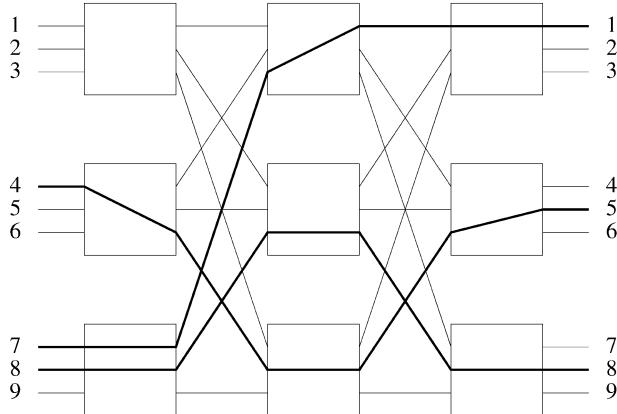


Fig. 2.13 Example of internal blocking in a three-stage Clos switch.

first stage, N input lines are broken up into r groups of n lines. Each group of lines goes into each first-stage switch module. There are m outputs in the first-stage switch module; each connects to all m middle-stage switch modules. Similarly, each middle-stage switch module has t outputs, so that it connects to all t third-stage switch modules. At the third stage, N output lines are provided as t groups of s lines.

A consideration with the three-stage Clos switch is that it may be blocking. It should be clear that a crossbar-based switch is *nonblocking*; that is, a path is always available to connect an idle input port to an idle output port. This is not always true for the three-stage Clos switch. Figure 2.13 shows a three-stage Clos switch with $N = 9$, $n = 3$, and $m = 3$. The bold lines indicate paths that are already in use. It is shown that input port 9 cannot be connected to either output port 4 or 6, even though both of these output lines are available.

By increasing the value of m (the number of outputs from each first-stage switch module or the number of middle-stage switch modules), the probability of blocking is reduced. To find the value of m needed for a nonblocking three-stage switch, let us refer to Figure 2.14.

We wish to establish a path from input port a to output port b . The worst situation for blocking occurs if all of the remaining $n - 1$ input lines and $n - 1$ output lines are busy and are connected to different middle-stage switch modules. Thus a total of $(n - 1) + (n - 1) = 2n - 2$ middle-stage switch modules are unavailable for creating a path from a to b . However, if one more middle-stage switch module exists, an appropriate link must be available for the connection. Thus, a three-stage Clos switch will be non-blocking if

$$m \geq (2n - 2) + 1 = 2n - 1.$$

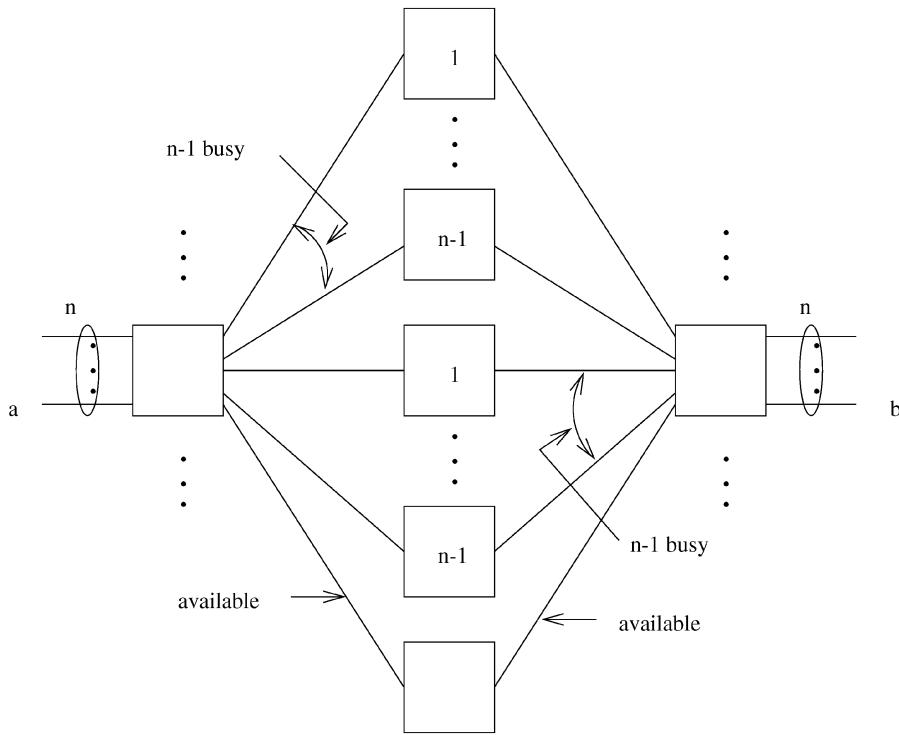


Fig. 2.14 Nonblocking condition for a three-stage Clos switch.

The total number N_x of crosspoints in a three-stage Clos switch when it is symmetric (that is, when $t = r$ and $s = n$) is

$$N_x = 2Nm + m\left(\frac{N}{n}\right)^2.$$

Substituting $m = 2n - 1$ into N_x , we obtain

$$N_x = 2N(2n - 1) + (2n - 1)\left(\frac{N}{n}\right)^2.$$

for a nonblocking switch. For a large switch size, n is large. We can approximate

$$N_x \approx 2N(2n) + 2n\left(\frac{N}{n}\right)^2 = 4Nn + 2\left(\frac{N^2}{n}\right).$$

To optimize the number of crosspoints, differentiate N_x with respect to n and set the result equal to 0. The result will be $n \approx (N/2)^{\frac{1}{2}}$. Substituting into N_x ,

$$N_x = 4\sqrt{2} N^{\frac{3}{2}} = O(N^{\frac{3}{2}}).$$

The three-stage Clos switch provides an advantage in that it reduces the hardware complexity from $O(N^2)$ in the case of the crossbar-based switch to $O(N^{\frac{3}{2}})$, and the switch can be designed to be nonblocking. Furthermore, it also provides more reliability since there is more than one possible path through the switch to connect from any input port to any output port. The main disadvantage of this switch type is that some fast and intelligent mechanism is needed to rearrange the connections in every cell time slot according to arrival cells so that internal blocking can be avoided. This will be the bottleneck when the switch size becomes large. In practice, it is difficult to avoid internal blocking although the switch itself is nonblocking. Once contention on the internal links occurs, the throughput is reduced. This can be improved by increasing the number of internal links between switch modules so that there are more paths for routing cells. Increasing the bandwidth of internal links is also helpful in that instead of having one cell for each internal link in each time slot, now more than one cell from the input module that are destined to the same third-stage module can be routed. Another way to reduce the internal blocking is routing cells in a random manner. If the center-stage switch modules have buffers, careful provision has to be made at the output ports in order to preserve cell sequencing.

(c) Multiplane Switches As shown in Figure 2.12(c), multiplane switches refer to the switches that have multiple (usually identical) switch planes. Multiplane switches are mainly proposed as a way to improve system throughput. By using some mechanisms to distribute the incoming traffic loading, cell collisions within the switches can be reduced. Additionally, more than one cell can be transmitted to the same output port by using each switch plane, so the output lines do not have to operate at higher speed than the input lines. Another advantage of multiplane switches is that they can be used for achieving reliability, since the loss of a whole switch plane will reduce the capacity but not the connectivity of the switches. However, cell sequencing may be disturbed unless cells belonging to the same connection are forced to use the same plane. The parallel banyan switch and the Sunshine switch [4] are examples of multiplane switches.

(d) Recirculation Switches Recirculation switches, as shown in Figure 2.12(d), are designed to handle the output port contention problem. By recirculating the cells that did not make it to their output ports during the current time slot back to the input ports via a set of recirculation paths,

the cell loss rate can be reduced. This results in system throughput improvement. The disadvantage of recirculation switches is that they require a large switch to accommodate the recirculation ports. Also, recirculation may cause out-of-sequence errors. Some mechanisms are needed to preserve the cell sequencing among the cells in the same connection. The best-known recirculation switches are the Starlite switch [6] and the Sunshine switch [4].

2.2.3 Buffering Strategies

In this subsection, we classify ATM switches according to their buffering strategies. Each type of switch is described, and its advantages and disadvantages are discussed.

2.2.3.1 Internally Buffered Switches Internally buffered switches are those that employ buffers within switch elements (SEs). An example of this switch type is the buffered banyan switch, as shown in Figure 2.15(a). These

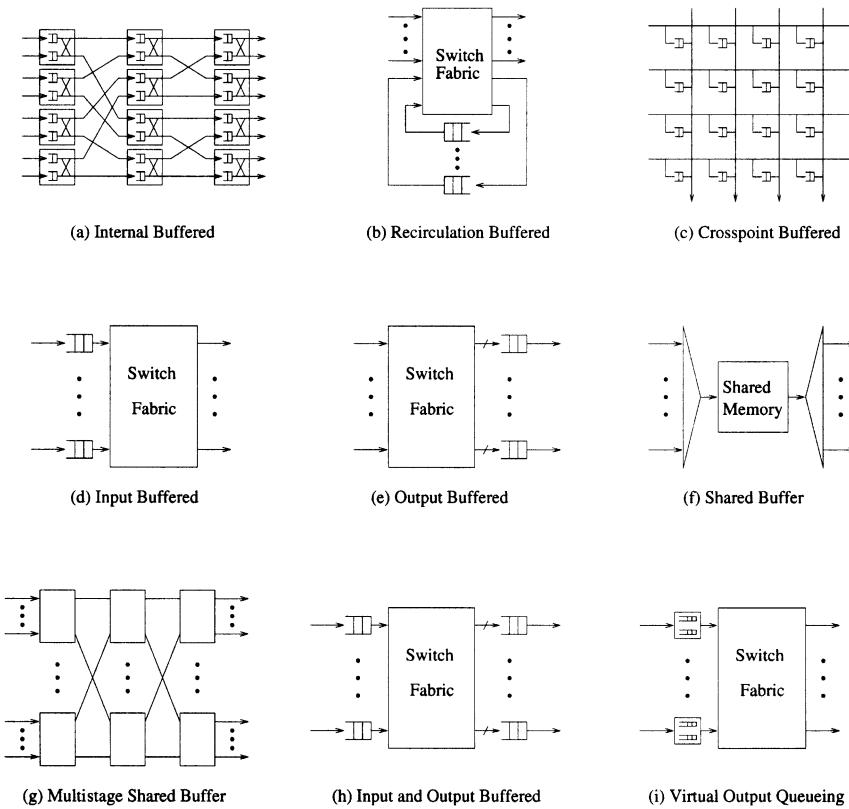


Fig. 2.15 Buffering strategies for ATM switches.

buffers are used to store internally blocked cells so that the cell loss rate can be reduced. Scalability of the switch can be easily achieved by replicating the SEs. However, this type of switch suffers from low throughput and high transfer delay that is caused by the delay from multiple stages. To meet QoS requirements, some scheduling and buffer management schemes need to be installed at the internal SEs, which will increase the implementation cost.

2.2.3.2 Recirculation Buffered Switches This type of switch is proposed to overcome the output port contention problem. As shown in Figure 2.15(b), the switch consists of both input/output ports and special ports called *recirculation* ports. When output port contention occurs, the switch allows the successful cell to go to the output port. Cells that have lost the contention are stored in a recirculation buffer and try again in the next time slot.

In order to preserve the cell sequence, a priority value is assigned to each cell. In each time slot, the priority level of the cells losing contention is increased by one so that these cells will have a better chance to be selected in the next time slot. If a cell has reached its highest priority level and the cell still has not gotten through, it will be discarded to avoid out-of-sequence errors.

The number of recirculation ports can be engineered to achieve acceptable cell loss rate. For instance, it has been shown that to achieve a cell loss rate of 10^{-6} at 80% load and Poisson arrivals, the ratio of recirculation ports to input ports must be 2.5. The Starlite switch [6] is an example of this type of switch.

The number of recirculation ports can be reduced dramatically by allowing more than one cell to arrive at the output port in each time slot. It has been shown that for a cell loss probability of 10^{-6} at 100% load and Poisson arrivals, the ratio of the recirculation ports to the input ports is reduced to 0.1 by allowing three cells arriving at each output port. The Sunshine [4] switch is an example of this switch type.

2.2.3.3 Crosspoint-Buffered Switches This type of switch, as shown in Figure 2.15(c), is the same as the crossbar-based switch discussed in Section 2.2.2.

2.2.3.4 Input-Buffered Switches The input-buffered switch, as shown in Figure 2.15(d), suffers from the HOL blocking problem and limits the throughput to 58.6% for uniform traffic. In order to increase the switch's throughput, a technique called *windowing* can be employed, where multiple cells from each input buffer are examined and considered for transmission to the output port. But at most one cell will be chosen in each time slot. The number of examined cells determines the window size. It has been shown that by increasing the window size to two, the maximum throughput is increased to 70%. Increasing the window size does not improve the maximum

throughput significantly, but increases the implementation complexity of input buffers and arbitration mechanism. This is because the input buffers cannot use simple FIFO memory any longer, and more cells need to be arbitrated in each time slot. Several techniques have been proposed to increase the throughput and are discussed in detail in Chapter 3.

2.2.3.5 Output-Buffered Switches The output-buffered switch, shown in Figure 2.15(e), allows all incoming cells to arrive at the output port. Because there is no HOL blocking, the switch can achieve 100% throughput. However, since the output buffer needs to store N cells in each time slot, its memory speed will limit the switch size. A concentrator can be used to alleviate the memory speed limitation problem so as to have a larger switch size. The disadvantage of this remedy is the inevitable cell loss in the concentrator.

2.2.3.6 Shared-Buffer Switches Figure 2.15(f) shows the shared buffer switch, which will be discussed in detail in Chapter 4.

2.2.3.7 Multistage Shared-Buffer Switches The shared-buffer architecture has been widely used to implement small-scale switches because of its high throughput, low delay, and high memory utilization. Although a large-scale switch can be realized by interconnecting multiple shared-buffer switch modules, as shown in Figure 2.15(g), the system performance is degraded due to the internal blocking. Due to different queue lengths in the first- and second-stage modules, maintaining cell sequence at the output module can be very complex and expensive.

2.2.3.8 Input- and Output-Buffered Switches Input- and output-buffered switches, as shown in Figure 2.15(h), are intended to combine the advantages of input buffering and output buffering. In input buffering, the input buffer speed is comparable to the input line rate. In output buffering, there are up to L ($1 < L < N$) cells that each output port can accept at each time slot. If there are more than L cells destined for the same output port, excess cells are stored in the input buffers instead of discarding them as in the concentrator. To achieve a desired throughput, the speedup factor L can be engineered based on the input traffic distribution. Since the output buffer memory only needs to operate at L times the line rate, a large-scale switch can be achieved by using input and output buffering. However, this type of switch requires a complicated arbitration mechanism to determine which of L cells among the N HOL cells may go to the output port.

Another kind of speedup is run the switch fabric at a higher rate than the input and output line rate. In other words, during each cell slot, there can be more than one cell transmitted from an input to an output.

2.2.3.9 Virtual-Output-Queueing Switches Virtual-output-queueing (VOQ) switches are proposed as a way to solve the HOL blocking problem encountered in the input-buffered switches. As shown in Figure 2.15(i), each

input buffer of the switch is logically divided into N *logical queues*. All these N logical queues of the input buffer share the same physical memory, and each contains the cells destined to each output port. The HOL blocking is thus reduced, and the throughput is increased. However, this type of switch requires a fast and intelligent arbitration mechanism. Since the HOL cells of all logical queues in the input buffers, whose total number is N^2 , need to be arbitrated in each time slot, that becomes the bottleneck of the switch. Several scheduling schemes for the switch using the VOQ structure are discussed in detail in Chapter 3.

2.3 PERFORMANCE OF BASIC SWITCHES

This section describes performance of three basic switches: input-buffered, output-buffered, and completely shared-buffer.

2.3.1 Input-Buffered Switches

We consider FIFO buffers in evaluating the performance of input queuing. We assume that only the cells at the head of the buffers can contend for their destined outputs. If there are more than one cell contending the same output, only one of them is allowed to pass through the switch and the others have to wait until the next time slot. When a HOL cell loses contention, at the same moment it may also block some cells behind it from reaching idle outputs. As a result, the maximum throughput of the switch is limited and cannot be 100%.

To determine the maximum throughput, we assume that all the input queues are saturated. That is, there are always cells waiting in each input buffer, and whenever a cell is transmitted through the switch, a new cell immediately replaces it at the head of the input queue. If there are k cells waiting at the heads of input queues addressed to the same output, one of them will be selected at random to pass through the switch. In other words, each of the HOL cells has equal probability ($1/k$) of being selected.

Consider all N cells at the heads of input buffers at time slot m . Depending on the destinations, they can be classified into N groups. Some groups may have more than one cell, and some may have none. For those that have more than one cell, one of the cells will be selected to pass through the switch, and the remaining cells have to stay until the next time slot. Denote by B_m^i the number of remaining cells destined for output i in the m th time slot, and by B^i the corresponding random variable in the steady state. Also, denote by A_m^i the number of cells moving to the heads of the input queues during the m th time slot and destined for output i , and by A^i the corresponding steady-state random variable. Note that a cell can only move to the head of an input queue if the HOL cell in the previous time slot was removed from that queue for transmission to an output. Hence, the state

transition of B_m^i can be represented by

$$B_m^i = \max(0, B_{m-1}^i + A_m^i - 1). \quad (2.1)$$

We assume that each new cell arrival at the head of an input queue has the equal probability $1/N$ of being destined for any given output. As a result, A_m^i has the following binomial distribution:

$$\Pr [A_m^i = k] = \binom{F_{m-1}}{k} \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{F_{m-1}-k}, \quad k = 0, 1, \dots, F_{m-1}, \quad (2.2)$$

where

$$F_{m-1} \triangleq N - \sum_{i=1}^N B_{m-1}^i. \quad (2.3)$$

F_{m-1} represents the total number of cells transmitted through the switch during the $(m-1)$ st time slot, which in saturation is equal to the total number of input queues that have a new cell moving into the HOL position in the m th time slot. That is,

$$F_{m-1} = \sum_{i=1}^N A_m^i. \quad (2.4)$$

When $N \rightarrow \infty$, A_m^i has a Poisson distribution with rate $\rho_m^i = F_{m-1}/N$. In steady state, $A_m^i \rightarrow A^i$ also has a Poisson distribution. The rate is $\rho_0 = \bar{F}/N$, where \bar{F} is the average of the number of cells passing through the switch, and ρ_0 is the utilization of output lines (i.e., the normalized switch throughput). The state transition of B^i is driven by the same Markov process as the $M/D/1$ queues in the steady state. Using the results for the mean steady-state queue size for an $M/D/1$ queue, for $N \rightarrow \infty$ we have

$$\overline{B^i} = \frac{\rho_0^2}{2(1 - \rho_0)}. \quad (2.5)$$

In the steady state, (2.3) becomes

$$\bar{F} = N - \sum_{i=1}^N \overline{B^i}. \quad (2.6)$$

By symmetry, $\overline{B^i}$ is equal for all i . In particular,

$$\overline{B^i} = \frac{1}{N} \sum_{i=1}^N \overline{B^i} = 1 - \frac{\bar{F}}{N} = 1 - \rho_0. \quad (2.7)$$

It follows from (2.5) and (2.7) that $\rho_0 = 2 - \sqrt{2} = 0.586$.

TABLE 2.1 The Maximum Throughput Achievable Using Input Queueing with FIFO Buffers

<i>N</i>	Throughput
1	1.0000
2	0.7500
3	0.6825
4	0.6553
5	0.6399
6	0.6302
7	0.6234
8	0.6184
∞	0.5858

When N is finite and small, the switch throughput can be calculated by modeling the system as a Markov chain. The numerical results are given in Table 2.1. Note that the throughput rapidly converges to 0.586 as N increases.

The results also imply that, if the input rate is greater than 0.586, the switch will become saturated, and the throughput will be 0.586. If the input rate is less than 0.586, every cell will get through the switch after some delay. To characterize the delay, a discrete-time *Geom/G/1* queuing model can be used to obtain an exact formula of the expected waiting time for $N \rightarrow \infty$.

The result is based on the following two assumptions:

1. The arrival process to each input queue is a Bernoulli process. That is, the probability that a cell arrives in each time slot is identical and independent of any other slot. We denote this probability as p , and call it the offered load.
2. Each cell is equally likely to be destined for any one output.

The *service time* for a cell at the HOL consists of the waiting time until it gets selected, plus one time slot for its transmission through the switch. As $N \rightarrow \infty$, in the steady state, the number of cells arriving at the heads of input queues and addressed to a particular output (say j) has the Poisson distribution with rate p . Hence, the service-time distribution for the discrete-time *Geom/G/1* model is the delay distribution of another queuing system: a discrete-time *M/D/1* queue with customers served in random order.

Using standard results for a discrete-time *Geom/G/1* queue, we obtain the mean cell waiting time for input queuing with FIFO buffers,

$$\bar{W} = \frac{p\bar{S}(\bar{S} - 1)}{2(1 - p\bar{S})} + \bar{S} - 1, \quad (2.8)$$

where S is the random variable of the service time obtained from the *M/D/1* model. This waiting time shown in Figure 2.16 for $N \rightarrow \infty$.

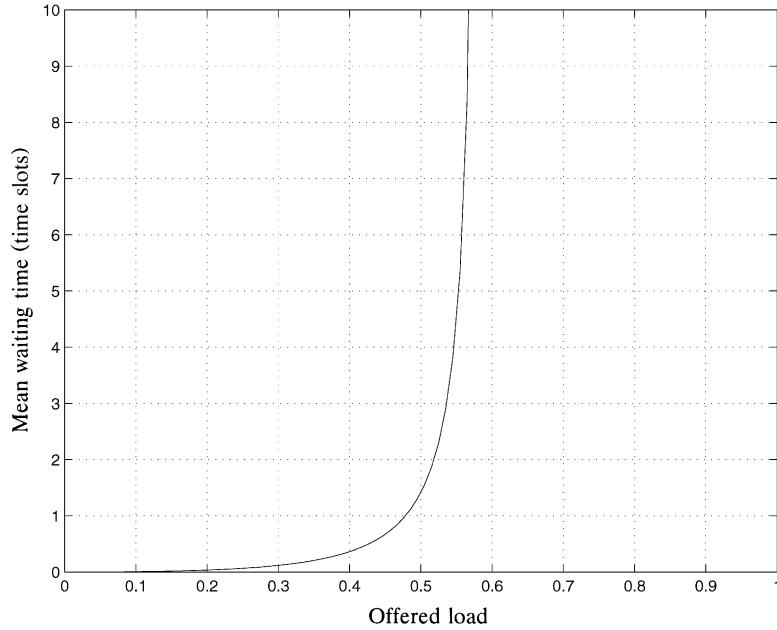


Fig. 2.16 The mean waiting time for input queuing with FIFO buffers for the limiting case for $N = \infty$.

2.3.2 Output-Buffered Switches

With output queuing, cells are only buffered at outputs, at each of which a separate FIFO is maintained. Consider a particular (i.e., tagged) output queue. Define the random variable A as the number of cell arrivals destined for the tagged output in a given time slot. Based on the same assumptions as in Section 2.3.1 on the arrivals, we have

$$a_k \triangleq \Pr[A = k] = \binom{N}{k} \left(\frac{p}{N}\right)^k \left(1 - \frac{p}{N}\right)^{N-k}, \quad k = 0, 1, 2, \dots, N \quad (2.9)$$

When $N \rightarrow \infty$, (2.9) becomes

$$a_k \triangleq \Pr[A = k] = \frac{p^k e^{-p}}{k!}, \quad k = 0, 1, 2, \dots. \quad (2.10)$$

Denote by Q_m the number of cells in the tagged queue at the end of the m th time slot, and by A_m the number of cell arrivals during the m th time slot. We have

$$Q_m = \min\{\max(0, Q_{m-1} + A_m - 1), b\}. \quad (2.11)$$

If $Q_{m-1} = 0$ and $A_m > 0$, there is no cell waiting at the beginning of the m th time slot, but we have A_m cells arriving. We assume that one of the arriving cells is immediately transmitted during the m th time slot; that is, a cell goes through the switch without any delay.

For finite N and finite b , this can be modeled as a finite-state, discrete-time Markov chain with state transition probabilities $P_{ij} \triangleq \Pr[Q_m = j | Q_{m-1} = i]$ as follows:

$$P_{ij} = \begin{cases} a_0 + a_1, & i = 0, \quad j = 0, \\ a_0, & 1 \leq i \leq b, \quad j = i - 1, \\ a_{j-i+1}, & 1 \leq j \leq b - 1, \quad 0 \leq i \leq j, \\ \sum_{m=j-i+1}^N a_m, & j = b, \quad 0 \leq i \leq j, \\ 0 & \text{otherwise,} \end{cases} \quad (2.12)$$

where a_k is given by (2.9) and (2.10) for a finite N and $N \rightarrow \infty$, respectively. The steady-state queue size can be obtained recursively from the following Markov chain balance equations:

$$\begin{aligned} q_1 &\triangleq \Pr[Q = 1] = \frac{1 - a_0 - a_1}{a_0} \cdot q_0 \\ q_n &\triangleq \Pr[Q = n] = \frac{1 - a_1}{a_0} \cdot q_{n-1} - \sum_{k=2}^n \frac{a_k}{a_0} \cdot q_{n-k}, \quad 2 \leq n \leq b, \end{aligned}$$

where

$$q_0 \triangleq \Pr[Q = 0] = \frac{1}{1 + \sum_{n=1}^b q_n/q_0}.$$

No cell will be transmitted on the tagged output line during the m th time slot if, and only if, $Q_{m-1} = 0$ and $A_m = 0$. Therefore, the switch throughput ρ_0 is represented as

$$\rho_0 = 1 - q_0 a_0.$$

A cell will be lost if, when emerging from the switch fabric, it finds the output buffer already containing b cells. The cell loss probability can be calculated as follows:

$$\Pr[\text{cell loss}] = 1 - \frac{\rho_0}{p},$$

where p is the offered load.

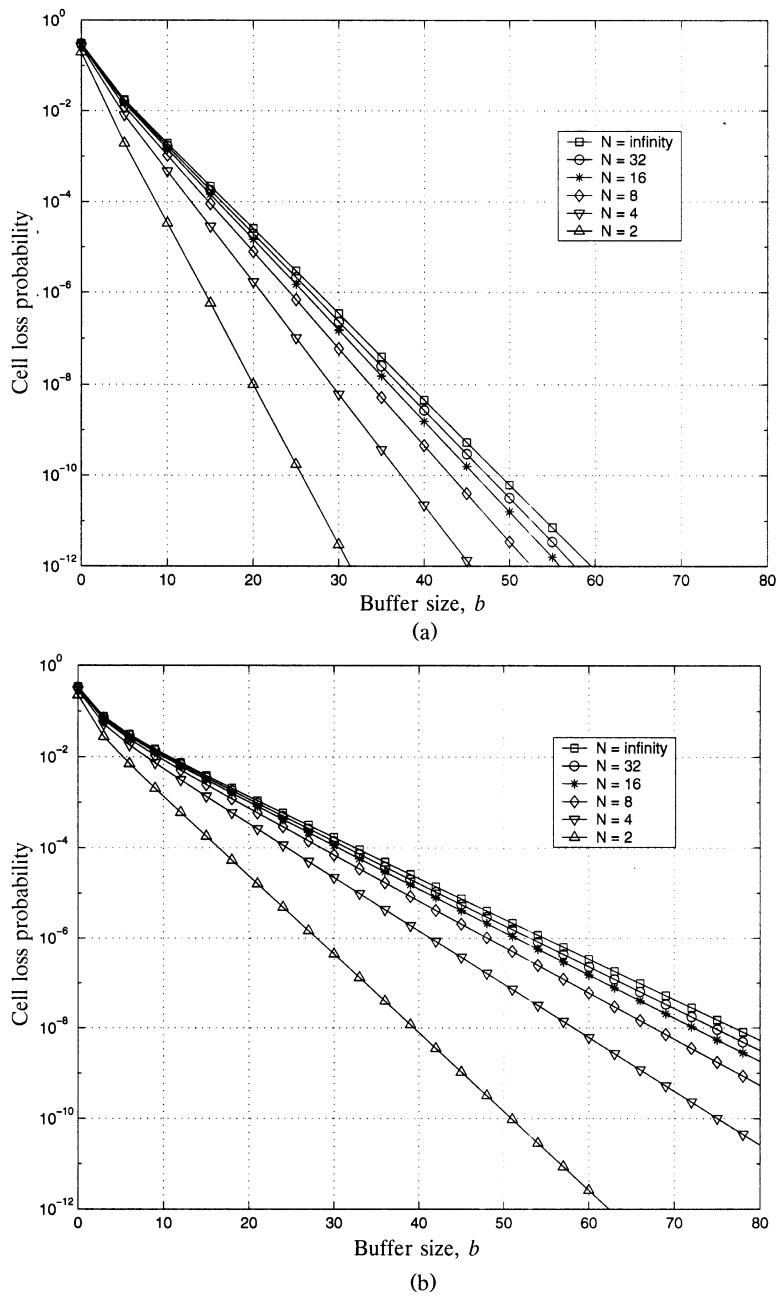


Fig. 2.17 The cell loss probability for output queuing as a function of the buffer size b and the switch size N , for offered loads (a) $p = 0.8$ and (b) $p = 0.9$.

Figure 2.17(a) and (b) show the cell loss probability for output queuing as a function of the output buffer size b for various switch size N and offered loads $p = 0.8$ and 0.9. At the 80% offered load, a buffer size of $b = 28$ is good enough to keep the cell loss probability below 10^{-6} for arbitrarily large N . The $N \rightarrow \infty$ curve can be a close upper bound for finite $N > 32$. Figure 2.18 shows the cell loss performance when $N \rightarrow \infty$ against the output buffer size b for offered loads p varying from 0.70 to 0.95.

Output queuing achieves the optimal throughput–delay performance. Cells are delayed unless it is unavoidable, when two or more cells arriving on different inputs are destined for the same output. With Little’s result, the mean waiting time \bar{W} can be obtained as follows:

$$\bar{W} = \frac{\bar{Q}}{\rho_0} = \frac{\sum_{n=1}^b nq_n}{1 - q_0 a_0}.$$

Figure 2.19 shows the numerical results for the mean waiting time as a function of the offered load p for $N \rightarrow \infty$ and various values of the output buffer size b . When $N \rightarrow \infty$ and $b \rightarrow \infty$, the mean waiting time is obtained from the $M/D/1$ queue as follows:

$$\bar{W} = \frac{p}{2(1-p)}.$$

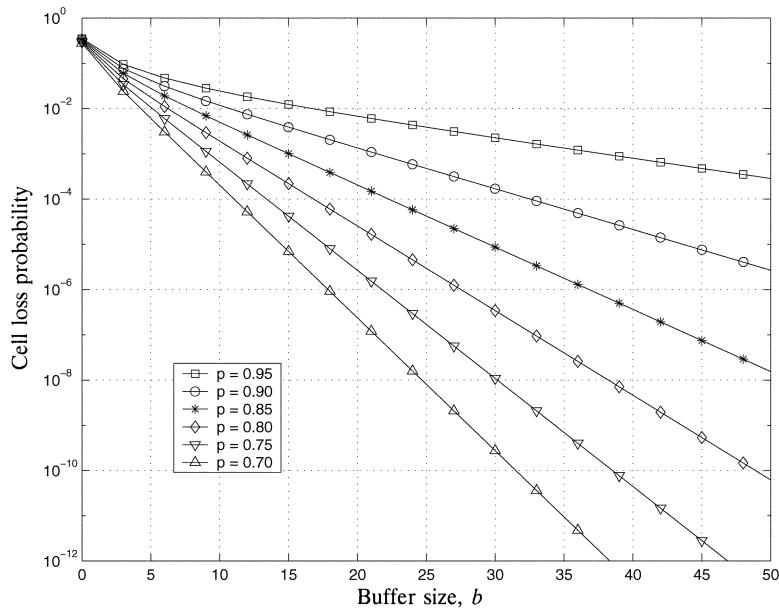


Fig. 2.18 The cell loss probability for output queuing as a function of the buffer size b and offered loads varying from $p = 0.70$ to $p = 0.95$, for the limiting case of $N \rightarrow \infty$.

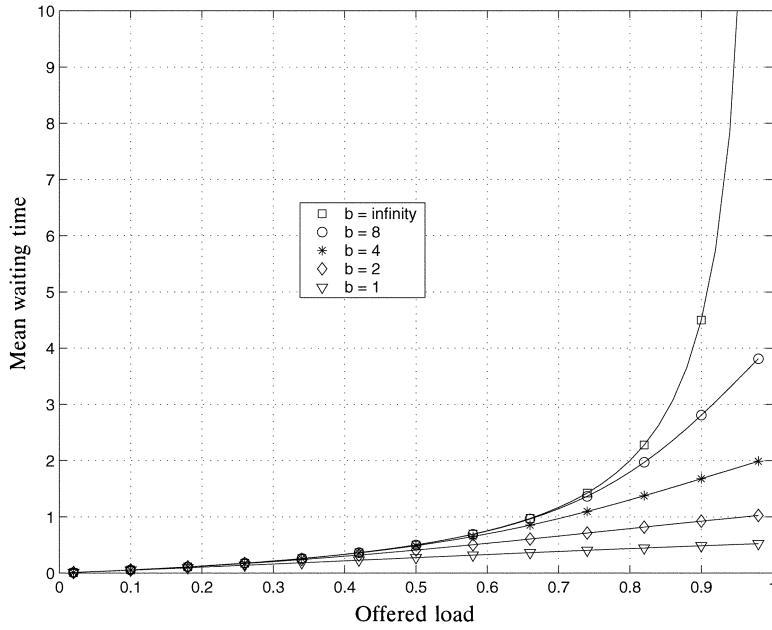


Fig. 2.19 The mean waiting time for output queuing as a function of the offered load p , for $N \rightarrow \infty$ and output FIFO sizes varying from $b = 1$ to ∞ .

2.3.3 Completely Shared-Buffer Switches

With complete buffer sharing, all cells are stored in a common buffer shared by all inputs and outputs. One can expect that it will need less buffer to achieve a given cell loss probability, due to the statistical nature of cell arrivals. Output queuing can be maintained logically with linked lists, so that no cells will be blocked from reaching idle outputs, and we still can achieve the optimal throughput–delay performance, as with dedicated output queuing.

In the following, we will take a look at how this approach improves the cell loss performance. Denote by Q_m^i the number of cells destined for output i in the buffer at the end of the m th time slot. The total number of cells in the shared buffer at the end of the m th time slot is $\sum_{i=1}^N Q_m^i$. If the buffer size is infinite, then

$$Q_m^i = \max \{0, Q_{m-1}^i + A_m^i - 1\}, \quad (2.13)$$

where A_m^i is the number of cells addressed to output i that arrive during the m th time slot.

With a finite buffer size, cell arrivals may fill up the shared buffer, and the resulting buffer overflow makes (2.13) only an approximation. However, we

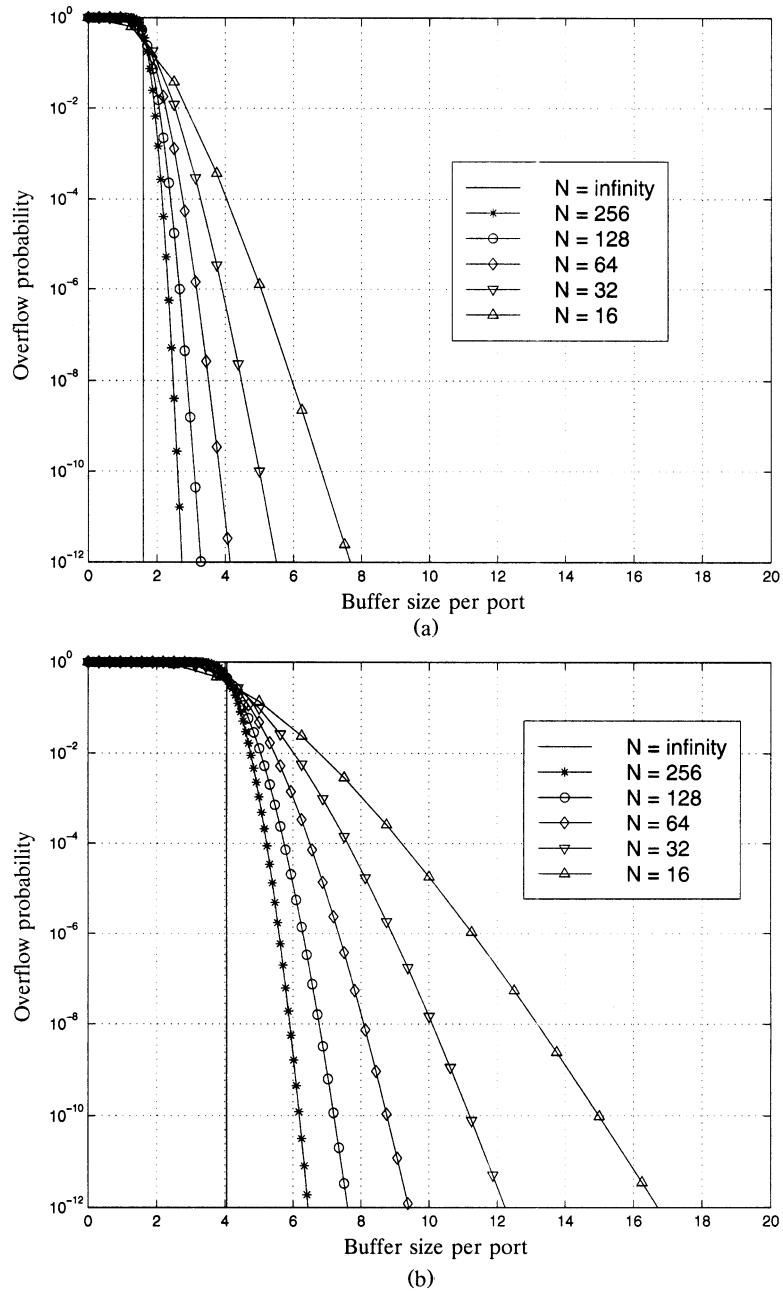


Fig. 2.20 The cell loss probability for completely shared buffering as a function of the buffer size per output, b , and the switch size N , for offered loads (a) $p = 0.8$ and (b) $p = 0.9$.

are only interested in the region of low cell loss probability (e.g., less than 10^{-6}), in which this approximation is still good.

When N is finite, A^i , which is the number of cell arrivals destined for output i in the steady state, is not independent of A^j ($j \neq i$). This is because at most N cells arrive at the switch, and a large number of cells arriving for one output implies a small number for the remaining outputs. As N goes to infinity, however, A^i becomes an independent Poisson random variable (with mean value p). Then Q^i , which is the number of cells in the buffer that are destined for output i in the steady state, also becomes independent of Q_j ($j \neq i$). We will use the Poisson and independence assumptions for finite N . These approximations are good for $N \geq 16$.

Therefore we model the steady-state distribution of $\sum_{i=1}^N Q^i$, the number of cells in the buffer, as the N -fold convolution of $N M/D/1$ queues. With the assumption of an infinite buffer size, we then approximate the cell loss probability by the overflow probability $\Pr[\sum_{i=1}^N Q^i \geq Nb]$. Figure 2.20(a) and (b) show the numerical results.

REFERENCES

1. X. Chen and J. F. Hayes, "Call scheduling in multicast packet switching," *Proc. IEEE ICC '92*, pp. 895–899, 1992.
2. I. Cidon et al., "Real-time packet switching: a performance analysis," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1576–1586, Dec. 1988.
3. FORE systems, Inc., "White paper: ATM switching architecture," Nov. 1993.
4. J. N. Giacopelli, J. J. Hickey, W. S. Marcus, W. D. Sincoskie, and M. Littlewood, "Sunshine: a high-performance self-routing broadband packet switch architecture," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1289–1298, Oct. 1991.
5. J. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Select. Areas Commun.*, vol. 5, no. 8, pp. 1264–1273, Oct. 1987.
6. A. Huang and S. Knauer, "STARLITE: a wideband digital switch," *Proc. IEEE GLOBECOM '84*, pp. 121–125, Dec. 1984.
7. M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input Versus Output Queueing on a Space Division Packet Switch," *IEEE Trans. Commun.*, vol. COM-35, No. 12, Dec. 1987.
8. T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano, "32 × 32 shared buffer type ATM switch VLSI's for B-ISDN's," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1239–1247, Oct. 1991.
9. S. C. Liew and T. T. Lee, " $N \log N$ dual shuffle-exchange network with error-correcting routing," *Proc. IEEE ICC '92*, Chicago, vol. 1, pp. 1173–1193, Jun. 1992.
10. P. Newman, "ATM switch design for private networks," *Issues in Broadband Networking*, 2.1, Jan. 1992.
11. S. Nojima, E. Tsutsui, H. Fukuda, and M. Hashimoto, "Integrated services packet network using bus-matrix switch," *IEEE J. Select. Areas Commun.*, vol. 5, no. 10, pp. 1284–1292, Oct. 1987.

12. Y. Shobatake, M. Motoyama, E. Shobatake, T. Kamitake, S. Shimizu, M. Noda, and K. Sakaue, "A one-chip scalable 8×8 ATM switch LSI employing shared buffer architecture," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1248–1254, Oct. 1991.
13. H. Suzuki, H. Nagano, T. Suzuki, T. Takeuchi, and S. Iwasaki, "Output-buffer switch architecture for asynchronous transfer mode," *Proc. IEEE ICC '89*, pp. 99–103, Jun. 1989.
14. F. A. Tobagi, T. K. Kwok, and F. M. Chiussi, "Architecture, performance and implementation of the tandem banyan fast packet switch," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1173–1193, Oct. 1991.
15. Y. S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: a simple, modular architecture for high-performance switching," *IEEE J. Select. Areas Commun.*, vol. 5, no. 8, pp. 1274–1283, Oct. 1987.

CHAPTER 3

INPUT-BUFFERED SWITCHES

When high-speed packet switches were constructed for the first time, they used either internal shared buffer or input buffer and suffered the problem of throughput limitation. As a result, most early-date research has focused on the output buffering architecture. Since the initial demand of switch capacity is at the range of a few to 10–20 Gbit/s, output buffered switches seem to be a good choice for their high delay throughput performance and memory utilization (for shared-memory switches). In the first few years of deploying ATM switches, output-buffered switches (including shared-memory switches) dominated the market. However, as the demand for large-capacity switches increases rapidly (either line rates or the switch port number increases), the speed requirement for the memory must increase accordingly. This limits the capacity of output-buffered switches. Therefore, in order to build larger-scale and higher-speed switches, people have focused on input-buffered or combined input–output-buffered switches with advanced scheduling and routing techniques, which are the main subjects of this chapter.

Input-buffered switches have two problems: (1) throughput limitation due to the head-of-line (HOL) blocking and (2) the need of arbitrating cells due to output port contention. The first problem can be circumvented by moderately increasing the switch fabric's operation speed or the number of routing paths to each output port (i.e., allowing multiple cells to arrive at the output port in the same time slot). The second problem is resolved by novel, fast arbitration schemes that will be described in this chapter. According to Moore's law, memory density doubles every 18 months. But the memory speed increases at a much slower rate. For instance, the memory speed in 2001 is 5 ns for state-of-the-art CMOS static RAM, compared with 6 ns one

or two years ago. On the other hand, the speed of logic circuits increases at a higher rate than that of memory. Recently, much research has been devoted to devising fast scheduling schemes to arbitrate cells from input ports to output ports.

Here are the factors used to compare different scheduling schemes: (1) throughput, (2) delay, (3) fairness for cells, independent of port positions, (4) implementation complexity, and (5) scalability as the line rate or the switch size increases. Furthermore, some scheduling schemes even consider per-flow scheduling at the input ports to meet the delay-throughput requirements for each flow, which of course greatly increases implementation complexity and cost. Scheduling cells on a per-flow basis at input ports is much more difficult than at output ports. For example, at an output port, cells (or packets) can be timestamped with values based on their allocated bandwidth and transmitted in ascending order of their timestamp values. However, at an input port, scheduling cells must take output port contention into account. This makes the problem so complicated that so far no feasible scheme has been devised.

A group of researchers attempted to use input-buffered switches to emulate output-buffered switches by moderately increasing the switch fabric operation speed (e.g., to twice the input line rate) together with some scheduling scheme. Although this has been shown to be possible, its implementation complexity is still too high to be practical.

The rest of this chapter is organized as follows. Section 3.1 describes a simple switch model with input buffers (optional) and output buffers, and an on-off traffic model for performance study. Section 3.2 presents several methods to improve the switch performance. The degradation is mainly caused by HOL blocking. Section 3.3 describes several schemes to resolve output port contention among the input ports. Section 3.4 shows how an input-buffered switch can emulate an output-buffered switch. Section 3.5 presents a new scheduling scheme that can achieve a delay bound for an input-buffered switch without trying to emulate an output-buffered switch.

3.1 A SIMPLE SWITCH MODEL

Figure 3.1 depicts a simple switch model where a packet switch is divided into several component blocks. In the middle there is a switch fabric that is responsible for transferring cells from inputs to outputs. We assume that the switch fabric is internally nonblocking and it takes a constant amount of time to deliver a group of conflict-free cells to their destination. Because of output contention, however, cells destined for the same output may not be able to be delivered at the same time, and some of them may have to be buffered at inputs. We also assume that each input link and output link has the same transmission speed to connect to the outside of the switch, but the internal switch fabric can have a higher speed to improve the performance. In this case, each output may require a buffer also.

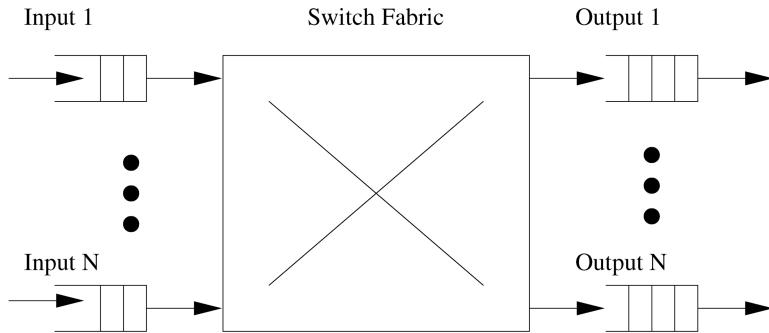


Fig. 3.1 A switch model.

3.1.1 Head-of-Line Blocking Phenomenon

A natural way to serve cells waiting at each input is first in, first out (FIFO). That is, in every time slot only the HOL cell is considered. When a HOL cell cannot be cleared due to its loss in output contention, it may block every cell behind, which may be destined for a currently idle output. In Figure 3.2, between the first two inputs, only one of the two contending HOL cells (A and B) can be cleared in this time slot. Although cell C (the second cell of the second input) is destined for a currently idle output, it cannot be cleared, because it is blocked by the HOL cell (cell B). This phenomenon is called HOL blocking, as described in Section 2.3.1. In this case, the inefficiency can be alleviated if the FIFO restriction is relaxed. For example, both cell A and cell C may be selected to go while cell B remains on the line until the next time slot. Scheduling methods to improve switch efficiency are discussed in Section 3.2.2.

However, no scheduling scheme can improve the case at the last input where only one of the two cells can be cleared in a time slot although each of them is destined for a distinct free output. Increasing internal bandwidth is the only way to improve this situation, and it is discussed in Section 3.2.1.

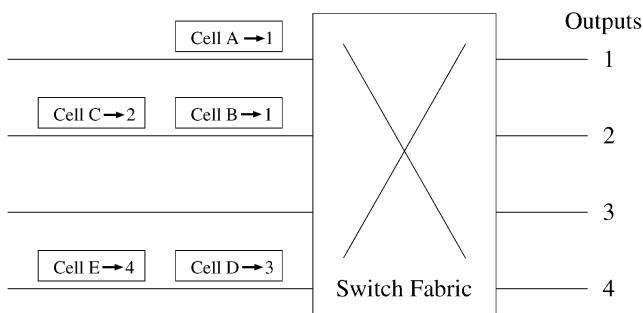


Fig. 3.2 Head-of-line blocking.

3.1.2 Traffic Models and Related Throughput Results

This subsection briefly describes how the FIFO service discipline limits the throughput under some different traffic models. Detailed analytical results are described in Section 2.3.1.

3.1.2.1 Bernoulli Arrival Process and Random Traffic Cells arrive at each input in a slot-by-slot manner. Under Bernoulli arrival process, the probability that there is a cell arriving in each time slot is identical and is independent of any other slot. This probability is referred as the offered load of the input. If each cell is equally likely to be destined for any output, the traffic becomes uniformly distributed over the switch.

Consider the FIFO service discipline at each input. Only the HOL cells contend for access to the switch outputs. If every cell is destined for a different output, the switch fabric allows each to pass through to its output. If k HOL cells are destined for the same output, one is allowed to pass through the switch, and the other $k - 1$ must wait until the next time slot. While one cell is waiting its turn for access to an output, other cells may be queued behind it and blocked from possibly reaching idle outputs. A Markov model can be established to evaluate the saturated throughput when N is small.¹ When N is large, the slot-by-slot number of HOL cells destined for a particular output becomes a Poisson process. As described in Section 2.3.1, the HOL blocking limits the maximum throughput to 0.586 when $N \rightarrow \infty$.

3.1.2.2 On-Off Model and Bursty Traffic In the bursty traffic model, each input alternates between active and idle periods of geometrically distributed duration. During an active period, cells destined for the same output arrive continuously in consecutive time slots. The probability that an active or an idle period will end at a time slot is fixed. Denote p and q as the probabilities for an active and for an idle period, respectively. The duration of an active or an idle period is geometrically distributed as expressed in the following:

$$\Pr[\text{An active period} = i \text{ slots}] = p(1 - p)^{i-1}, \quad i \geq 1,$$

$$\Pr[\text{An idle period} = j \text{ slots}] = q(1 - q)^j, \quad j \geq 0.$$

Note that it is assumed that there is at least one cell in an active period. An active period is usually called a *burst*. The mean burst length is then given by

$$b = \sum_{i=1}^{\infty} ip(1 - p)^{i-1} = \frac{1}{p},$$

¹Consider the state as one of the different destination combinations among the HOL cells.

and the offered load ρ is the portion of time that a time slot is active:

$$\rho = \frac{1/p}{1/p + \sum_{j=0}^{\infty} jq(1-q)^j} = \frac{q}{q + p - pq}.$$

Under bursty traffic, it has been shown that, when N is large, the throughput of an input-buffered switch is between 0.5 and 0.586, depending on the burstiness [19].

3.2 METHODS FOR IMPROVING PERFORMANCE

The throughput limitation of an input-buffered switch is primarily due to the bandwidth constraint and the inefficiency of scheduling cells from inputs to outputs. To improve the throughput performance, we can either develop more efficient scheduling methods or simply increase the internal capacity.

3.2.1 Increasing Internal Capacity

3.2.1.1 Multiline (Input Smoothing) Figure 3.3 illustrates an arrangement where the cells within a frame of b time slots at each of the N inputs are simultaneously launched into a switch fabric of size $Nb \times Nb$ [13, 9]. At most Nb cells enter the fabric, of which b can be simultaneously received at each output. In this architecture, the out-of-sequence problem may occur at any output buffer. Although intellectually interesting, input smoothing does not seem to have much practical value.

3.2.1.2 Speedup A speedup factor of c means that the switch fabric runs c times as fast as the input and output ports [20, 12]. A time slot is further divided into c cycles, and cells are transferred from inputs to outputs in every cycle. Each input (output) can transmit (accept) c cells in a time slot.

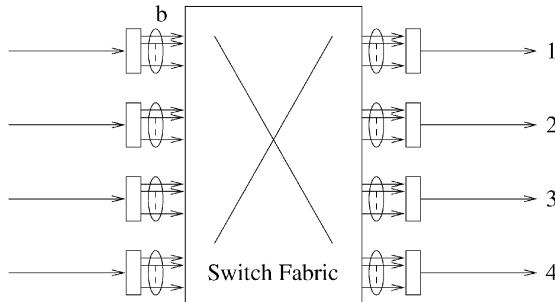


Fig. 3.3 Input smoothing.

Simulation studies show that a speedup factor of 2 yields 100% throughput [20, 12].

There is another meaning when people talk about “speedup” in the literature. At most one cell can be transferred from an input in a time slot, but during the same period of time an output can accept up to c cells [27, 6, 28]. In bursty traffic mode, a factor of 2 only achieves 82.8% to 88.5% throughput, depending on the degree of input traffic correlation (burstiness) [19].

3.2.1.3 Parallel Switch The parallel switch consists of K identical switch planes [21]. Each switch plane has its own input buffer and shares output buffers with other planes. The parallel switch with $K = 2$ achieves the maximum throughput of 1.0. This is because the maximum throughput of each switch plane is more than 0.586 for arbitrary switch size N . Since each input port distributes cells to different switch planes, the cell sequence is out of order at the output port. This type of parallel switch requires timestamps, and cell sequence regeneration at the output buffers. In addition, the hardware resources needed to implement the switch are K times as much as for a single switch plane.

3.2.2 Increasing Scheduling Efficiency

3.2.2.1 Window-Based Lookahead Selection Throughput can be increased by relaxing the strict FIFO queuing discipline at input buffers. Although each input still sends at most one cell into the switch fabric per time slot, it is not necessarily the first cell in the queue. On the other hand, no more than one cell destined for the same output is allowed to pass through the switch fabric in a time slot. At the beginning of each time slot, the first w cells in each input queue sequentially contend for access to the switch outputs. The cells at the heads of the input queues (HOL cells) contend first. Due to output conflict, some inputs may not be selected to transmit the HOL cells, and they send their second cells in line to contend for access to the remaining outputs that are not yet assigned to receive cells in this time slot. This contention process is repeated up to w times in each time slot. It allows the w cells in an input buffer’s *window* to sequentially contend for any idle outputs until the input is selected to transmit a cell. A window size of $w = 1$ corresponds to input queuing with FIFO buffers.

Table 3.1 shows the maximum throughput achievable for various switch and window sizes (N and w , respectively). The values were obtained by simulation. The throughput is significantly improved on increasing the window size from $w = 1$ (i.e., FIFO buffers) to $w = 2, 3$, and 4 . Thereafter, however, the improvement diminishes, and input queuing with even an infinite window ($w = \infty$) does not attain the optimal delay–throughput performance of output queuing. This is because input queuing limits each input to send at most one cell into the switch fabric per time slot, which prevents cells from reaching idle outputs.

TABLE 3.1 The Maximum Throughput Achievable with Input Queuing for Various Switch Sizes N and Window Sizes w

N	Window Size w							
	1	2	3	4	5	6	7	8
2	0.75	0.84	0.89	0.92	0.93	0.94	0.95	0.96
4	0.66	0.76	0.81	0.85	0.87	0.89	0.91	0.92
8	0.62	0.72	0.78	0.82	0.85	0.87	0.88	0.89
16	0.60	0.71	0.77	0.81	0.84	0.86	0.87	0.88
32	0.59	0.70	0.76	0.80	0.83	0.85	0.87	0.88
64	0.59	0.70	0.76	0.80	0.83	0.85	0.86	0.88
128	0.59	0.70	0.76	0.80	0.83	0.85	0.86	0.88

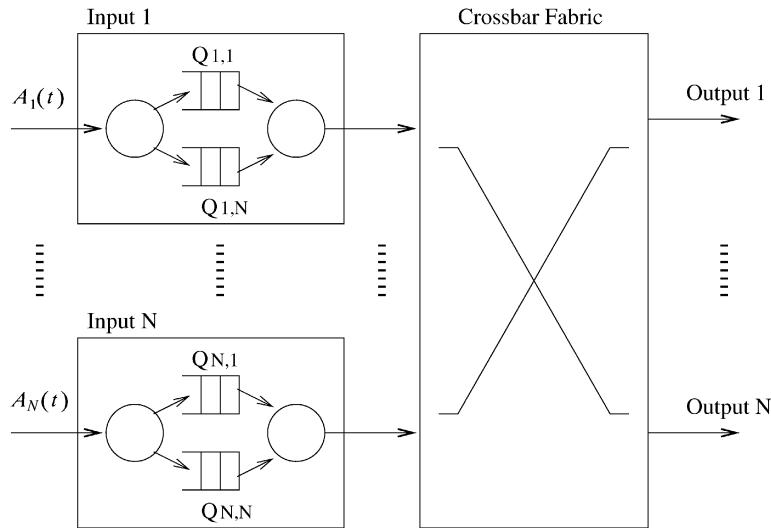


Fig. 3.4 Virtual output queue at the input ports.

3.2.2.2 VOQ-Based Matching Another way to alleviate the HOL blocking in input-buffered switches is for every input to provide a single and separate FIFO for each output. Such a FIFO is called a virtual output queue (VOQ), as shown in Figure 3.4. For example, $\text{VOQ}_{i,j}$ stores cells arriving at input port i and destined for output port j .

With virtual output queuing, an input may have cells granted access by more than one output. Since each input can transfer only one cell in a time slot, the others have to wait, and their corresponding outputs will be idle. This inefficiency can be alleviated if the algorithm runs iteratively. In more intelligent schemes, matching methods can be applied to have the optimal scheduling. Three matching methods are introduced: maximal matching, maximum matching, and stable matching.

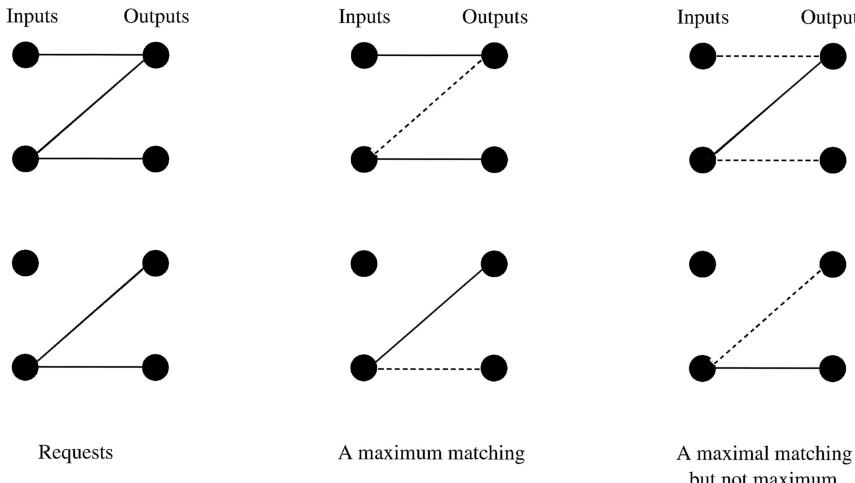


Fig. 3.5 A maximum matching and a maximal matching. Dotted lines represent the request pairs not in the matching.

A *maximum matching* is one that matches the maximum number of inputs and outputs, while a *maximal matching* is a matching where no more matches can be made without modifying the existing matches. See Figure 3.5 for an example.

The definition of stable matching assumes there is a priority list for each input and for each output. An input priority list includes all the cells queued at the input, while an output priority list concerns all the cells destined for the output (although some of them are still waiting at inputs). A matching is said to be *stable* if for each cell c waiting in an input queue, one of the following conditions holds:

1. Cell c is part of the matching, i.e., c will be transferred from the input side to the output side during this phase.
2. A cell that is ahead of c in its input priority list is part of the matching.
3. A cell that is ahead of c in its output priority list is part of the matching.

Conditions 2 and 3 may be simultaneously satisfied, but condition 1 excludes the other two. Figure 3.6 illustrates how the matching algorithm works for a 3×3 switch [32]. The letter in each cell denotes the output port where the cell should be forwarded, while the number denotes its order in the preference list of that output. The light arrows indicate the requests made by outputs, while the dark arrows represent the requests granted by inputs. During the first iterations each output asks for its most preferred cell enqueued at the inputs [see Figure 3.6(a)]. In turn, input 2 grants the only

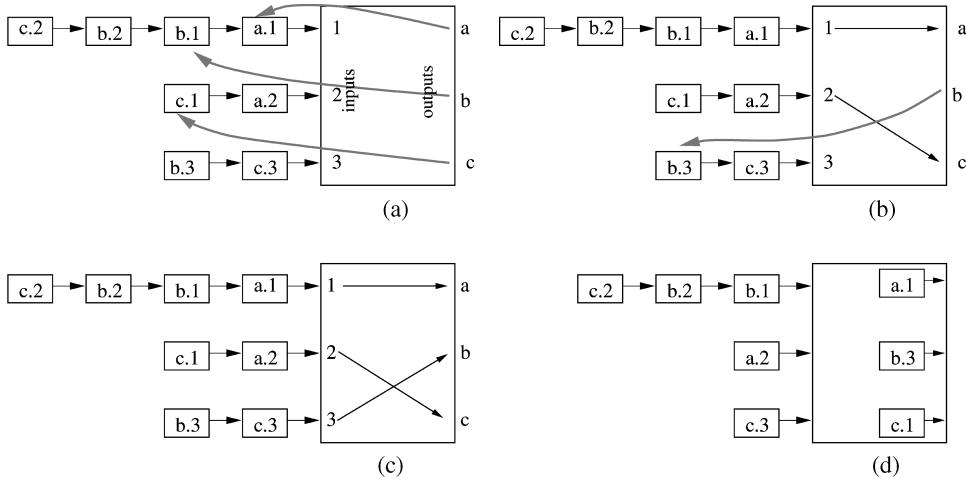


Fig. 3.6 Example to illustrate a stable matching for a 3×3 switch.

request it receives to output c , while input 1 grants the request corresponding to its most preferred cell (i.e. the request issued by output a for cell $a.1$). Thus, after the first iteration, outputs a and c are matched to their most preferred cells. During the next iteration, the unmatched output port b requests its most preferred cell from input² 3 [see Figure 3.6(b)]. As a result, input 3 grants the output b 's request and the matching completes. Figure 3.6(d) shows the switch's state after cells are transferred according to the resulting matching.

3.3 SCHEDULING ALGORITHMS

The arbitration process of a switch decides which cells at input buffers will be transferred to outputs. In the distributed manner, each output has its own arbiter, and operating independently from the others. An arbitration scheme decides which information should be passed from inputs to arbiters, and based on that information, how each output arbiter picks one among all input cells destined for the output.

An arbitration scheme is essentially a service discipline that arranges the service order among the input cells. We have three basic arbitration approaches: random selection, FIFO, and round-robin. In random selection, a cell is randomly selected among those cells. The FIFO arbitration picks the oldest cell. The slightly more complicated round-robin scheme is widely used because of its fairness. Inputs are numbered, say from 1 to N . Each output arbiter memorizes the last input that is granted access, say input i . Then

²Note that since output b was rejected in the first iteration by input 1, it does not longer consider this input in the current iteration.

input $i + 1$ has the highest priority to be granted access in the next round. If input $i + 1$ is idle in the next round, then input $i + 2$ has the highest priority, and so on so forth.

In some simple approaches, only essential information is exchanged between inputs and outputs: an input informs an output whether it has a cell for the output, and an output tells whether that cell is granted access. Some additional parameters, such as priority and timestamp, can be used to enhance the arbitration efficiency.

3.3.1 Parallel Iterative Matching (PIM)

The PIM scheme [2] uses random selection to solve the contention in inputs and outputs. Input cells are first queued in VOQs. Each iteration consists of three steps. All inputs and outputs are initially unmatched, and only those inputs and outputs that are not matched at the end of an iteration will be eligible to participate in the next matching iteration. The three steps in each iteration operate in parallel on each input and output as follows:

1. Each unmatched input sends a request to every output for which it has a queued cell.
2. If an unmatched output receives multiple requests, it grants one by randomly selecting a request over all requests. Each request has equal probability to be granted.
3. If an input receives multiple grants, it accepts one by randomly selecting an output among them.

It has been shown that, on average, 75% of the remaining grants will be matched in each iteration. Thus, the algorithm converges at $O(\log N)$ iterations. Because of the random selection, it is not necessary to store the port number granted in the previous iteration. However, implementing a random function at high speed may be too expensive.

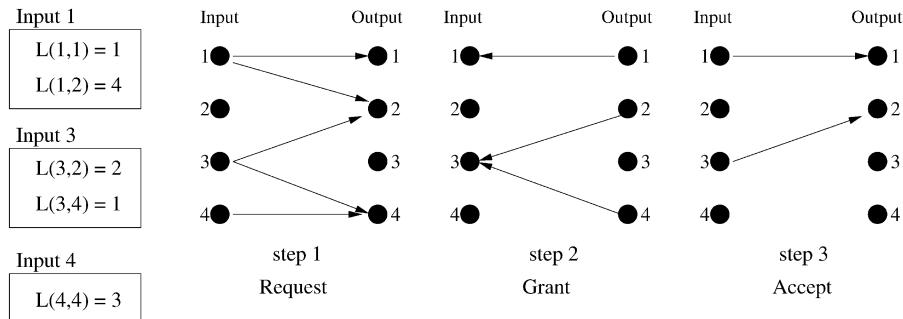
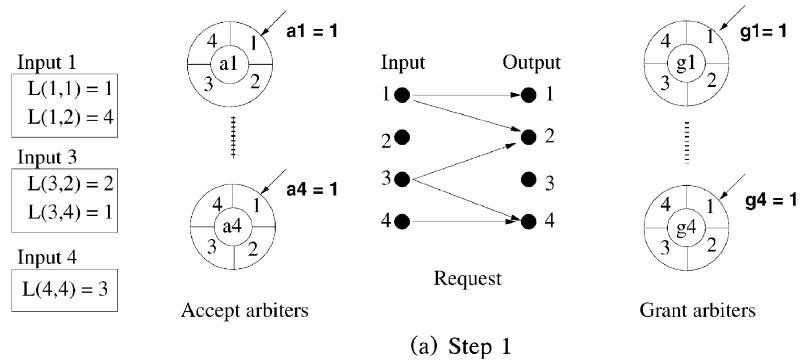
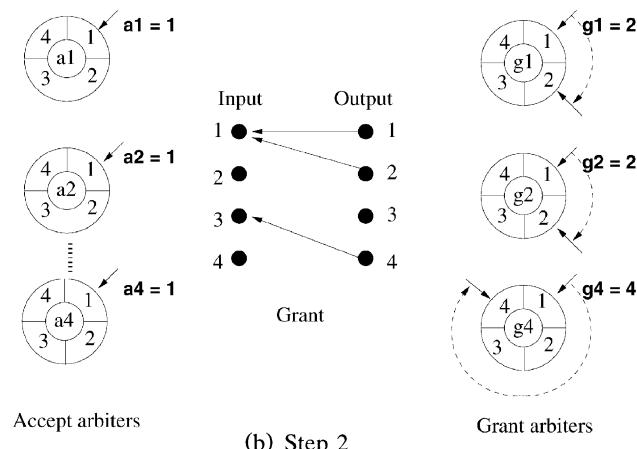


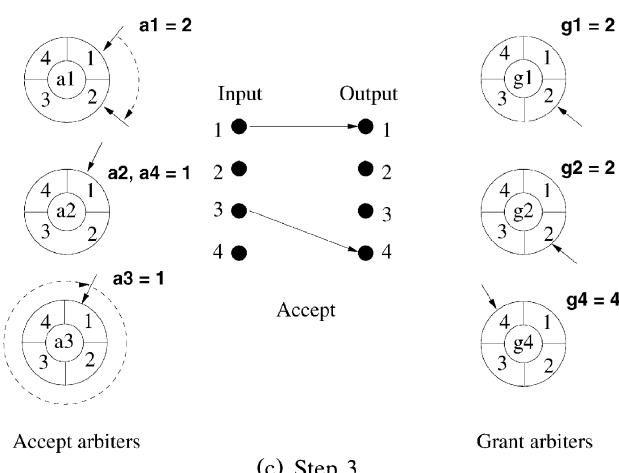
Fig. 3.7 An example of parallel iterative matching. $L(x, y) = z$ means that there are z cells on the VOQ from input x to output y .



(a) Step 1



(b) Step 2



(c) Step 3

Fig. 3.8 An example of iRRM (same input cell distribution as in PIM).

Figure 3.7 shows how PIM works. Under uniform traffic, PIM achieves 63% and 100% throughput for 1 and N iterations, respectively.

3.3.2 Iterative Round-Robin Matching (*iRRM*)

The *iRRM* scheme [23] works similarly to PIM, but uses the round-robin schedulers instead of random selection at both inputs and outputs. Each arbiter maintains a pointer pointing at the port that has the highest priority. Such a pointer is called the *accept pointer* a_i at input i or the *grant pointer* g_j at output j . The algorithm is run as follows:

1. Each unmatched input³ sends a request to every output for which it has a queued cell.
2. If an unmatched output receives any requests, it chooses the one that appears next in a round-robin schedule, starting from the highest-priority element. The output notifies each input whether or not its request was granted. The pointer g_i is incremented (modulo N) to one location beyond the granted input.
3. If an input receives a multiple grant, it accepts the one that appears next in its round-robin schedule, starting from the highest-priority element. Similarly, the pointer a_j is incremented (modulo N) to one location beyond the accepted output.

An example is shown in Figure 3.8. In this example, we assume that the initial value of each grant pointer is input 1 (e.g., $g_i = 1$). Similarly, each accept pointer is initially pointing to output 1 (e.g., $a_j = 1$). During step 1, the inputs request transmission to all outputs that they have a cell destined for. In step 2, among all received requests, each grant arbiter selects the requesting input that is nearest to the one currently pointed to. Output 1 chooses input 1, output 2 chooses input 1, output 3 has no requests, and output 4 chooses input 3. Then, each grant pointer moves one position beyond the selected one. In this case, $g_1 = 2$, $g_2 = 2$, $g_3 = 1$, and $g_4 = 4$. In step 3, each accept pointer decides which grant is accepted, as the grant pointers did. In this example, input 1 accepts output 1, and input 3 accepts output 4, then $a_1 = 2$, $a_2 = 1$, $a_3 = 1$, and $a_4 = 1$. Notice that the pointer a_3 accepted the grant issued by output 4, so the pointer returns to position 1.

3.3.3 Iterative Round-Robin with SLIP(*iSLIP*)

An enhanced scheme (*iSLIP*) was presented in [24, 25]. The difference is that in this scheme, the grant pointers update their positions only if their grants are accepted. In this scheme, starvation is avoided because a recently matched pair gets the lowest priority. The steps for this scheme are as

³At the beginning each input is unmatched.

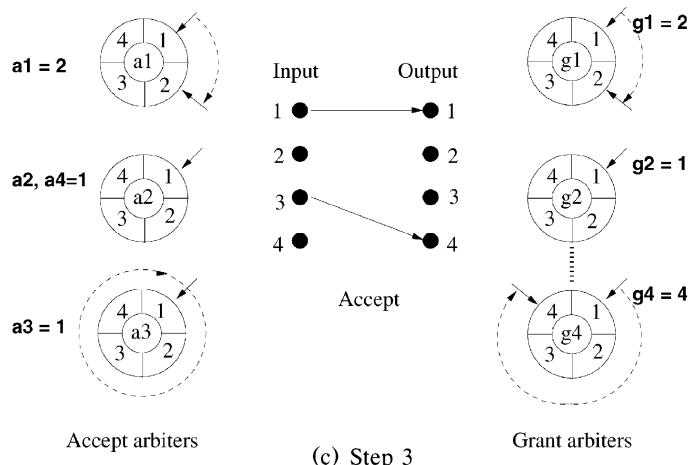
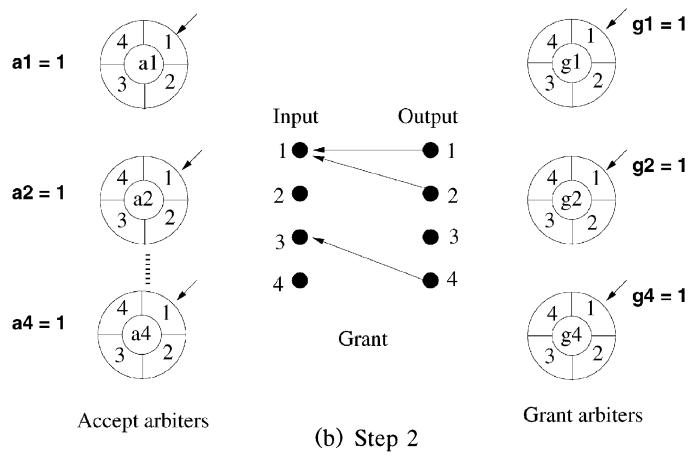
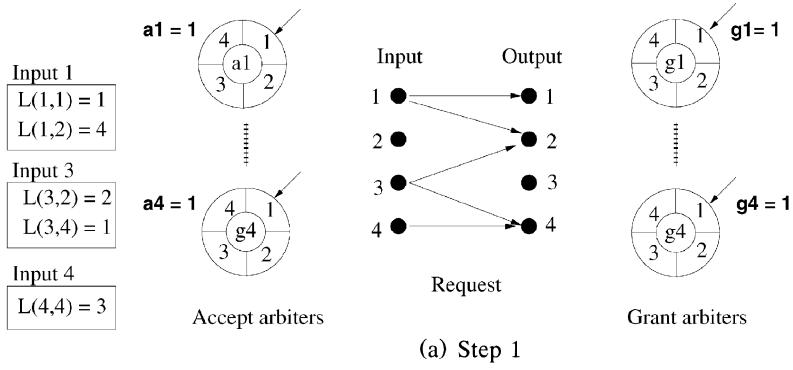


Fig. 3.9 iSLIP example.

follows:

1. Each unmatched input sends a request to every output for which it has a queued cell.
2. If an unmatched output receives multiple requests, it chooses the one that appears next in a fixed, round-robin schedule starting from the highest-priority element. The output notifies each input whether or not its request was granted.
3. If an input receives multiple grants, it accepts the one that appears next in a fixed round-robin schedule starting from the highest-priority element. The pointer a_j is incremented (modulo N) to one location beyond the accepted output. *The accept pointers a_i are updated only in the first iteration.*
4. The grant pointer g_i is incremented (modulo N) to one location beyond the granted input if and only if the grant is accepted in step 3 of the first iteration. Like the accept pointers, *the pointers g_i are updated only in the first iteration.*

Because of the round-robin moving of the pointers, we expect the algorithm to provide a fair allocation of bandwidth among all flows. This scheme contains $2N$ arbiters, each of which is implementable with low complexity. The throughput offered with this algorithm is 100% for any number of iterations, due to the desynchronization effect (see Section 3.3.4). A matching example of this scheme is shown in Figure 3.9. Considering the example from the *iRRM* discussion, initially all pointers a_j and g_i are set to 1. In step 2 of *iSLIP*, the output accepts the request that is closer to the pointed input in a clockwise direction; however, in a manner different from *iRRM*, the pointers g_i are not updated in this step. They wait for the acceptance result. In step 3, the inputs accept the grant that is closer to the one pointed to by a_i . The accept pointers change to one position beyond the accepted one, $a_1 = 2$, $a_2 = 1$, $a_3 = 1$, and $a_4 = 1$. Then, after the accept pointers decide which grant is accepted, the grant pointers change to one position beyond the accepted grant (i.e., a nonaccepted grant produces no change in a grant pointer position). The new values for these pointers are $g_1 = 2$, $g_2 = 1$, $g_3 = 4$ and $g_4 = 1$. In the following iterations, only the unmatched input and outputs are considered and the pointers are not modified (i.e., updating occurs in the first iteration only).

3.3.4 Dual Round-Robin Matching (DRRM)

The DRRM scheme [3, 4] works similarly to *iSLIP*, also using the round-robin selection instead of random selection. But it starts the round-robin selection at inputs. An input arbiter is used to select a nonempty VOQ according to the round-robin service discipline. After the selection, each input sends a request, if necessary, to the destined output arbiter. An output arbiter receives up to N requests. It chooses one of them based on the round-robin

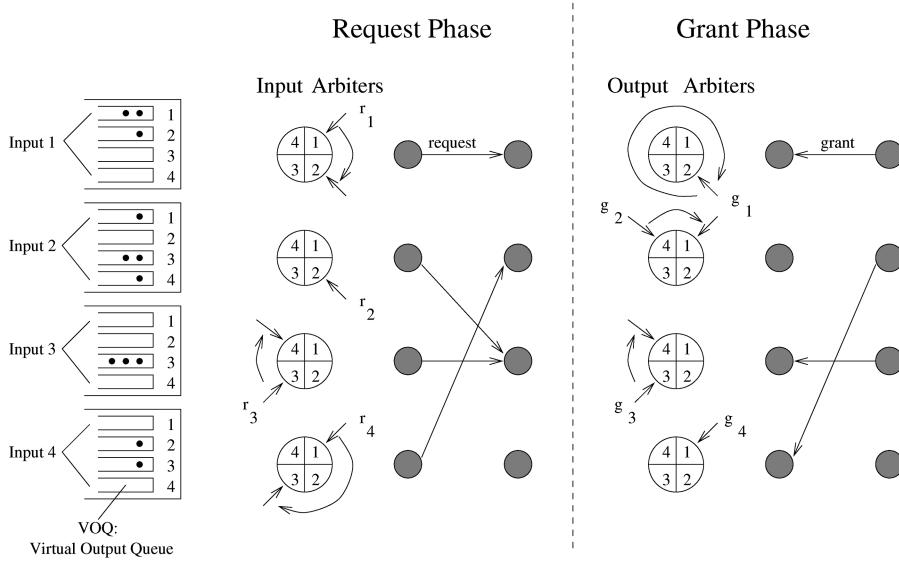
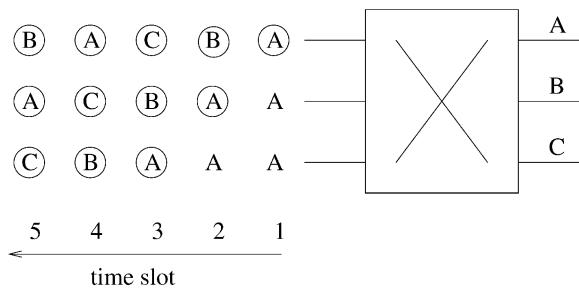


Fig. 3.10 An example of the dual round-robin scheduling algorithm. (©2000 IEEE.)

service discipline, and sends a grant to the winner input port. Because of the two sets of independent round-robin arbiters, this arbitration scheme is called dual round-robin (DRR) arbitration.

The DRR arbitration has four steps in a cycle: (1) each input arbiter performs request selection, and (2) sends a request to the output arbiters; (3) each output arbiter performs grant arbitration, and (4) the output arbiters send grant signals to input arbiters. Figure 3.10 shows an example of the DRR arbitration algorithm. In a request phase, each input chooses a VOQ and sends a request to an output arbiter. Assume input 1 has cells destined for both outputs 1 and 2. Since its round-robin pointer, r_1 , is pointing to 1, input arbiter 1 sends a request to output 1 and updates its pointer to 2. Let us consider output 3 in the grant phase. Since its round-robin pointer, g_3 , is pointing to 3, output arbiter 3 grants access to input 3 and updates its pointer to 4. Like iSLIP, DRRM has the desynchronization effect. The input arbiters granted in different time slots have different pointer values, and each of them requests a different output, resulting in desynchronization. However, the DRR scheme requires less time to do arbitration and is easier to implement. This is because less information exchange is needed between input arbiters and output arbiters. In other words, DRRM saves the initial transmission time required to send requests from inputs to outputs in iSLIP.

Consider the fully loaded situation in which every VOQ always has cells. Figure 3.11 shows the HOL cells chosen from each input port in different time slots. In time slot 1, each input port chooses a cell destined for output A. Among those three cells, only one (the first one in this example) is granted and the other two have to wait at HOL. The round-robin (RR) pointer of the



(B) Indicate that the cell is granted in the time slot.

Fig. 3.11 The desynchronization effect of DRRM under the fully loaded situation. Only HOL cells at each input are shown for illustration. (©2000 IEEE.)

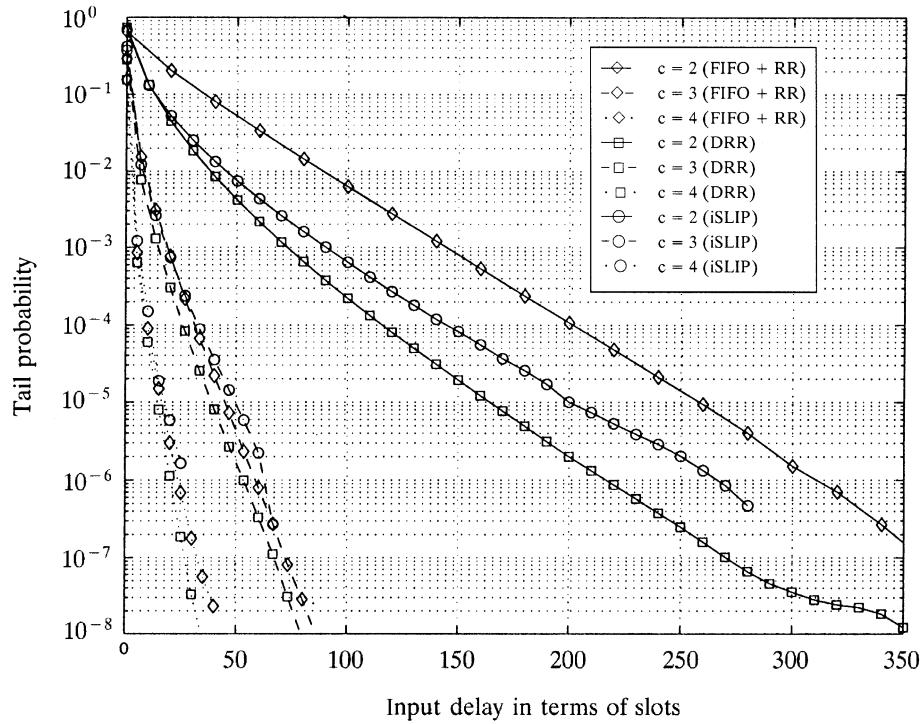


Fig. 3.12 Comparison of tail probability of input delay under three arbitration schemes.

first input advances to point to output B in time slot 2, and a cell destined for B is chosen and then granted because of no contenders. The other two inputs have their HOL cells unchanged, both destined for output A . Only one of them (the one from the second input) is granted, and the other has to wait until the third time slot. At that time, the round-robin pointers among the three inputs have been desynchronized and point to C , B , and A , respectively. As a result, all three cells chosen are granted.

Figure 3.12 shows the tail probability under FIFO + RR (FIFO for input selection and RR for round-robin arbitration), DRR, and *iSLIP* arbitration schemes. The switch size is 256, and the average burst length is 10 cell slots (with the on-off model). DRR's and *iSLIP*'s performance are comparable at a speedup of 2, while all three schemes have almost the same performance at speedups $c \geq 3$.

3.3.5 Round-Robin Greedy Scheduling

Although *iSLIP* and DRRM are efficient scheduling algorithms to achieve high switch throughput, they have a timing constraint that the scheduling has to be completed within one cell time slot. That constraint can become a bottleneck when the switch size or a port speed increases. For instance, when considering a 64-byte fixed-length cell at a port speed of 40 Gbit/s (OC-786), the computation time available for maximal-sized matching is only 12.8 ns.

To relax the scheduling timing constraint, a pipeline-based scheduling algorithm called *round-robin greedy scheduling* (RRGS) was proposed by Smiljanic et al. [30].

Before we describe pipelined RRGS, nonpipelined RRGS is described. Consider an $N \times N$ crossbar switch, where each input port i , $i \in \{0, 1, \dots, N - 1\}$, has N logical queues, corresponding to each of the N outputs. All packets are fixed-size cells. The input of the RRGS protocol is the state of all input/output queues, or a set $C = \{(i, j) \mid \text{there is at least one packet at input } i \text{ for output } j\}$. The output of the protocol is a schedule, or a set $S = \{(i, j) \mid \text{packet will be sent from input } i \text{ to output } j\}$. Note that in each time slot, an input can only transmit one packet, and an output can receive only one packet. The schedule for the k th time slot is determined as follows:

- *Step 1:* $I_k = \{0, 1, \dots, N - 1\}$ is the set of all inputs; $O_k = \{0, 1, \dots, N - 1\}$ is the set of all outputs. $i = (\text{const} - k) \bmod N$ (such choice of an input that starts a schedule will enable a simple implementation).
- *Step 2:* If I_k is empty, stop; otherwise, choose the next input in a round-robin fashion according to $i = (i + 1) \bmod N$.
- *Step 3:* Choose in a round-robin fashion the output j from O_k such that $(i, j) \in C_k$. If there is none, remove i from I_k and go to step 2.
- *Step 4:* Remove input i from I_k , and output j from O_k . Add (i, j) to S_k . Go to step 2.

The scheduling of a given time slot in RRGS consists of N phases. In each phase, one input chooses one of the remaining outputs for transmission during that time slot. A phase consists of the request from the input module (IM) to the RR arbiter, RR selection, and acknowledgement from the RR arbiter to the IM. The RR order in which inputs choose outputs shifts cyclically for each time slot, so that it ensures equal access for all inputs.

Now, we describe pipelined RRGS by applying the nonpipelined concept. We assume that N is an odd number to simplify the discussion. When N is an even number, the basic concept is the same, but there are minor changes. N separate schedules are in progress simultaneously, for N distinct time slots in the future. Each phase of a particular schedule involves just one input. In any given time slot, other inputs may simultaneously perform the phases of schedules for other distinct time slots in the future. While N time slots are required to complete the N phases of a given schedule, N phases of N different schedules may be completed within one time slot using a pipeline approach, and computing the N schedules in parallel. But this is effectively equivalent to the completion of one schedule every time slot. In RRGS, a specific time slot in the future is made available to the inputs for scheduling in a round-robin fashion. Input i , which starts a schedule for the k th time slot (in the future), chooses an output in a RR fashion, and sends to the next input, $(i + 1) \bmod N$, a set O_k that indicates the remaining output ports that are still free to receive packets during the k th time slot. Any input i , on receiving from the previous input, $(i - 1) \bmod N$, the set O_k of available outputs for the k th time slot, chooses one output if possible from this set, and sends to the next input, $(i + 1) \bmod N$, the modified set O_k , if input i did not complete the schedule for the k th time slot, T_k . An input i that completes a schedule for the k th time slot should not forward the modified set O_k to the next input, $(i + 1) \bmod N$. Thus input $(i + 1) \bmod N$, which did not receive the set O_k in the current time slot, will be starting a new schedule (for a new time slot) in the next time slot. Step 1 of RRGS implies that an input refrains from forwarding O_k once in N time slots. The input i that does not forward O_k should be the last one that chooses an output for the k th time slot. Thus, since each RR arbiter has only to select one candidate within one time slot, RRGS dramatically relaxes the scheduling time constraint compared with iSLIP and DRRM.

Figure 3.13 shows an example of a timing diagram of a pipelined 5×5 RRGS algorithm.

A simple structure for the central controller that executes the RRGS algorithm is shown in Figure 3.14. A RR arbiter associated to each input module communicates only with the RR arbiters associated to adjacent input modules, and the complex interconnection between input and output modules is avoided. It stores addresses of the reserved outputs into the memory (M).

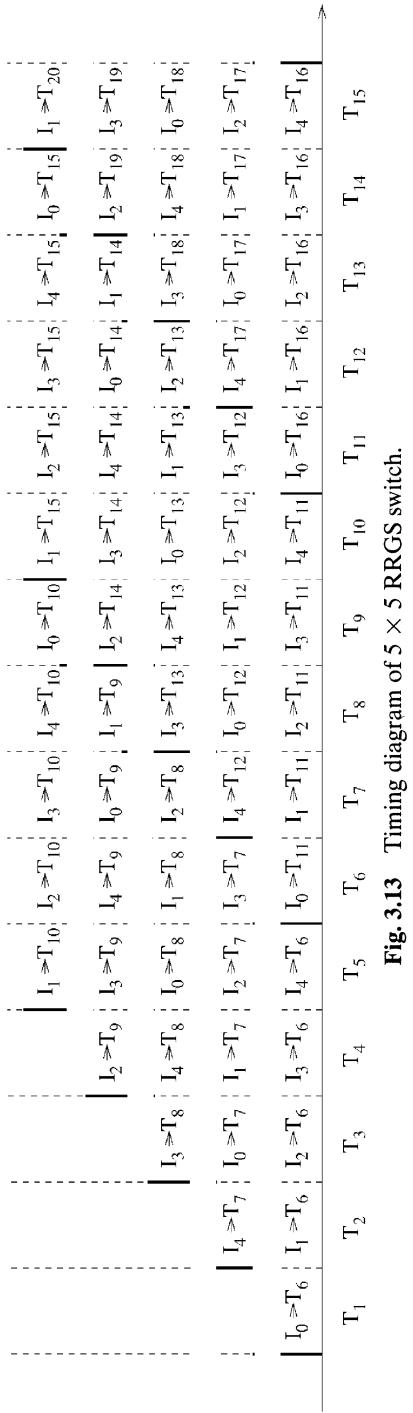


Fig. 3.13 Timing diagram of 5×5 RRGS switch.

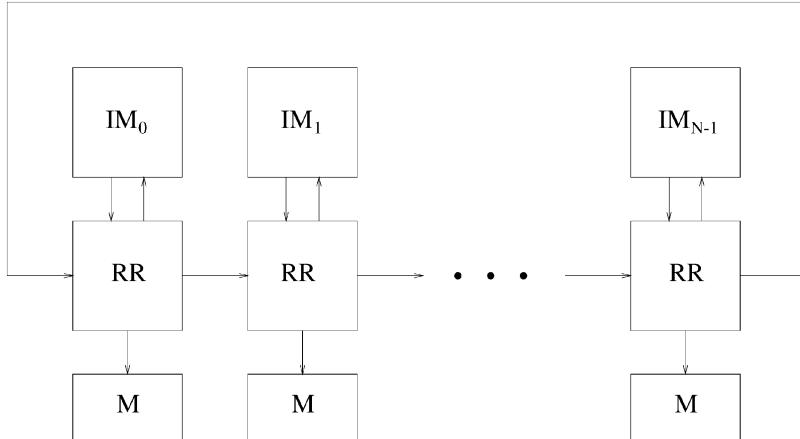


Fig. 3.14 Central controller for RRGS protocol.

The average delay of RRGS, D_{RRGS} , is approximately given as [30]

$$D_{RRGS} = \frac{1 - p/N}{1 - p} + \frac{N}{2}. \quad (3.1)$$

The price that RRGS pays for its simplicity is the additional pipeline delay, which is on average equal to $N/2$ time slots. This pipeline delay is not critical for the assumed very short packet transmission time. Smiljanić et al. [30] showed that RRGS provides better performance than *i*SLIP with one iteration for a heavy load.

Smiljanić also proposed so-called *weighted RRGS* (WRRGS), which guarantees prereserved bandwidth [31], while preserving the advantage of RRGS. WRRGS can flexibly share the bandwidth of any output among the inputs.

3.3.6 Design of Round-Robin Arbiters / Selectors

A challenge to building a large-scale switch lies in the stringent arbitration time constraint for output contention resolution. This section describes how to design the RR-based arbiters.

3.3.6.1 Bidirectional Arbiter A bidirectional arbiter for output resolution control operates using three output-control signals, DL , DH , and UH as shown in Figure 3.15 [10]. Each signal is transmitted on one bus line. DL and DH are downstream signals, while UH is the upstream signal.

The three signal bus functions are summarized as follows:

- DL : Selects highest requested crosspoint in group L .
- DH : Selects highest requested crosspoint in group H .
- UH : Identifies whether group H has request or not.

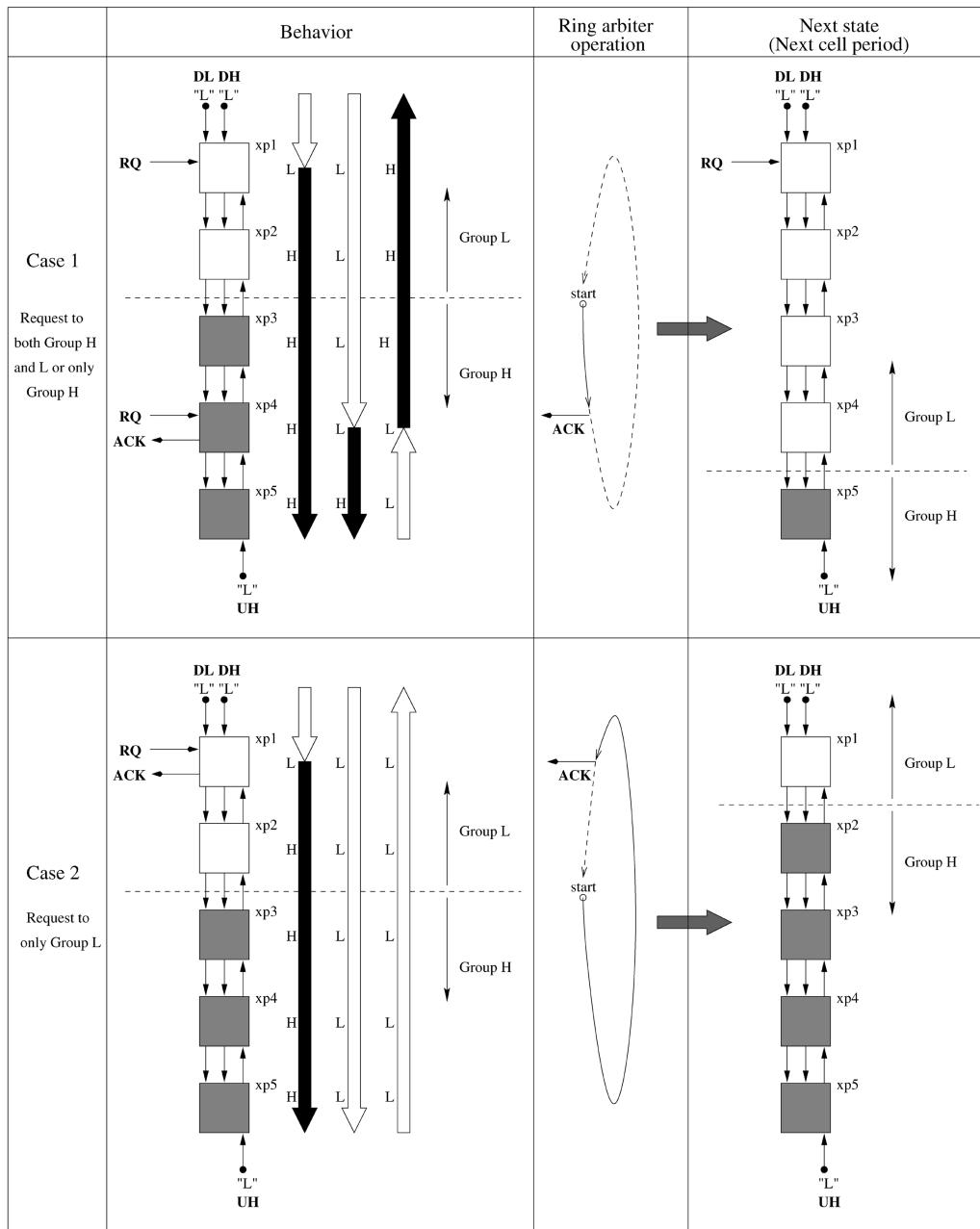


Fig. 3.15 Block diagram of the bidirectional arbiter. (©1994 IEEE.)

TABLE 3.2 A Part of the Logic Table of the Bidirectional Arbiter

Level Combination				next	
DL	DH	UH	GI ^a	ACK	GI ^a
H	H	H	H	L	L
H	H	H	L	L	L
H	H	L	H	L	H
H	H	L	L	L	H
L	H	H	H	L	H
L	H	H	L	L	H
L	H	L	H	L	H
L	H	L	L	H	L
H	L	H	H	H	L
H	L	H	L	L	L
H	L	L	H	H	L
H	L	L	L	L	H
L	L	H	H	H	L
L	L	H	L	L	L
L	L	L	H	H	L
L	L	L	L	L	xp1 (case 1 in Fig. 3.15)
H	L	L	L	L	H
L	L	H	L	L	xp1 (case 1 in Fig. 3.15)
L	L	L	H	H	L
L	L	L	L	H	xp1 (case 2 in Fig. 3.15)

^aGroup Indication—indicating which group the crosspoint belongs to.

The arbitration procedure can be analyzed into two cases in Figure 3.15. Case 1 has group *H* active or both *L* and *H* active. In this case, the highest request in group *H* will be selected. Case 2 has only group *L* active. In that case, the highest request in group *L* will be selected.

In case 1, the *DH* signal locates the highest crosspoint in Group *H*, xp4, and triggers the ACK signal. The *UH* signal indicates that there is at least one request in group *H* to group *L*. The *DL* signal finds the highest crosspoint, xp1, in group *L*, but no ACK signal is sent from xp1, because of the state of the *UH* signal. In the next cell period, xp1 to xp4 form group *L*, and xp5 is group *H*.

In case 2, the *UH* signal shows that there is no request in group *H*; the *DL* signal finds the highest request in group *L*, xp1, and triggers the ACK signal.

Note that both selected crosspoints in case 1 and case 2 are the same crosspoints that would be selected by a theoretical RR arbiter.

The bidirectional arbiter operates two times faster than a normal ring arbiter (unidirectional arbiter), but twice as many transmitted signals are required. The bidirectional arbiter can be implemented with simple hardware. Only 200 or so gates are required to achieve the distributed arbitration at each crosspoint, according to the logic table shown in Table 3.2.

3.3.6.2 Token Tunneling This section introduces a more efficient arbitration mechanism called *token tunneling* [4]. Input requests are arranged into groups. A token starts at the RR pointer position and runs through all

XPU - crosspoint unit

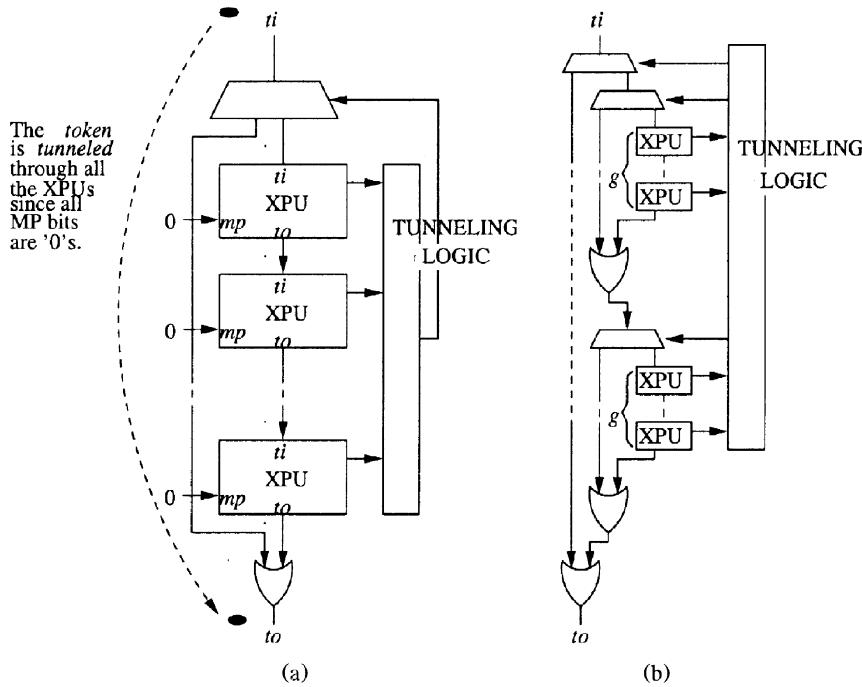


Fig. 3.16 The token tunneling scheme. (©2000 IEEE.)

requests before selecting the one with the highest priority. As shown in Figure 3.16(a), when the token arrives in a group where all requests are 0, it can skip this group taking the *tunnel* directly from the input of the group to the output. The arbitration time thus becomes proportional to the number of groups, instead of the number of ports.

Suppose each group handles n requests. Each crosspoint unit (XPU) contributes a two-gate delay for arbitration. The token rippling through all the XPU's in the group where the token is generated and all the XPU's in the group where the token is terminated contributes a $4n$ -gate delay. There are altogether N/n groups, and at most $N/n - 2$ groups will be tunneled through. This contributes a $2(N/n - 2)$ -gate delay. Therefore, the worst case time complexity of the basic token tunneling method is $D_t = 4n + 2(N/n - 2)$ gates of delay, with 2 gates of delay contributed by the tunneling in each group. This occurs when there is only one active request and it is at the farthest position from the round-robin pointer—or example, the request is at the bottommost XPU while the round-robin pointer points to the topmost one.

By tunneling through smaller XPU groups of size g and having a *hierarchy* of these groups as shown in Figure 3.16(b), it is possible to further reduce the

worst case arbitration delay to $D_t = 4g + 5d + 2(N/n - 2)$ gates, where $d = \lceil \log_2(n/g) \rceil$. The hierarchical method basically decreases the time spent in the group where the token is generated and in the group where the token is terminated. For $N = 256$, $n = 16$, and $g = 2$, the basic token tunneling method requires a 92-gate delay, whereas the hierarchical method requires only a 51-gate delay.

3.4 OUTPUT-QUEUING EMULATION

The major drawback of input queuing is that the queuing delay between inputs and outputs is variable, which makes delay control more difficult. Can an input–output-buffered switch with a certain speedup behave identically⁴ to an output-queued switch? The answer is yes, with the help of a better scheduling scheme. This section first introduces some basic concepts and then highlights some scheduling schemes that achieve the goal.

3.4.1 Most-Urgent-Cell-First Algorithm (MUCFA)

The MUCFA scheme [29] schedules cells according to their *urgency*. A shadow switch with output queuing is considered in the definition of the urgency of a cell. The urgency, which is also called the time to leave (TL), is the time after the present that the cell will depart from the output-queued (OQ) switch. This value is calculated when a cell arrives. Since the buffers of the OQ switch are FIFO, the urgency of a cell at any time equals the number of cells ahead of it in the output buffer at that time. It gets decremented after every time slot. Each output has a record of the urgency value of every cell destined for it. The algorithm is run as follows:

1. At the beginning of each phase, each output sends a request for the most urgent cell (i.e., the one with the smallest TL) to the corresponding input.
2. If an input receives more than one request, then it will grant to that output whose cell has the smallest urgency number. If there is a tie between two or more outputs, a supporting scheme is used. For example, the output with the smallest port number wins, or the winner is selected in a RR fashion.
3. Outputs that lose contention will send a request for their next most urgent cell.
4. The above steps run iteratively until no more matching is possible. Then cells are transferred, and MUCFA goes to the next phase.

An example is shown in Figure 3.17. Each number represents a queued cell, and the number itself indicates the urgency of the cell. Each input maintains three VOQs, one for each output. Part (a) shows the initial state of

⁴Under identical inputs, the departure time of every cell from both switches is identical.

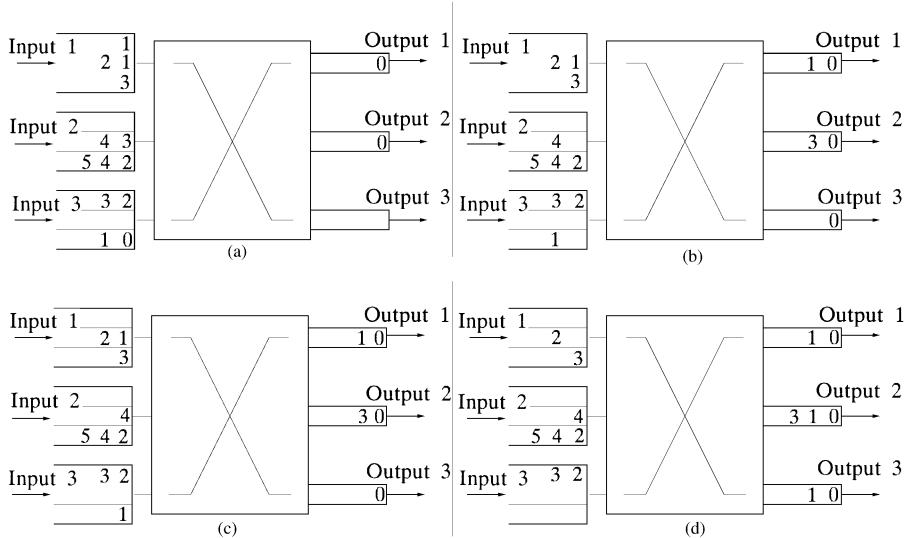


Fig. 3.17 An example of two phases of MUCFA.

the first matching phase. Output 1 sends a request to input 1, since the HOL cell in $\text{VOQ}_{1,1}$ is the most urgent for it. Output 2 sends a request to input 1, since the HOL cell in $\text{VOQ}_{1,2}$ is the most urgent for it. Output 3 sends a request to input 3, since the HOL cell in $\text{VOQ}_{3,3}$ is the most urgent for it. Part (b) illustrates matching results of the first phase, where cells from $\text{VOQ}_{1,1}$, $\text{VOQ}_{2,2}$, and $\text{VOQ}_{3,3}$ are transferred. Part (c) shows the initial state of the second phase, while part (d) gives the matching results of the second phase, in which HOL cells from $\text{VOQ}_{1,2}$ and $\text{VOQ}_{3,3}$ are matched.

It has been shown that, under an internal speedup of 4, a switch with VOQ and MUCFA scheduling can behave identically to an OQ switch, regardless of the nature of the arrival traffic.

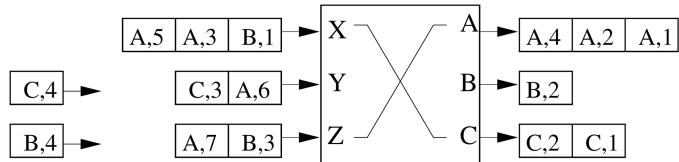
3.4.2 Chuang et al.'s Results

This category of algorithms is based on an implementation of priority lists for each arbiter to select a matching pair [7]. The input priority list is formed by positioning each arriving cell at a particular place in the input queue. The relative ordering among other queued cells remains unchanged. This kind of queue is called a *push-in queue*. Some metrics are used for each arriving cell to determine the location. Furthermore, if cells are removed from the queue in an arbitrary order, we call it a *push-in arbitrary out* (PIAO) queue. If the cell at the head of queue is always removed next, we call it a *push-in first out* (PIFO) queue.

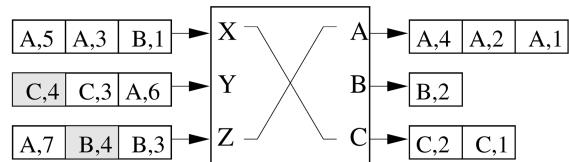
The algorithms described in this section also assume a shadow OQ switch, based on which the following terms are defined:

1. The *time to leave* $\text{TL}(c)$ is the time slot in which cell c would leave the shadow OQ switch. Of course, $\text{TL}(c)$ is also the time slot in which cell c must leave the real switch for the identical behavior to be achieved.
2. The *output cushion* $\text{OC}(c)$ is the number of cells waiting in the output buffer at cell c 's output port that have a lower TL value than cell c . If a cell has a small (or zero) output cushion, then it is urgent for it to be delivered to its output so that it can depart when its time to leave is reached. If it has a large output cushion, it may be temporarily set aside while more urgent cells are delivered to their outputs. Since the switch is work-conserving, a cell's output cushion is decremented after every time slot. A cell's output cushion increases only when a newly arriving cell is destined for the same output and has a more urgent TL.
3. The *input thread* $\text{IT}(c)$ is the number of cells ahead of cell c in its input priority list. $\text{IT}(c)$ represents the number of cells currently at the input that have to be transferred to their outputs more urgently than cell c . A cell's input thread is decremented only when a cell ahead of it is transferred from the input, and is possibly incremented when a new cell arrives. It would be undesirable for a cell to simultaneously have a large input thread and a small output cushion—the cells ahead of it at the input might prevent it from reaching its output before its TL. This motivates the definition of *slackness*.
4. The *slackness* $L(c)$ equals the output cushion of cell c minus its input thread, that is, $L(c) = \text{OC}(c) - \text{IT}(c)$. Slackness is a measure of how large a cell's output cushion is with respect to its input thread. If a cell's slackness is small, then it is urgent for it to be transferred to its output. If a cell has a large slackness, then it may be kept at the input for a while.

3.4.2.1 Critical Cell First (CCF) CCF is a scheme of inserting cells in input queues that are PIFO queues. An arriving cell is inserted as far from the head of its input queue as possible so that the input thread of the cell is not larger than its output cushion (i.e., the slackness is positive). Suppose that cell c arrives at input port P . Let x be the number of cells waiting in the output buffer at cell c 's output port. Those cells have a lower TL value than cell c or the output cushion $\text{OC}(c)$ of c . Insert cell c into $(x + 1)$ th position from the front of the input queue at P . As shown in Figure 3.18, each cell is represented by its destined output port and the TL. For example, cell $(B, 4)$ is destined for output B and has a TL value equal to 4. Part (a) shows the initial state of the input queues. Part (b) shows the insertion of two incoming cells $(C, 4)$ and $(B, 4)$ to ports Y and Z , respectively. Cell $(C, 4)$ is inserted at



(a)



(b)

Fig. 3.18 Example of CCF priority placement.

the third place of port Y , and cell $(B, 4)$ at the second place of port Z . Hence, upon arrival, both cells have zero slackness. If the size of the priority list is less than x cells, then place c at the end of the input priority list. In this case, cell c has a positive slackness. Therefore, every cell has a nonnegative slackness on arrival.

3.4.2.2 Last In, Highest Priority (LIHP) LIHP is also a scheme of inserting cells at input queues. It was proposed mainly to show and demonstrate the sufficient speedup to make an input–output queued switch emulate an output-queued switch. LIHP places a newly arriving cell right at the front of the input priority list, providing a zero input thread [$IT(c) = 0$] for the arriving cell. See Figure 3.19 for an example. The scheduling in every arbitration phase is a stable matching based on the TL value and the position in its input priority list of each queued cell.

The necessary and sufficient speedup is $2 - 1/N$ for an $N \times N$ input-output-queued switch to exactly emulate a $N \times N$ output-queued switch with FIFO service discipline.

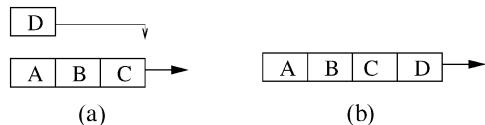


Fig. 3.19 An example of placement for LIHP. (a) The initial state. (b) The new cell is placed at the highest-priority position.

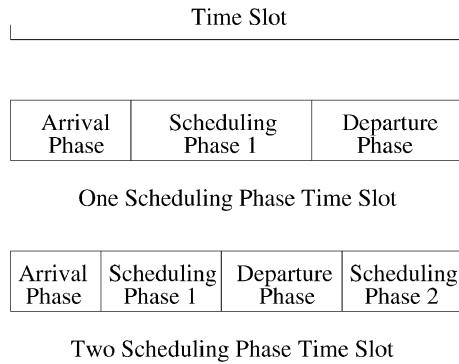


Fig. 3.20 One-scheduling-phase and two-scheduling-phase time slots.

The necessary condition can be shown by the example as shown below. Since the speedup $2 - 1/N$ represents a nonintegral distribution of arbitration phases per slot between one and two, we first describe how scheduling phases are distributed. A speedup of $2 - 1/N$ corresponds to having one *truncated* time slot out of every N time slots; the truncated time slot has just one scheduling phase, whereas the other $N - 1$ time slots have two scheduling phases each. Figure 3.20 shows the difference between one-phase and two-phase time slots. We assume that the scheduling algorithm does not know in advance whether a time slot is truncated.

Recall that a cell is represented as a tuple (P, TL) , where P represents which output port the cell is destined to and TL represents the time to leave for the cell. For example, the cell $(C, 7)$ must be scheduled for port C before the end of time slot 7.

The input traffic pattern that provides the lower bound for an $N \times N$ input-output-queued switch is given as follows. The traffic pattern spans N time slots, the last of which is truncated:

1. In the first time slot, all input ports receive cells destined for the same output port, P_1 .
2. In the second time slot, the input port that had the lowest TL in the previous time slot does not receive any more cells. In addition, the rest of the input ports receive cells destined for the same output port, P_2 .
3. In the i th time slot, the input ports that had the lowest TL in each of the $i - 1$ previous time slots do not receive any more cells. In addition, the rest of the input ports must receive cells destined for the same output port, P_i .

One can repeat the traffic pattern just mentioned as many times as is required to create arbitrarily long traffic patterns. Figure 3.21 shows the

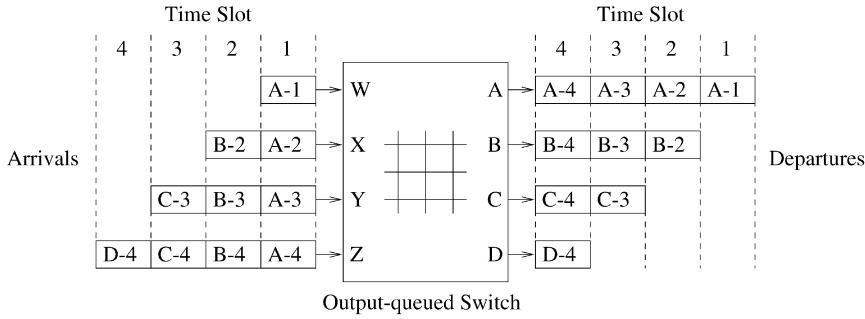


Fig. 3.21 Lower-bound input traffic pattern for a 4×4 switch.

above sequence of cells for a 4×4 switch. The departure events from the output-queued switch are depicted on the right, and the arrival events are on the left. For simplicity, we present the proof of our lower bound on this 4×4 switch instead of a general $N \times N$ switch.

Figure 3.22 shows the only possible schedule for transferring these cells across in seven phases. Of the four time slots, the last one is truncated, giving a total of seven phases. Cell A-1 must leave the input side during the first phase, since the input-output-queued switch does not know whether the first time slot is truncated. Similarly, cells B-2, C-3, and D-4 must leave during the third, fifth, and seventh phases, respectively [see Fig. 3.22(a)]. Cell A-2 must leave the input side by the end of the third phase. However, it cannot leave during the first or the third phase, because of contention. Therefore, it must

Phase	PA	PB	PC	PD
1	A-1			
2				
3		B-2		
4				
5			C-3	
6				
7				D-4

(a)

Phase	PA	PB	PC	PD
1	A-1			
2	A-2			
3		B-2		
4		B-3		
5			C-3	
6			C-4	
7				D-4

(b)

Phase	PA	PB	PC	PD
1	A-1			
2	A-2			
3	A-3	B-2		
4		B-3		
5		B-4	C-3	
6			C-4	
7				D-4

(c)

Phase	PA	PB	PC	PD
1	A-1			
2	A-2			
3	A-3	B-2		
4	A-4	B-3		
5		B-4	C-3	
6			C-4	
7				D-4

(d)

Fig. 3.22 Scheduling order for the lower-bound input traffic pattern in Figure 3.21.

depart during the second phase. Similarly, cells B-3 and C-4 must depart during the fourth and sixth phases, respectively [see Fig. 3.22(b)]. Continuing this elimination process [see Fig. 3.22(c) and (d)], there is only one possible scheduling order. For this input traffic pattern, the switch needs all seven phases in four time slots, which corresponds to a minimum speedup of $\frac{7}{4}$ (or $2 - \frac{1}{4}$). The proof of the general case for an $N \times N$ switch is a straightforward extension of the 4×4 example.

3.5 LOWEST-OUTPUT-OCCUPANCY-CELL-FIRST ALGORITHM (LOOFA)

The LOOFA is a work-conserving scheduling algorithm [16]. It provides 100% throughput and a cell delay bound for feasible traffic, using a speedup of 2. An input-output-queued architecture is considered. Two versions of this scheme have been presented: the greedy and the best-first. This scheme considers three different parameters associated with a cell, say cell c , to perform a match: the number of cells in its destined output queue, or *output occupancy*, $\text{OCC}(c)$; the timestamp of a cell, or cell age, $\text{TS}(c)$; and the smallest port number, to break ties. Under the speedup of 2, each time slot has two phases. During each phase, the *greedy version* of this algorithm works as follows (see Fig. 3.23 for an example):

1. Initially, all inputs and outputs are unmatched.
2. Each unmatched input selects an active VOQ (i.e., a VOQ that has at least one cell queued) going to the unmatched output with the lowest

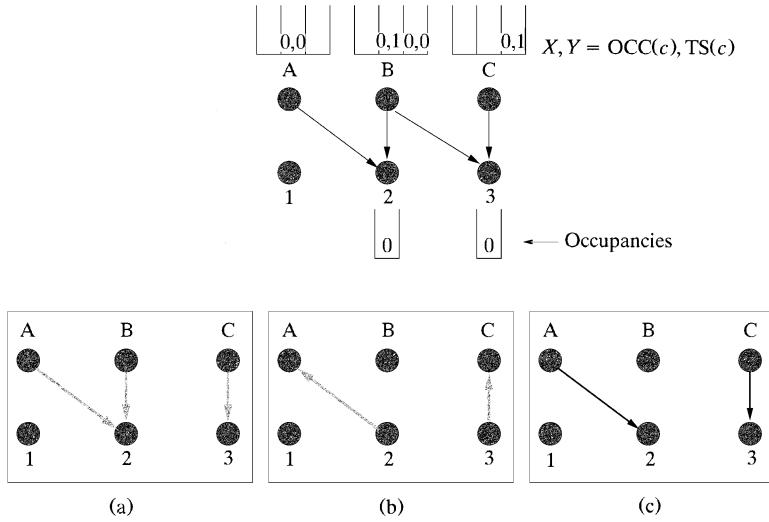


Fig. 3.23 A matching example with the greedy LOOFA.

occupancy, and sends a request to that output. Ties are broken by selecting the smallest output port number. See Figure 3.23(a).

3. Each output, on receiving requests from multiple inputs, selects the one with the smallest OCC and sends the grant to that input. Ties are broken by selecting the smallest port number.
4. Return to step 2 until no more connections can be made.

An example of the greedy version is shown in Figure 3.23. The tuple X, Y in the VOQ represents the output occupancy $OCC(c)$ and the timestamp $TS(c)$ of cell c , respectively. In the upper part of the figure, the arrows indicate the destinations for all different cells at the input ports. The gray arrows in the lower part of the figure indicate the exchange of requests and grants. The black arrows indicate the final match. Part (a) shows that each input sends a request to the output with the lowest occupancy. Output 2 receives two requests, one from A and the other from B , while output 3 receives a request from input C . Part (b) illustrates that, between the two requests, output 2 chooses input A , the one with lower TS. Output 3 chooses the only request, input C .

The *best-first* version works as follows:

1. Initially, all inputs and outputs are unmatched.
2. Among all unmatched outputs, the output with the lowest occupancy is selected. Ties are broken by selecting the smallest output port number. All inputs that have a cell destined for the selected output send a request to it.
3. The output selects the cell request input with the smallest time stamp and sends the grant to the input. Ties are broken by selecting the smallest input port number.
4. Return to step 2 until no more connections can be made (or N iterations are completed).

Figure 3.24 shows a matching example with the best-first version. The selection of the output with the lowest $OCC(c)$ results in a tie: Outputs 2 and 3 have the lowest OCC. This tie is broken by selecting output 2, since this port number is the smaller. Therefore, inputs A and B send a request to this output as shown in part (b), while part (c) illustrates that output 2 grants the oldest cell, input A . Part (d) shows the matching result after the first iteration. The second iteration begins in part (e) when output 3 is chosen as the unmatched output port with the lowest OCC with requests from inputs B and C . Input B is chosen in part (f) for its lowest $TS(c)$. Part (g) depicts the final match.

Both algorithms achieve a maximal matching, with the greedy version achieving it in less iterations. On the other hand, it has been proven that, when combined with the oldest-cell-first input selection scheme, the best-first

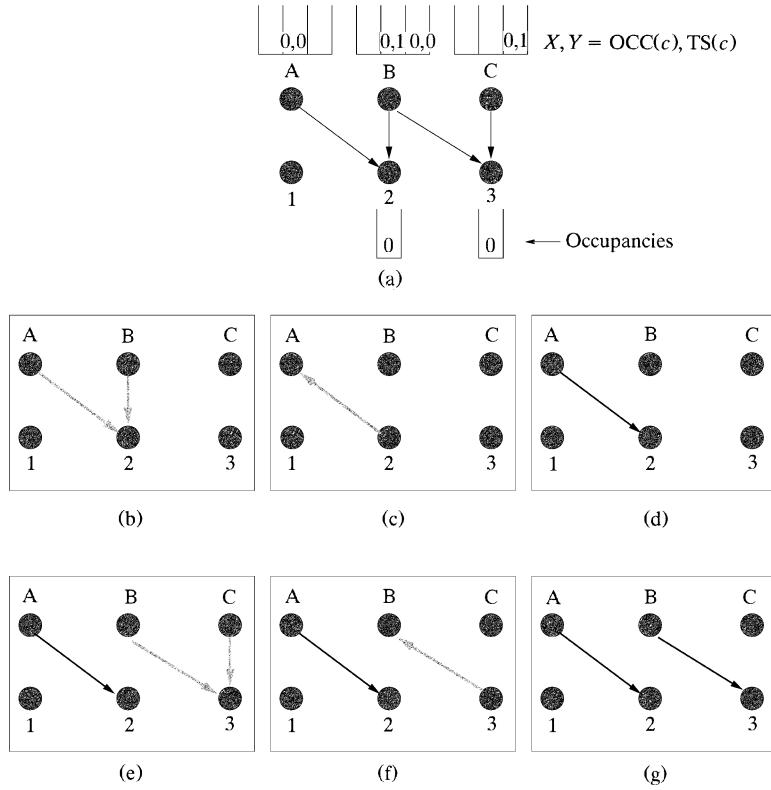


Fig. 3.24 A matching example with the best-first version of LOOFA.

version provides delay bounds for rate-controlled input traffic under a speedup of 2. Denote by D_a and D_o the arbitration delay and the output queuing delay of any cell. It can be shown that that $D_a \leq 4N/(S - 1)$ and $D_o \leq 2N$ cell slots, where S is the speedup factor.

REFERENCES

1. M. Ajmone, A. Bianco, and E. Leonardi, “RPA: a simple efficient and flexible policy for input buffered ATM switches,” *IEEE Commun. Lett.*, vol. 1, no. 3, pp. 83–86, May 1997.
2. T. Anderson, S. Owicki, J. Saxe, and C. Thacker, “High speed scheduling for local area networks,” *Proc. 5th Int. Conf. on Architecture Support for Programming Languages and Operating Systems*, pp. 98–110, Oct. 1992.
3. H. J. Chao and J. S. Park, “Centralized contention resolution schemes for a large-capacity optical ATM switch,” *Proc. IEEE ATM Workshop*, Fairfax, VA, May 1998.

4. H. J. Chao, "Satrun: a terabit packet switch using dual round-robin," *IEEE Commun. Mag.*, vol. 38, no. 12, pp. 78–79, Dec 2000.
5. A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithm for providing bandwidth and delay guarantees in input-buffered crossbars with speedup," *Proc. IEEE IWQoS '98*, May 1998, pp. 235–244.
6. J. S.-C. Chen and T. E. Stern, "Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch," *IEEE J. Select. Areas Commun.*, vol. 9, no. 3, pp. 439–449, Apr. 1991.
7. S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE J. Select. Areas Commun.*, vol. 17, no. 6, pp. 1030–1039, Jun. 1999.
8. A. Descloux, "Contention probabilities in packet switching networks with strung input processes," *Proc. ITC 12*, 1988.
9. H. ElGebaly, J. Muzio, and F. ElGuibaly, "Input smoothing with buffering: a new technique for queuing in fast packet switching," *Proc. IEEE Pacific Rim Conf. on Communications, Computer, and Signal Processing*, pp. 59–62, 1995.
10. K. Genda, Y. Doi, K. Endo, T. Kawamura, and S. Sasaki, "A 160 Gb/s ATM switching system using an internal speed-up crossbar switch," *Proc. GLOBECOM '94*, Nov. 1994, pp. 123–133, 1998.
11. J. N. Giacopelli, W. D. Sincoskie, and M. Littlewood, "Sunshine: a high performance self routing broadband packet switch architecture," *Proc. Int. Switching Symp.*, Jun. 1990.
12. R. Guerin and K. N. Sivarajan, "Delay and throughput performance of speeded-up input-queueing packet switches," IBM Research Report RC 20892, Jun. 1997.
13. M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1587–1597, Dec. 1988.
14. A. Huang and S. Knauer, "Starlite: a wideband digital switch," *Proc. IEEE Globecom '84*, pp. 121–125, Dec. 1984.
15. I. Iliadis and W. E. Denzel, "Performance of packet switches with input and output queueing," *Proc. ICC '90*, Atlanta, GA, pp. 747–753, Apr. 1990.
16. P. Krishna, N. Patel, A. Charny, and R. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE J. Select. Areas Commun.*, vol. 17, no. 6, June 1999.
17. R. O. LaMaire and D. N. Serpanos, "Two dimensional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Trans. on Networking*, Vol. 2, No. 5, Oct. 1994.
18. T. T. Lee, "A modular architecture for very large packet switches," *IEEE Trans. Commun.*, vol. 38, no. 7, pp. 1097–1106, Jul. 1990.
19. S. Q. Li, "Performance of a nonblocking space-division packet switch with correlated input traffic," *IEEE Trans. Commun.*, vol. 40, no. 1, pp. 97–107, Jan. 1992.
20. S. C. Liew, "Performance of various input-buffered and output-buffered ATM switch design principles under bursty traffic: simulation study," *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 1371–1379, Feb./Mar./Apr. 1994.
21. P. Newman, "A fast packet switch for the integrated services backbone network," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1468–1479, 1988.

22. M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet scheduling in input-queued cell-based switches," *IEEE J. Select. Areas Commun.*, 1998, 1998.
23. N. McKeown, P. Varaiya, and J. Warland, "Scheduling cells in an input-queued switch," *IEEE Electron. Lett.*, pp. 2174–2175, Dec. 9th 1993.
24. N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. thesis, *University of California at Berkeley*, 1995.
25. N. McKeown, "The *i*SLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188–201, Apr. 1999.
26. A. Mekkitikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," in *Proc. ICCCN'96*, 1996.
27. Y. Oie, M. Murata, K. Kubota, and H. Miyahara, "Effect of speedup in nonblocking packet switch," *Proc. ICC '89*, pp. 410–414, Jun. 1989.
28. A. Pattavina and G. Bruzzi, "Analysis of input and output queueing for nonblocking ATM switches," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 314–328, Jun. 1993.
29. B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," Technical Report, *CSL-TR-97-738*, Computer Lab., Stanford University.
30. A. Smiljanić, R. Fan, and G. Ramamurthy, "RRGS—round-robin greedy scheduling for electronic/optical switches," *Proc. IEEE Globecom '99*, pp. 1244–1250, 1999.
31. A. Smiljanić, "Flexible bandwidth allocation in terabit packet switches," *Proc. IEEE Conf. on High-Performance Switching and Routing*, pp. 223–241, 2000.
32. I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input and output queueing switch," *Proc. IEEE IWQoS '98*, pp. 218–224, May 1998.

CHAPTER 4

SHARED-MEMORY SWITCHES

In shared-memory switches, all input and output ports have access to a common memory. In every cell time slot, all input ports can store incoming cells and all output ports can retrieve their outgoing cells (if any). A shared-memory switch works essentially as an output-buffered switch, and therefore also achieves the optimal throughput and delay performance. Furthermore, for a given cell loss rate, a shared-memory switch requires less buffers than other switches.

Because of centralized memory management to achieve buffer sharing, however, the switch size is limited by the memory read/write access time, within which N incoming and N outgoing cells in a time slot need to be accessed. As shown in the formula given below, the memory access cycle must be shorter than $1/2N$ of the cell slot, which is the transmission time of a cell on the link:

$$\text{memory access cycle} \leq \frac{\text{cell length}}{2 \cdot N \cdot \text{link speed}}.$$

For instance, with a cell slot of $2.83 \mu\text{s}$ (53-byte cells at the line rate of 149.76 Mbit/s , or $155.52 \text{ Mbit/s} \times 26/27$) and with a memory cycle time of 10 ns , the switch size is limited to 141.

Several commercial ATM switch systems based on the shared memory architecture provide a capacity of several tens of gigabits per second. Some people may argue that memory density doubles every 18 months and so the memory saving by the shared-memory architecture is not that significant. However, since the memory used in the ATM switch requires high speed

(e.g., 5–10-ns cycle time), it is expensive. Thus, reducing the total buffer size can considerably reduce the implementation cost. Some shared-memory switch chip sets have the capability of integrating with other space switches to build a large-capacity switch (e.g., a few hundred gigabits per second).

Although the shared-memory switch has the advantage of saving buffer size, the buffer can be occupied by one or a few output ports that are congested and thus leave no room for other cells destined for other output ports. Thus, there is normally a cap on the buffer size that can be used by any output port.

The following sections discuss different approaches to organize the shared memory and necessary control logics. The basic idea of the shared-memory switch is to use logical queues to link the cells destined for the same output port. Section 4.1 describes this basic concept, the structure of the logical queues, and the pointer processing associated with writing and reading cells to and from the shared memory. Section 4.2 describes a different approach to implement the shared-memory switch by using a content-addressable memory (CAM) instead of a random access memory (RAM) as in most approaches. Although CAM is not as cost-effective and fast as RAM, the idea of using CAM to implement the shared-memory switch is interesting because it eliminates the need of maintaining logical queues. The switch size is limited by the memory chip's speed constraint, but several approaches have been proposed to increase it, such as the space-time-space approach in Section 4.3 and multistage shared-memory switches in Section 4.4. Section 4.5 describes several shared-memory switch architectures to accommodate multicasting capability.

4.1 LINKED LIST APPROACH

Figure 4.1 illustrates the concept of the shared-memory switch architecture. Cells arriving on all input lines are time-division multiplexed into a single stream, which is then converted to a parallel word stream and fed to the

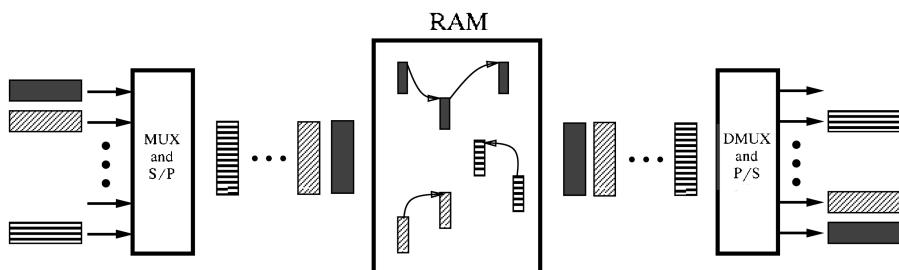
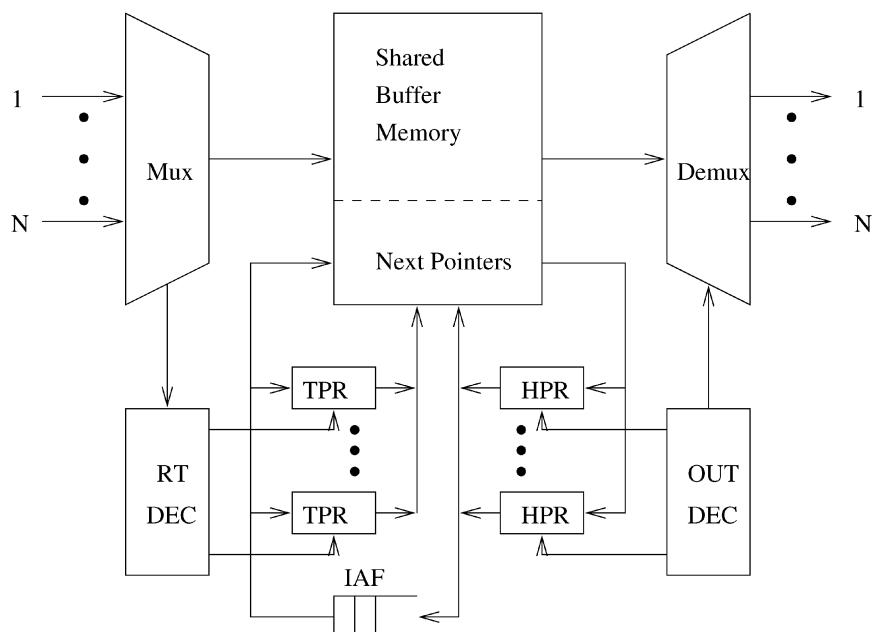


Fig. 4.1 Logical queues in a shared-memory switch.

common memory for storage. Internally to the memory, cells are organized into separate logical queues, one for each output line. Cells destined for the same output port are linked together in the same logical queue. On the other hand, an output stream of cells is formed by retrieving the head-of-line (HOL) cells from the output queues sequentially, one per queue; the output stream is then time-division demultiplexed, and cells are transmitted on the output lines. Each logical queue is confined by two pointers, the head pointer (HP) and the tail pointer (TP). The former points to the first cell of a logical queue, while the latter points to the last cell of a logical queue or to a vacant location to which the next arriving cell will be stored.

Figure 4.2 depicts a linked-list-based shared-memory switch where a logical queue is maintained for each output to link all cells in the memory destined for the output. Each logical queue is essentially operated as a FIFO queue.

The switch operation is as follows. Incoming cells are time-division multiplexed to two synchronized streams: a stream of data cells to the memory,



Mux: Multiplexer

RT DEC: Route decoder

TPR: Tail Pointer Register

IAF: Idle Address FIFO

Demux: Demultiplexer

OUT DEC: Output Decoder

HPR: Head Pointer Register

Fig. 4.2 Basic structure of a linked-list-based shared-memory switch.

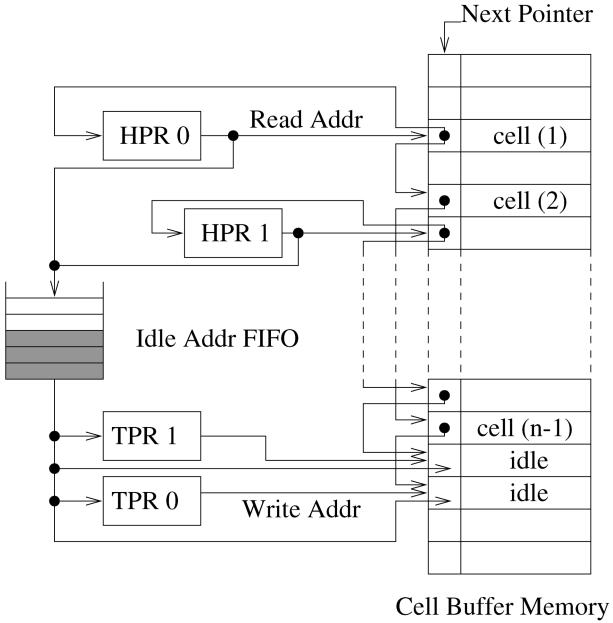


Fig. 4.3 Two linked-list logical queues.

and a stream of the corresponding headers to the route decoder (RT DEC) for maintaining logical queues. The RT DEC then decodes the header stream one by one, accesses the corresponding tail pointer register (TPR), and triggers the WRITE process of the logical queue. An idle address of the cell memory is simultaneously read from an idle address FIFO (IAF) that holds all vacant cell locations of the shared memory.¹ This is the next-address pointer in the cell memory, pointing to the address for storing the next arriving cell. This address is also put into the corresponding TPR to update the logical queue. Figure 4.3 shows two logical queues, where next pointers (NPs) are used to link the cells in the same logical queue. TPR 0 and head pointer register (HPR) 0 correspond to output port 0, where $n - 1$ cells are linked together. TPR 1 and HPR 1 correspond to output port 1. The end of each logical queue indicates the address into which the next cell to the output will be written.

The READ process of the switch is the reverse of the WRITE process. The cells pointed to by the HPRs (heads of the logical queues) are read out of the memory one by one. The resulting cell stream is demultiplexed into the outputs of the switch. The addresses of the leaving cells then become idle, and will be put into the IAF.

¹Since FIFO chips are more expensive than memory chips, an alternative to implementing the IAF is to store vacant cells' addresses in memory chips rather than in FIFO chips, and link the addresses in a logical queue. However, this approach requires more memory access cycles in each cell slot than using FIFO chips and thus can only support a smaller switch size.

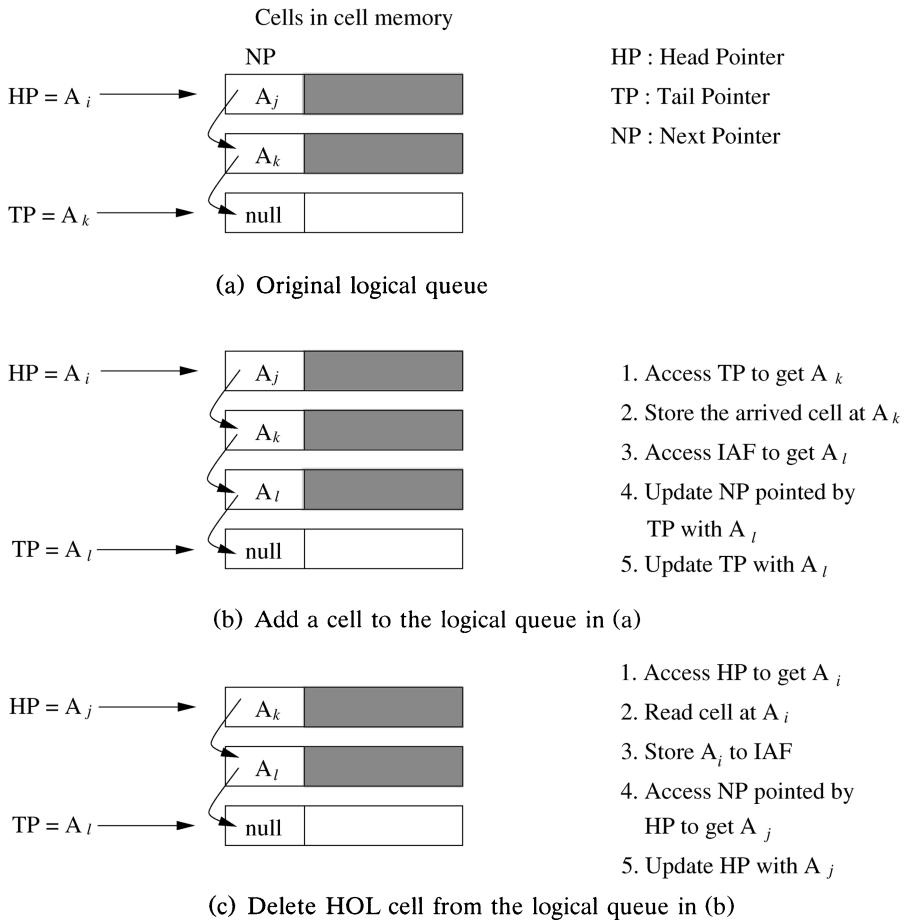


Fig. 4.4 Adding and deleting a cell in a logical queue for the tail pointer pointing to a vacant location.

Figure 4.4 illustrates an example of adding and deleting a cell from a logical queue. In this example, the TP points to a vacant location in which the next arriving cell will be stored. When a cell arrives, it is stored in the cell memory at a vacant location pointed to by the TP [A_k in Fig. 4.4(a)]. An empty location is obtained from the IAF (e.g., A_l). The NP field pointed to by the TP is changed from null to the next arriving cell's address [A_l in Fig. 4.4(b)]. After that, the TP is replaced by the next arriving cell's address (A_l). When a cell is to be transmitted, the HOL cell of the logical queue pointed to by the HP [A_i in Fig. 4.4(b)] is accessed for transmission. The cell to which the NP points [A_j in Fig. 4.4(b)] becomes the HOL cell. As a result, the HP is updated with the new HOL cell's address [A_j in Fig. 4.4(c)]. Meanwhile, the vacant location (A_i) is put into in the IAF for use by future arriving cells.

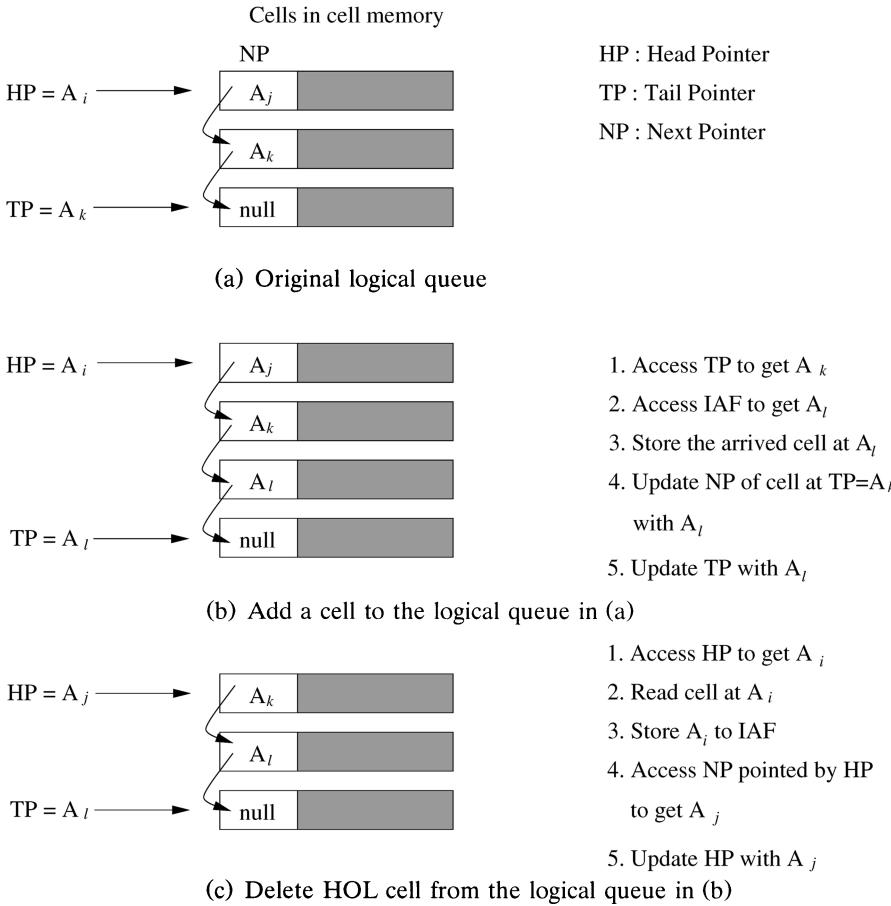


Fig. 4.5 Adding and deleting a cell in a logical queue for the tail pointer pointing to the last cell of the queue.

Figure 4.5 illustrates an example of adding and deleting a cell in a logical queue for the tail pointer pointing to the last cell of the queue. The writing procedures are different in Figures 4.4 and 4.5, while the reading procedures are identical. While the approach in Figure 4.4 may have the advantage of better applying parallelism for pointer processing, it wastes one cell buffer for each logical queue. For a practical shared-memory switch with 32 or 64 ports, the wasted buffer space is negligible compared with the total buffer size of several tens of thousand. However, if the logical queues are arranged on a per connection basis (e.g., in packet fair queueing scheduling), the wasted buffer space can be quite large, as the number of connections can be several tens of thousands. As a result, the approach in Figure 4.5 will be a better choice in this situation.

As shown in Figures 4.4 and 4.5, each write and read operation requires 5 memory access cycles.² For an $N \times N$ switch, the number of memory access cycles in each time slot is $10N$. For a time slot of $2.83 \mu\text{s}$ and a memory cycle time of 10 ns, the maximum number of switch ports is 28, which is much smaller than the number obtained when considering only the time constraint on the data path. (For example, it was concluded at the beginning of the chapter that using the same memory technology, the switch size can be 141. That is because we did not consider the time spent on updating the pointers of the logical queues.) As a result, pipeline and parallelism techniques are usually employed to reduce the required memory cycles in each time slot. For instance, the pointer updating and cell storage/access can be executed simultaneously. Or the pointer registers can be stored inside a pointer processor instead of in an external memory, thus reducing the access time and increasing the parallelism of pointer operations.

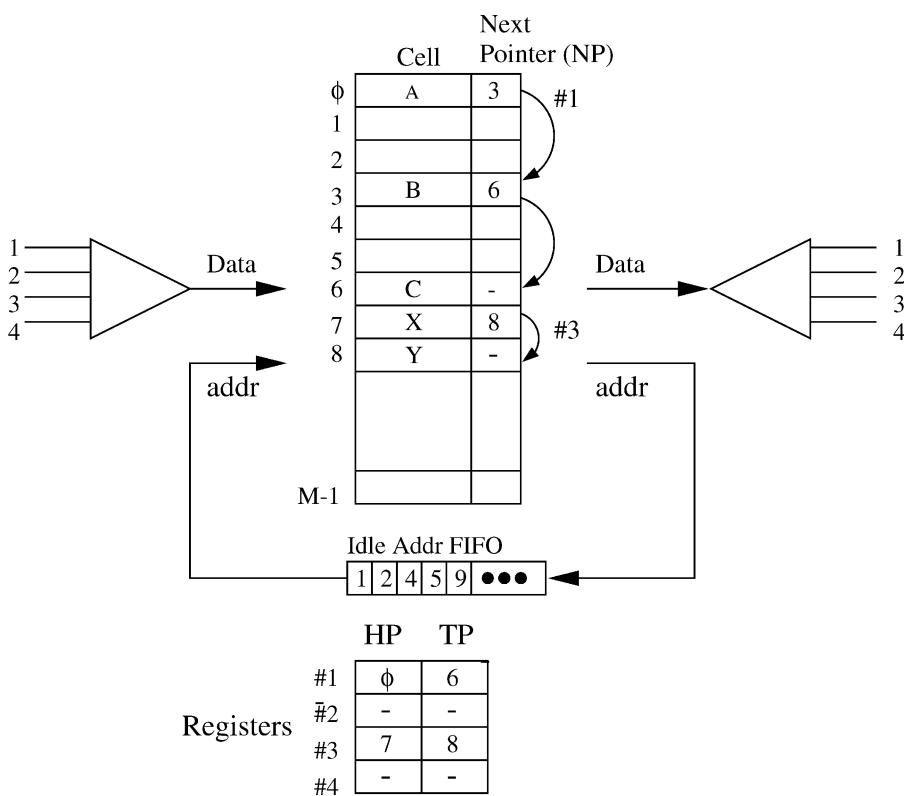


Fig. 4.6 An example of a 4×4 shared-memory switch.

²Here, we assume the entire cell can be written to the shared memory in one memory cycle. In real implementations, cells are usually written to the memory in multiple cycles. The number of cycles depends on the memory bus width and the cell size.

Figure 4.6 shows an example of a 4×4 shared-memory switch. In this example, the TP points to the last cell of a logical queue. Cells A , B , and C , destined for output port 1, form a logical queue with HP = 0 and TP = 6 and are linked by NPs as shown in the figure. Cells X and Y , destined for output port 3 form another logical queue with HP = 7 and TP = 8. Assuming a new cell D destined for output 1 arrives, it will be written to memory location 1 (this address is provided by the IAF). Both the NP at location 6 and TP 1 are updated with value 1. When a cell from output 1 is to be read for transmission, HP 1 is first accessed to locate the HOL cell (cell A at location 0). Once it is transmitted, HP 1 is updated with the NP at location 0, which is 3.

Alternatively, the logical queues can be organized with dedicated FIFO queues, one for each output port [10]. The switch architecture is shown in Figure 4.7. The performance is the same as using linked lists. The necessary amount of buffering is higher due to the buffering required for the FIFO queues, but the implementation is simpler and the multicasting and priority control easier to implement. For instance, when an incoming cell is broadcast to all output ports, it requires at least N pointer updates in Figure 4.2, while in Figure 4.7 the new address of the incoming cell can be written to all FIFO queues at the same time. The size of each FIFO in Figure 4.7 is $M \log M$, while the size of each pointer register in Figure 4.2 is only $\log M$, where M is the number of cells that can be stored in the shared memory. This approach

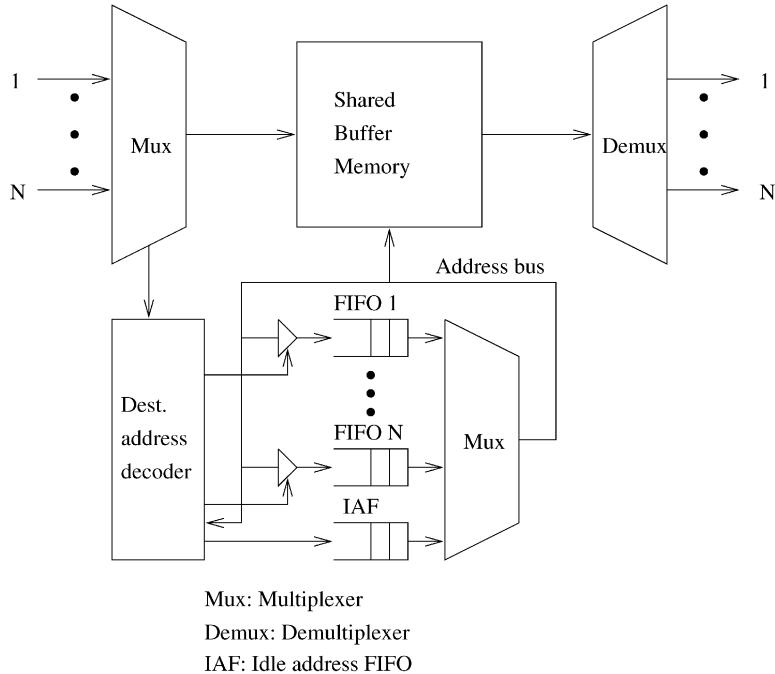


Fig. 4.7 Using FIFOs to maintain the logical queues.

provides better reliability than the linked-list approach (Fig. 4.2) in that if the cell address stored in FIFO is corrupted due to hardware failure, only one cell is affected. However, in the linked-list approach, if the pointer in the linked list is corrupted, cells in the remaining list will be either routed to an incorrect output port, or never be accessed.

4.2 CONTENT-ADDRESSABLE MEMORY APPROACH

In the CAM-based switch [14], the shared-buffer RAM is replaced by a CAM-RAM structure, where the RAM stores the cell and the CAM stores a tag used to reference the cell. This approach eliminates the need of maintaining the logical queues. A cell can be uniquely identified by its output port number and a sequence number, and these together constitute the tag. A cell is read by searching for the desired tag. Figure 4.8 shows the switch architecture. Write circuits serve input ports and read circuits serve output ports, both on a round-robin basis, as follows.

For a *write*:

1. Read the write sequence number $WS[i]$ from the write sequence RAM (WSRAM) (corresponding to the destination port i), and use this value (s) for the cell's tag $\{i, s\}$.

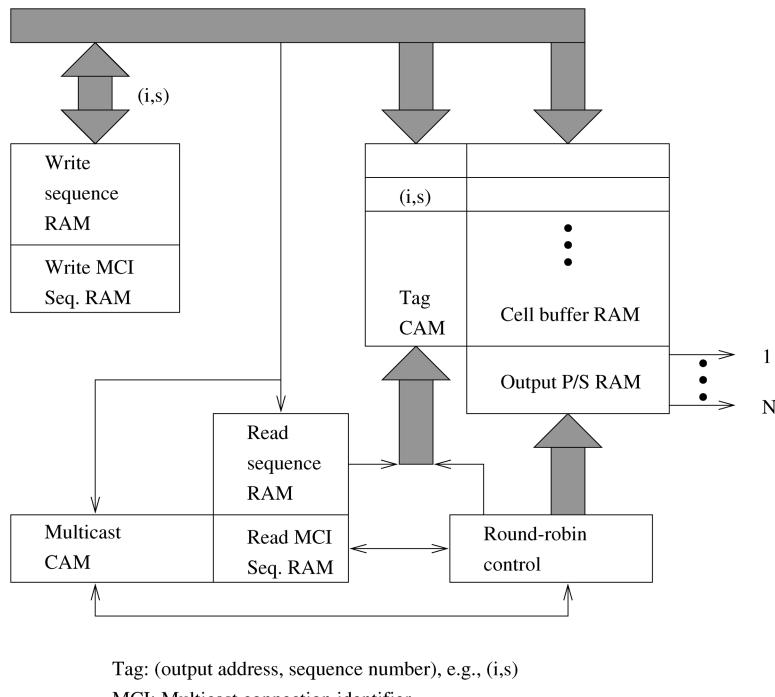


Fig. 4.8 Basic architecture of a CAM-based shared-memory switch.

2. Search the tag CAM for the first empty location, emp .
3. Write the cell into the buffer $B[\text{emp}] = \text{cell}$, and $\{i, s\}$ into the associated tag.
4. Increment the sequence number s by one, and update $\text{WS}[i]$ with $s + 1$.

For a *read*:

1. Read the read sequence number $\text{RS}[j]$ from the read sequence RAM (RSRAM) (corresponding to the destination port j), say t .
2. Search for the tag with the value $\{j, t\}$.
3. Read the cell in the buffer associated with the tag with the value $\{j, t\}$.
4. Increment the sequence number t by one and update $\text{RS}[i]$ with $t + 1$.

This technique replaces link storage indirectly with content-addressable tag storage, and read/write pointers/registers with sequence numbers. A single extra “validity” bit to identify if the buffer is empty is added to each tag, effectively replacing the IAF (and its pointers). No address decoder is required in the CAM-RAM buffer. Moreover, since the CAM does not output the address of tag matches (*hits*), it requires no address encoder (usually a source of CAM area overhead). Both linked-list address registers and CAM-access sequence number registers must be initialized to known values on power-up. If it is necessary to monitor the length of each queue, additional counters are required for the linked-list case, while sequence numbers can simply be subtracted in the CAM-access case. Table 4.1 summa-

TABLE 4.1 Comparison of Linked List and CAM Access^a

Bits	Linked List	CAM Access
Cell storage (decode/encode)	RAM (decode) $256 \times 424 = 108,544$	CAM/RAM (neither) $256 \times 424 = 108,544$
lookup	Link: RAM $256 \times 8 = 2048$	Tag:CAM $256 \times (4 + 7) = 2816$
Write and read reference (queue length checking)	Address registers (additional counters) $2 \times 16 \times 8 = 256$	Sequence number registers (compare W and R numbers) $2 \times 16 \times 7 = 224$
Idle address storage (additional overhead)	IAF (pointer maintenance, extra memory block) $256 \times 8 = 2048$	CAM valid bit (none)
Total	112,896	111,840

^aSample switch size is 16×16 with $2^8 = 256$ -cell buffer capacity; 7-bit sequence numbers are used.

rizes this comparison, and provides bit counts for an example single-chip configuration. Although the number of bits in the linked-list and CAM-access approaches are comparable, the latter is less attractive, due to the lower speed and higher implementation cost for the CAM chip.

4.3 SPACE-TIME-SPACE APPROACH

Figure 4.9 depicts an 8×8 example to show the basic configuration of the space-time-space (STS) shared-memory switch [12]. Separate buffer memories are shared among all input and output ports via crosspoint space-division switches. The multiplexing and demultiplexing stages in the traditional shared-memory switch are replaced with crosspoint space switches; thus the resulting structure is referred to as STS-type. Since there is no time-division multiplexing, the required memory access speed may be drastically reduced.

The WRITE process is as follows. The destination of each incoming cell is inspected in a header detector, and is forwarded to the control block that controls the input-side space switch and thus the connection between the

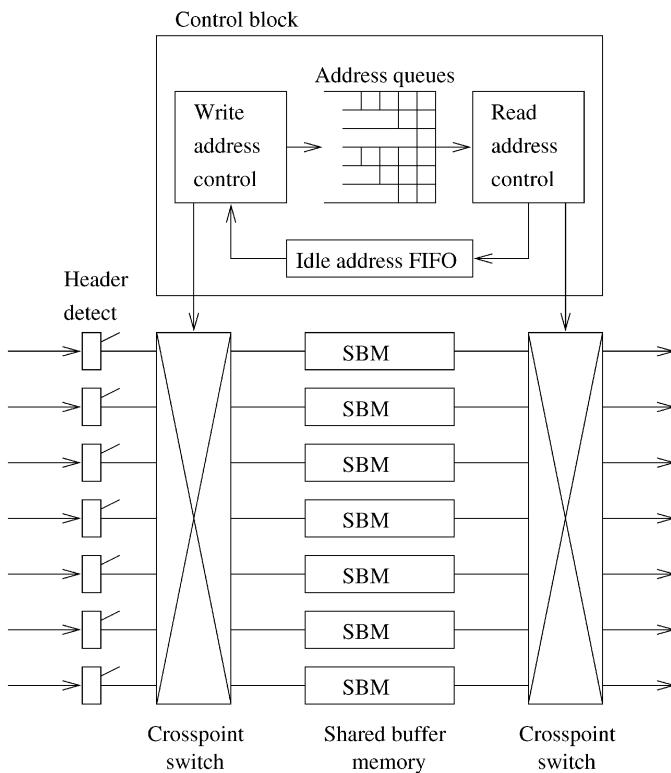


Fig. 4.9 Basic configuration of STS-type shared-memory ATM switch.

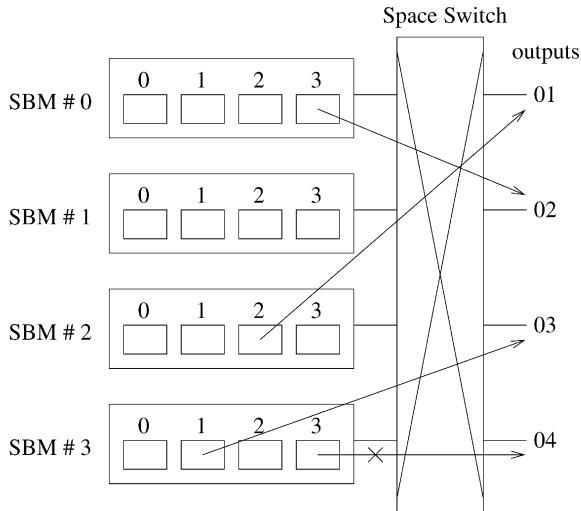


Fig. 4.10 Blocking in STS-type shared-memory switch.

input ports and the buffer memories. As the number of shared-buffer memories (SBMs) is equal to or greater than the number of input ports, each incoming cell can surely be written into an SBM as long as no SBM is full. In order to realize the buffer sharing effectively, cells are written to the least-occupied SBM first and the most-occupied SBM last. When a cell is written into an SBM, its position (the SBM number and the address of the cell in the SBM) is queued in the address queue. The read address selector picks out the first address from each address queue and controls the output-side space switch to connect the picked cells (SBMs) with the corresponding output ports.

It may occur that two or more cells picked for different output ports are from the same SBM. An example is shown in Figure 4.10. Thus, to increase the switch's throughput it requires some kind of internal speedup to allow more than one cell to be read out from an SBM in every cell slot. Another disadvantage of this switch is the requirement of searching for the least-occupied SBM (may need multiple searches in a time slot), which may cause a system bottleneck when the number of SBMs is large.

4.4 MULTISTAGE SHARED-MEMORY SWITCHES

Several proposed switch architectures interconnect small-scale shared-memory switch modules with a multistage network to build a large-scale switch. Among them are Washington University gigabit switch [16], concentrator-based growable switch architecture [4], multinet switch [8], Siemens switch [5], and Alcatel switch [2].

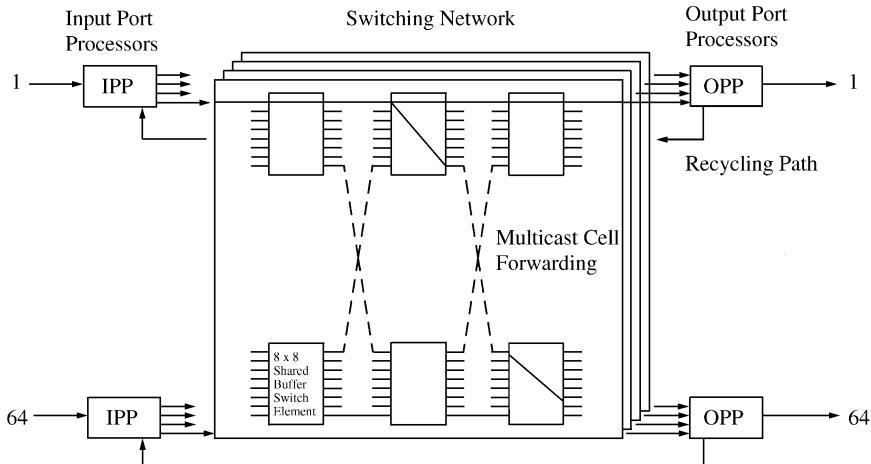


Fig. 4.11 Washington University gigabit switch. (© 1997 IEEE.)

4.4.1 Washington University Gigabit Switch

Turner proposed an ATM switch architecture called the Washington University gigabit switch (WUGS) [16]. The overall architecture is shown in Figure 4.11. It consists of three main components: the input port processors (IPPs), the output port processors (OPPs), and the central switching network. The IPP receives cells from the incoming links, buffers them while awaiting transmission through the central switching network, and performs the virtual-path–circuit translation required to route cells to their proper outputs. The OPP resequences cells received from the switching network and queues them while they await transmission on the outgoing link. This resequencing operation increases the delay and implementation complexity. Each OPP is connected to its corresponding IPP, providing the ability to recycle cells belonging to multicast connections. The central switching network is made up of switching elements (SEs) with eight inputs and eight outputs and a common buffer to resolve local contention. The SEs switch cells to the proper output using information contained in the cell header, or distribute cells dynamically to provide load balancing. Adjacent switch elements employ a simple hardware flow control mechanism to regulate the flow of cells between successive stages, eliminating the possibility of cell loss within the switching network.

The switching network uses a Benes network topology. The Benes network extends to arbitrarily large configurations by way of a recursive expansion. Figure 4.11 shows a 64-port Benes network. A 512-port network can be constructed by taking 8 copies of the 64-port network and adding the first and fifth stage on either side, with 64 switch elements a copy. Output j of the i th switch element in the first stage is then connected to input i of the j th 64-port network. Similarly, output j of the i th 64-port network is connected

to input i of the j th switch element in the fifth stage. Repeating in this way, a network of arbitrary large size can be theoretically obtained. For $N = 8^k$, the Benes network constructed using 8-port switch elements has $2k - 1$ stages. Since $k = \log_8 N$, the number of switch elements scales in proportion to $N \log_8 N$, which is the best possible scaling characteristic. In [16], it is shown that when dynamic load distribution is performed in the first $k - 1$ stages of the Benes network, the load on the internal data paths of the switching network cannot exceed the load on the external ports; that is, the network achieves ideal load balancing. This is true for both point-to-point and multipoint traffic.

4.4.2 Concentrator-Based Growable Switch Architecture

A concentrator-based growable switch architecture is shown in Figure 4.12 [4]. The 8×8 output ATM switches (with the shared-memory structure) are each preceded by an $N \times 8$ concentrator. The $N \times 8$ concentrator is preceded by a front-end broadcast network (i.e., a memoryless cell distribution network). The concentrators are preceded by address filters that only accept cells that are destined to its group of dedicated outputs. The valid cells in each concentrator are then buffered for a FIFO operation. In other words, each concentrator is an N -input 8-output FIFO buffer. The actual ATM cell switching is performed in the ATM output switch. Obviously, this is increasingly challenging as N becomes large. For instance, a 512×8 concentrator can be built using a column of eight 64×8 concentrators followed by another 64×8 concentrator. At 2.5 Gbit/s per port, a 64×64 switch has a capacity of 160 Gbit/s, and a 512×512 switch has a capacity of 1.28 Tbit/s.

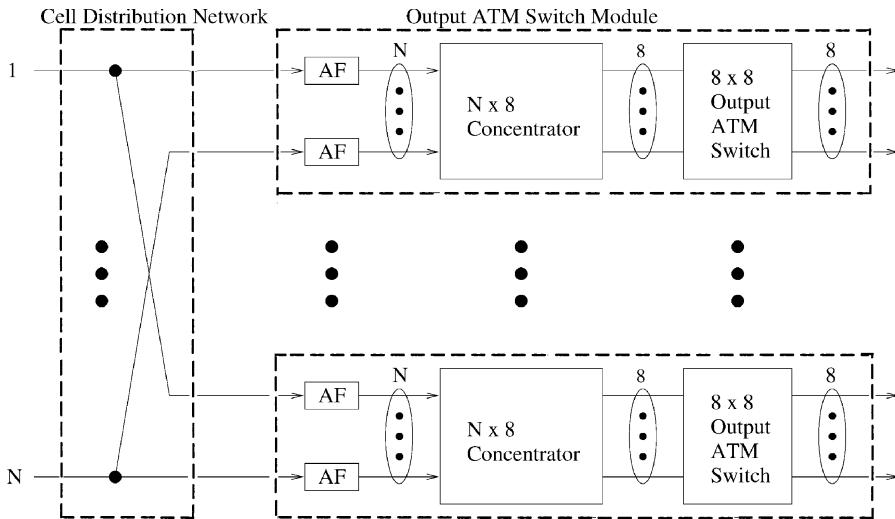


Fig. 4.12 A concentrator-based growable switch architecture.

4.5 MULTICAST SHARED-MEMORY SWITCHES

Of the various ATM switch architectures, the shared-memory switch provides the greatest benefits. Since its memory space is shared among its switch ports, it achieves high buffer utilization efficiency. Under conditions of identical memory size, the cell-loss probability of the shared-memory switches is smaller than that of output-buffered switches. Additionally, multicast operations can be easily supported by this switch architecture. In this section, some classes of multicast shared-memory switches are described along with their advantages and disadvantages.

4.5.1 Shared-Memory Switch with a Multicast Logical Queue

The simplest way to implement multicasting in a shared-memory switch is to link all multicasting cells in a logical queue as shown in Figure 4.13. The advantage of this approach is that the number of logical queues that need to be updated in every cell slot is minimized. The multicast routing table stores routing information, such as a bit map of output ports to which the multicast cells are routed. In this example, multicast cells always have higher priority than unicast cells. That is called *strict priority*, where unicast cells will be served only when the multicast logical queue is empty or the multicast HOL cell is not destined for the output port for which they are destined. Of course, there can be other service policies between unicast and multicast cells

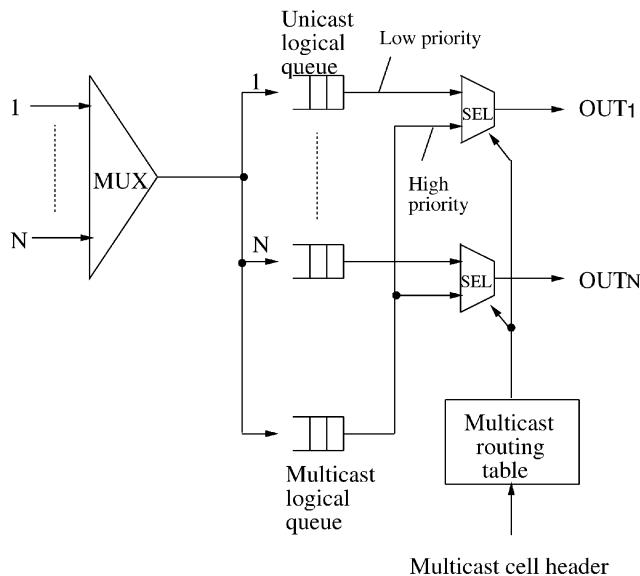


Fig. 4.13 A shared-memory ATM switch with a multicast logical queue.

than the strict priority. For instance, they can be served in a round-robin manner, or a weighted round-robin manner with the weight depending on, for example, the ratio of unicast and multicast traffic load. However, the major disadvantage of this approach is there may be HOL blocking for the multicast logical queue when schemes other than the strict priority are used. This is because when the multicast HOL cell is blocked because of yielding its turn to unicast cells, it may block other multicast cells that are behind it, though they could otherwise have been transmitted to some idle outputs.

4.5.2 Shared-Memory Switch with Cell Copy

In this approach, a multicast cell is replicated into multiple copies for all its multicast connections by a cell-copy circuit located before the routing switch. Bianchini and Kim [7] proposed a nonblocking cell-copy circuit for multicast purpose in an ATM switch. Each copy is stored in the SBM. The address for each copy is queued in output address queues (AQs).

Figure 4.14 shows an example of a shared-memory switch with a cell-copy circuit. A multicast cell arrives and is replicated into three copies for output ports 0, 1, and 3. These three copies of the cell are stored at different locations of the shared buffer memory. Meanwhile, their addresses are stored in corresponding AQs. In this scheme, after replicating the multicast cell, each copy is treated the same way as an unicast cell. It provides fairness among ATM cells. The major disadvantage of this approach is that, in the worst case, N cells might be replicated to N^2 cells. Thus, the number of replicated cells to the SBM in each switch cycle could be $O(N^2)$. Since only

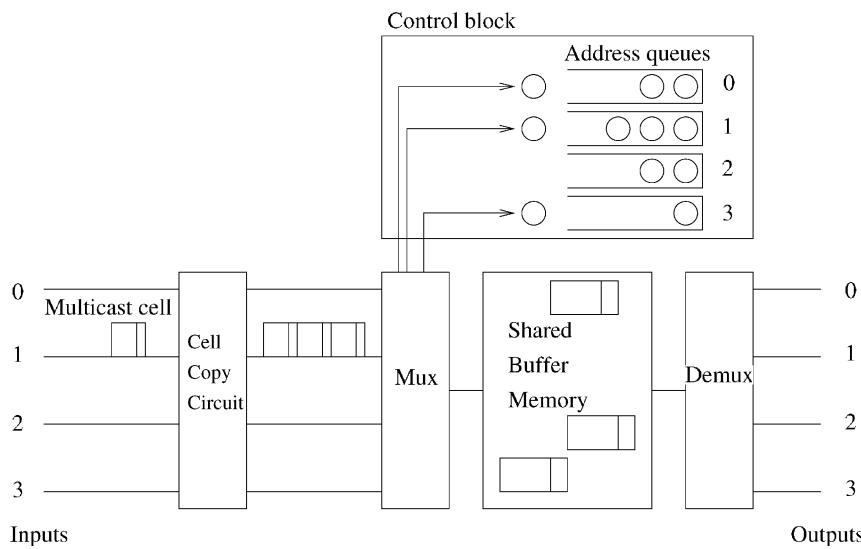


Fig. 4.14 A shared-memory ATM switch with cell-copy circuit.

at most N cells could be transmitted, this would result in storing $O(N^2)$ cells in each cycle. For a finite memory space, it would result in considerable cell loss. The replication of multicast cells would also require $O(N^2)$ cells to be written to the SBM, which would further limit the switch size on account of the memory speed. Moreover, adding the cell-copy circuit in the switch increases the hardware complexity.

4.5.3 Shared-Memory Switch with Address Copy

Saito et al. proposed a new scheme that requires less memory for the same cell loss probability [13]. A similar architecture was proposed in [11]. In this scheme, an address-copy circuit is used in the controller. When a multicast cell arrives, only a single copy is stored in the SBM. Its address is copied by the address-copy circuit and queued into multiple address queues, as shown in Figure 4.15. The multicast cell will be read multiple times to different output ports. Thereafter, the cell's address becomes available for the next arriving cells.

In this architecture, multicast cell counters (MCCs) are required to hold the number of copies of each multicast cell. The MCC values are loaded as new cells (unicast or multicast) arrive and are decreased by one each time a copy of a cell is read out from the SBM.

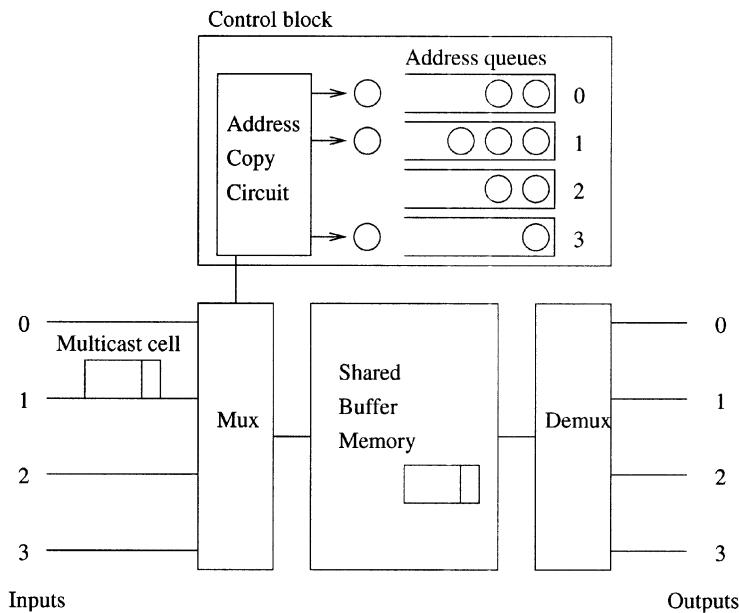


Fig. 4.15 A shared-memory ATM switch with address-copy circuit.

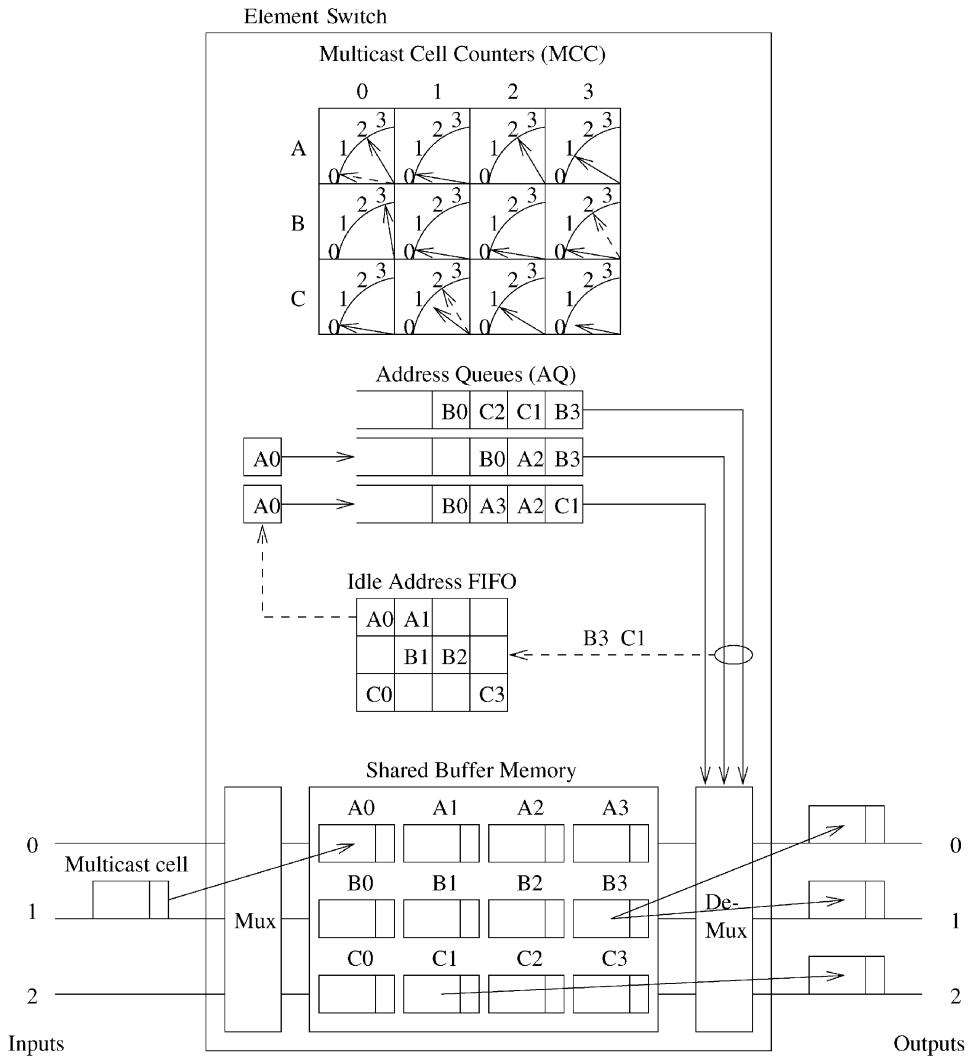


Fig. 4.16 An example of multicast function in an address-copy switch.

Figure 4.16 illustrates an example of a 3×3 ATM switch. In this example, a multicast cell arriving at input port 1 is destined for output ports 1 and 2. The multicast cell is written into A0 in the SBM, and its address is provided by an IAF. Other vacant cells' addresses stored in the IAF are A1, B1, B2, C0, and C3. A0 is copied and queued in the Address queues 1 and 2. At the same time, the MCC of A0 is set to two. Another multicast cell stored at B3 is destined for output ports 0 and 1. As it is transmitted to both output ports 0 and 1, the associated MCC becomes zero and the address B3 is released to

the IAF. Within the same time slot, the cell stored at C1 is read to output port 2. Since it still has one copy to be sent to output port 0, its address C1 is not released until it is transmitted to output port 0.

Although in each time slot the maximum number of cells to be written into the buffer is N , the number of replications of cell addresses for incoming cells could be $O(N^2)$ in the broadcast case, which still limits the switch size. More specifically, since the address queues or the MCCs are in practice stored in the same memory, there can be up to N^2 memory accesses in each cell slot, which may cause a system bottleneck for a large N .

REFERENCES

1. H. Ahmadi and W. E. Denzel, "A survey of modern high-performance switching techniques," *IEEE J. Select. Areas Commun.*, vol. 7, no. 7, pp. 1091–1103, Sep. 1989.
2. T. R. Banniza, G. J. Eilenberger, B. Pauwels, and Y. Therasse, "Design and technology aspects of VLSI's for ATM switches," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1255–1264, Oct. 1991.
3. N. Endo, T. Kozaki, T. Ohuchi, H. Kuwahara, and S. Gohara, "Shared buffer memory switch for an ATM exchange," *IEEE Trans. Commun.*, vol. 41, no. 1, pp. 237–245, Jan. 1993.
4. K. Y. Eng and M. J. Karol, "State of the art in gigabit ATM switching," *Proc. IEEE BSS '95*, Poland, pp. 3–20, Apr. 1995.
5. W. Fischer, O. Fundneider, E. H. Goeldner, and K. A. Lutz, "A scalable ATM switching system architecture," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1299–1307, Oct. 1991.
6. J. Garcia-Haro and A. Jajszczyk, "ATM shared-memory switching architectures," *IEEE Network Magazine*, pp. 18–26, Jul./Aug. 1994.
7. R. P. Bianchini Jr. and H. S. Kim, "Design of a nonblocking shared-memory copy network for ATM," *Proc. IEEE INFOCOM '92*, pp. 876–885, 1992.
8. H. S. Kim, "Design and performance of multinet switch: a multistage ATM switch architecture with partially shared buffers," *IEEE/ACM Trans. Networking*, vol. 2, no. 6, pp. 571–580, Dec. 1994.
9. T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano, "32 × 32 shared buffer type ATM switch VLSI's for B-ISDN's," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1239–1247, Oct. 1991.
10. H. Lee, K. H. Kook, C. S. Rim, K. P. Jun, and S. K. Lim, "A limited shared output buffer switch for ATM," *Fourth Int. Conf. on Data Commun. Systems and Their Performance*, Barcelona, pp. 163–179, Jun. 1990.
11. T.-H. Lee and S. J. Liu, "Multicasting in a shared buffer memory switch," *Proc. IEEE TENCON '93*, Beijing, pp. 209–212, 1993.

12. K. Oshima, H. Yamanaka, H. Saito, H. Yamada, S. Kohama, H. Kondoh, and Y. Matsuda, "A new ATM switch architecture based on STS-type shared buffering and its LSI implementation," *Proc. IEICE*, pp. 359–363, 1992.
13. H. Saito, H. Yamanaka, H. Yamada, M. Tuzuki, H. Kondoh, Y. Matsuda, and K. Oshima, "Multicast function and its LSI implementation in a shared multi-buffer ATM switch," *Proc. IEEE Infocom '94*, pp. 315–322, 1994.
14. K. J. Schultz and P. G. Gulak, "CAM-based single-chip shared buffer ATM switch," *Proc. IEEE ICC '94*, pp. 1190–1195, May 1994.
15. F. A. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks," *Proc. IEEE*, vol. 78, no. 1, pp. 133–167, Jan. 1990.
16. T. Cheney, J. A. Fingerhut, M. Flucke, and J. S. Turner, "Design of a Gigabit ATM Switch," *Proc. IEEE INFOCOM '97*, pp. 1392–1399, 1997.

CHAPTER 5

BANYAN-BASED SWITCHES

The very early theoretical work on multistage interconnection networks (MINs) was done in the context of circuit-switched telephone networks [7, 4]. The aim was to design a nonblocking multistage switch with number of crosspoints less than in a single-stage crossbar switch. After many such networks were studied and introduced for interconnecting multiple processors and memories in parallel computer systems, several types of them, such as *banyan* and *shuffle-exchange* networks, were proposed [20, 12, 21] as switching fabrics because in them several cells can be routed in parallel and the switching function can be implemented regularly in hardware.

In this chapter, we describe banyan-family switches, which have attracted many researchers, over more than two decades, for building interconnection networks. Section 5.1 classifies banyan-family switch architectures according to their nature and properties. Section 5.2 describes Batcher-sorting network switch architecture. Section 5.3 introduces output-contention resolution algorithms in banyan-family switches. Section 5.4 describes the Sunshine switch, which extends the Batcher–banyan switching architecture. Section 5.5 describes some work on deflection routing over banyan-family networks. Section 5.6 introduces a self-routing copy network where the nonblocking property of banyan networks is generalized to support multicasting.

5.1 BANYAN NETWORKS

The banyan class of interconnection networks was originally defined in [9]. It has the property that there is exactly one path from any input to any output.

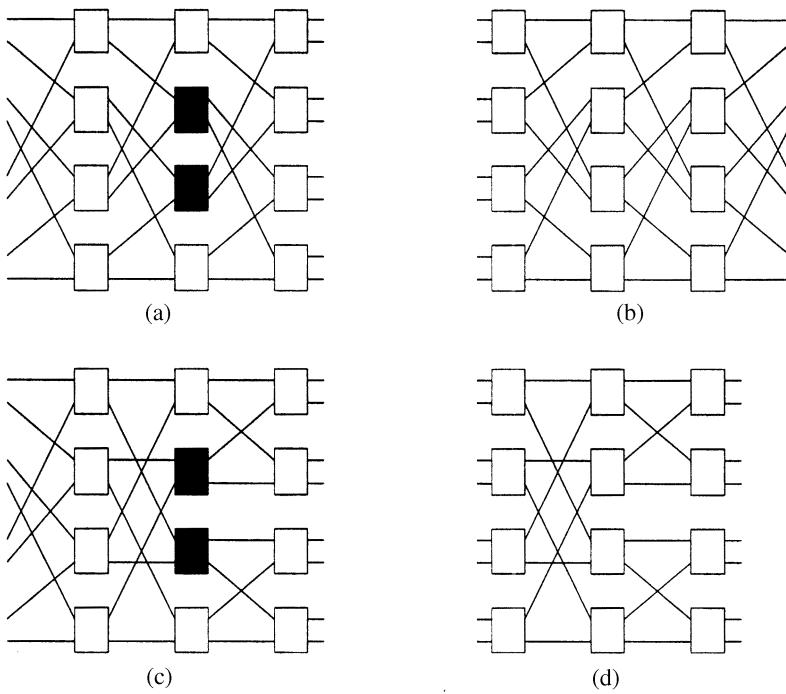


Fig. 5.1 Four different banyan-class networks: (a) shuffle-exchange (omega) network; (b) reverse shuffle-exchange network; (c) narrow-sense banyan network; (d) baseline network. We can see that (a) and (b) are isomorphic by interchanging the two shaded nodes.

Figure 5.1 shows four networks belonging to this class: the shuffle-exchange network (narrow-sense also called the omega network), the reverse shuffle-exchange network, the narrow-sense banyan network, and the baseline network.

The principal common properties of these networks are: (1) they consist of $n = \log_2 N$ stages and $N/2$ nodes per stage,¹ (2) they have the self-routing property that the unique n -bit destination address can be used to route a cell from any input to any output, each bit for one stage, and (3) their regularity and interconnection pattern are very attractive for VLSI implementation. Figure 5.2 shows a routing example in an 8×8 banyan network, where the bold lines indicate the routing paths. On the right hand side, the address of each output destination is labeled as a string of n bits, $b_1 \dots b_n$. A cell's destination address is encoded into the header of the cell. In the first stage, the most significant bit b_1 is examined. If it is a 0, the cell will be forwarded to the upper outgoing link; if it is a 1, the cell will be forwarded to the lower outgoing link. In the next stage, the next most significant bit b_2 will be

¹A regular $N \times N$ network can also be constructed from identical $b \times b$ switching nodes in k stages, where $N = b^k$.

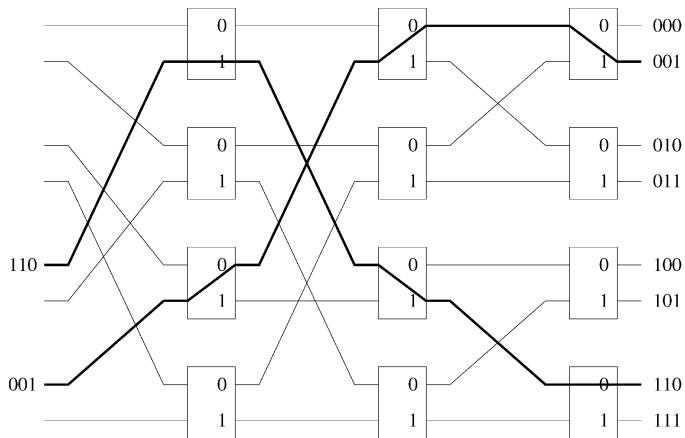


Fig. 5.2 An 8×8 banyan network.

examined and the routing performed in the same manner.

The internal blocking refers to the case where a cell is lost due to the contention on a link inside the network. Figure 5.3 shows an example of internal blocking in an 8×8 banyan network. However, the banyan network will be internally nonblocking if both conditions below are satisfied:

- there is no idle input between any two active inputs;
- the output addresses of the cells are in either ascending order or descending order.

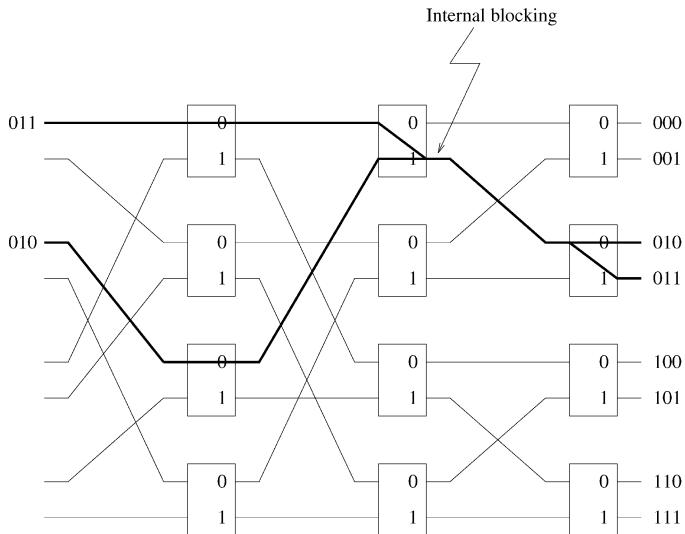


Fig. 5.3 Internal blocking in an 8×8 banyan network.

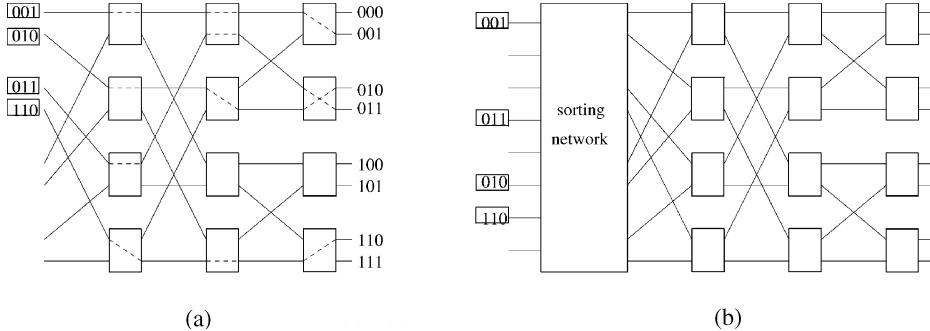


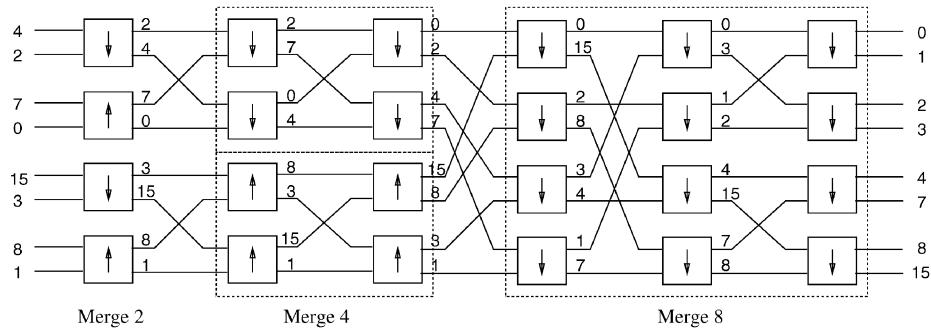
Fig. 5.4 An example showing that the banyan network is nonblocking for sorted inputs. (b) Nonblocking sort–banyan network.

See Figure 5.4. Suppose the banyan network is preceded by a network that concentrates the cells and sorts the cells according to their output destinations. The overall sort–banyan network will be internally nonblocking.

5.2 BATCHER-SORTING NETWORK

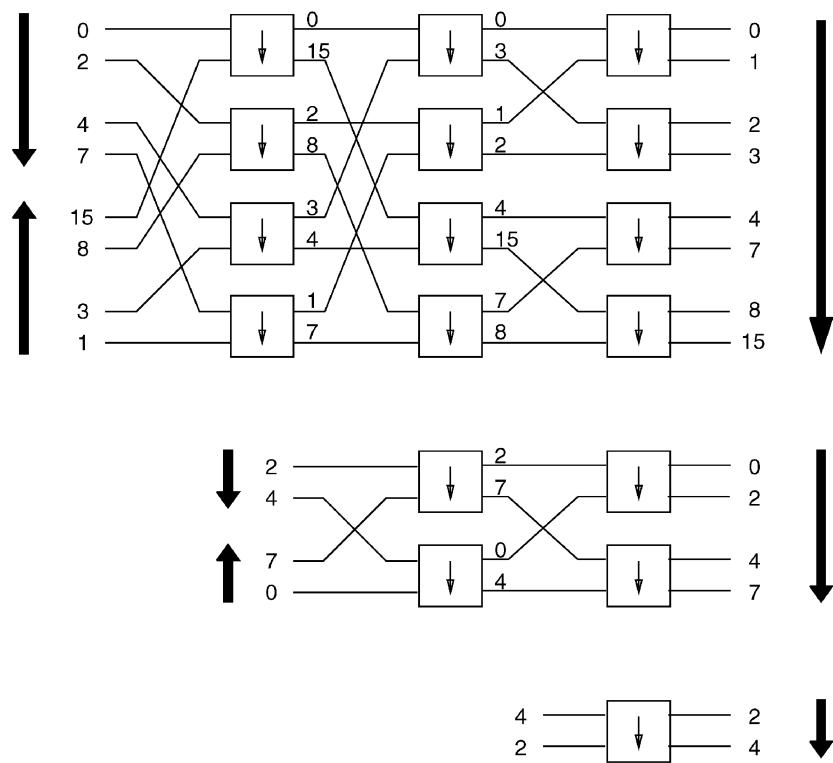
A sorting network is formed by a series of merge networks of different sizes. Figure 5.5(a) shows an 8×8 Batcher-sorting network [3] consisting of merge networks of three different sizes. A merge network [see Fig. 5.5(b)] is built from 2×2 sorting elements in stages, and the pattern of exchange connections between each pair of adjacent stages is the same as in a banyan network. We can observe that, if the order of the destinations of the first half input cells is ascending and that of the second half is descending, then the merge network will sort the cells into an ascending list at the outputs. An 8×8 sorting network will be formed if an 8×8 merge network is preceded by two 4×4 merge networks and four 2×2 merge (sorting) elements. A completely random list of eight input cells will be first sorted into four sorted lists of two cells, then two sorted lists of four cells, and finally a sorted list of eight cells.

A $N \times N$ merge network consists of $\log_2 N$ stages and $(N \log_2 N)/2$ elements. A sorting network has $1 + 2 + \dots + \log_2 N = (\log_2 N)(\log_2 N + 1)/2$ stages and $(N \log_2 N)(\log_2 N + 1)/4$ elements. Figure 5.6 shows a 64×64 Batcher-banyan switch network, where the last six stages of switch elements belong to the banyan network.



- Batcher sorting network consists of multiple merge networks.
- Total number of stages = $(1 + 2 + \dots + \log N) = (\log N / 2)(1 + \log N)$

(a) A Batcher-sorting network



(b) The corresponding merge network

Fig. 5.5 Basic structure of a Batcher-sorting network.

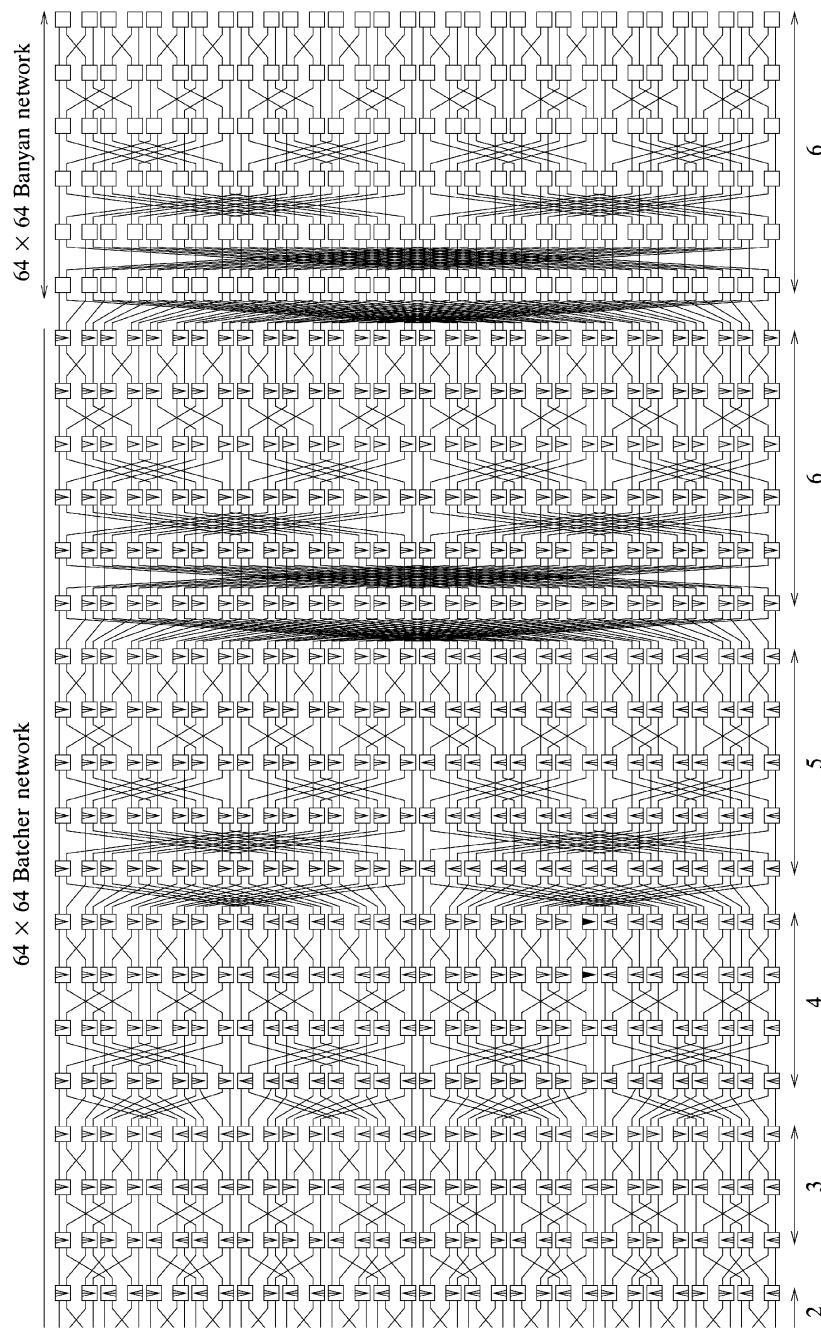
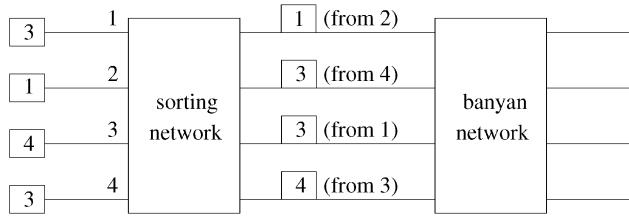
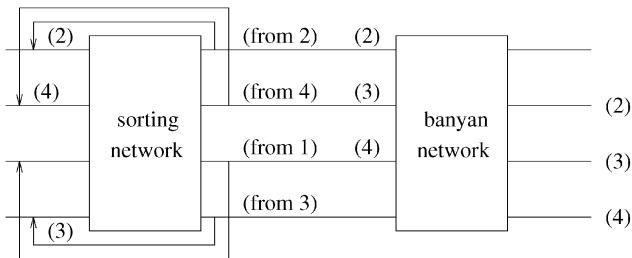


Fig. 5.6 A 64×64 Batcher–banyan switch network.



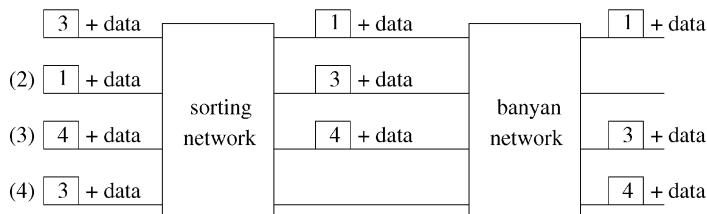
(a) Phase I: Send and resolve request

- Send source-destination pair through sorting network
- Sort destination in non-decreasing order
- Purge adjacent requests with same destination



(b) Phase II: Acknowledge winning ports

- Send ACK with source to ports winning contention
- Route ACK through Batcher-banyan network



(c) Phase III: Send with data

- Acknowledged ports send cells through Batcher-banyan network
- Cells not acknowledged are buffered and retry in the next slot

Fig. 5.7 The three-phase algorithm.

5.3 OUTPUT CONTENTION RESOLUTION ALGORITHMS

5.3.1 Three-Phase Implementation

The following three-phase algorithm is a solution for output contention resolution in a Batcher–banyan switch (see Figure 5.7).

In the first (arbitration) phase of the algorithm, each input port i sends a short request only consisting of a source–destination pair to the sorting network, where the requests are sorted in nondecreasing order according to the destination address. In other words, conflicting requests are sorted at adjacent positions, and a request wins the contention only if its destination is different from the one above it in the sorted list.

As the input ports do not know the result of the arbitration, the requests that won the arbitration must send an acknowledgment to the input ports via an interconnection network in the second phase (the so-called acknowledgment phase). The feedback network in Figure 5.7(b) consists of N fixed connections, each from an output of the Batcher network to the corresponding input of the Batcher network. Each acknowledgment carries the source that has won the contention back to an input of the Batcher network. These acknowledgments (sources) are routed through the entire Batcher–banyan network at distinct outputs according to the source address. When these acknowledgments are feedbacked to the inputs through an identical fixed network, each input port knows if it has won the contention. The input ports that finally receive an acknowledgment are guaranteed output-conflict-free.

These input ports then transmit the full cell in the third and final phase [see Fig. 5.7(c)] through the same Batcher–banyan network. Input ports that fail to receive an acknowledgment retain the cell in a buffer for a retry in the next time slot when the three-phase cycle is repeated.

5.3.2 Ring Reservation

A Batcher–banyan cell switch design with ring reservation is shown in Figure 5.8 [5]. The switch comprises the Batcher–banyan switch fabric, several switch interfaces, and a ring head-end (RHE) and timing generator.

A switch interface supports ring reservation and provides input cell buffering, synchronization of cells sent to the switch fabric, and output cell buffering. Cells entering the switch are buffered in a FIFO until they can participate in the reservation procedure. When an output is successfully reserved, the cell is delivered to the switch fabric at the beginning of the next cell cycle, and the next queued cell can begin to participate in the reservations. When the cell emerges from the output of the switch fabric, it is buffered in the interface before being transmitted to its destination.

The RHE provides two switch synchronization signals (bit clock and cell cycle start) and three ring reservation signals (ring clock, ring data, and ring sync). The ring data signal is a series of output reservation bits, and the ring sync signal indicates the position of the first output port in the ring data

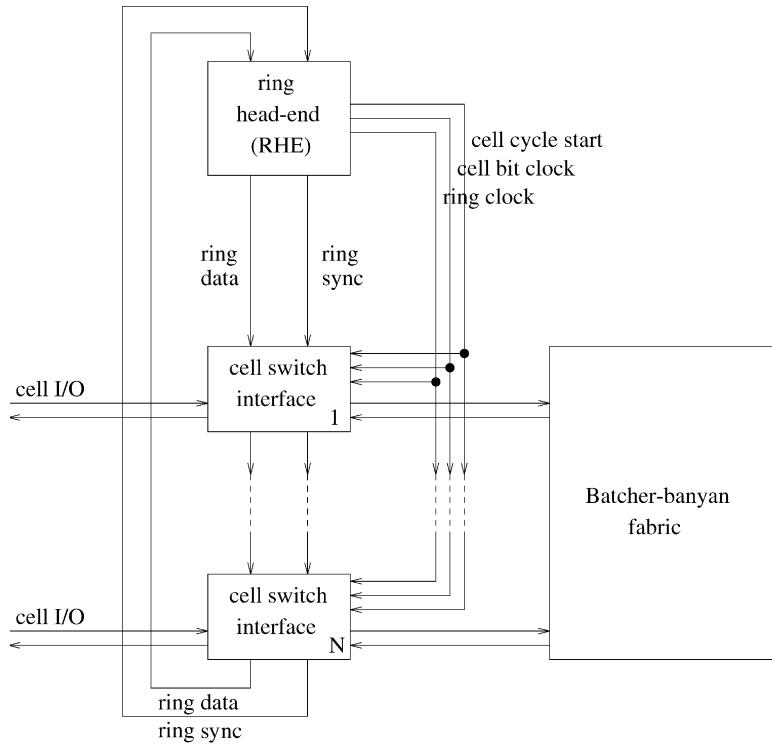


Fig. 5.8 A Batcher–banyan switch with ring reservation.

series. These two signals are circulated through the RHE and switch interfaces, one bit at a time, during the reservation process. Ring reservation is performed at the beginning of each cell cycle after every ring interface has the header of the oldest cell copies. The ring data in the RHE and each ring interface is cleared (IDLE) at every cell cycle start. The ring data series then begins to circulate through the interface bit by bit. Each interface maintains a port counter, which is incremented in every ring data bit time. The port counter is compared with the destination of the oldest cell in every bit time to indicate if the cell is destined for the output in the next bit time. During each ring data bit time, each switch interface examines both the ring sync and the ring data bit. If the ring sync signal is true, which means the next ring data bit corresponds to the first output, then the port counter is reset in the next bit time. If the destination of the cell matches with the port counter and the ring data bit is IDLE, the switch interface writes BUSY on the ring to show that the output has been occupied in the next switch cycle. If the ring data bit is already BUSY, or if the port counter does not match the destination of the oldest cell, the ring data bit is left unchanged. Since each interface makes no more than one reservation per switch cycle, no collisions can take place in the switch fabric. While the ring reservation is being

Input cells	Time slot #1		Time slot #2		Time slot #3		Time slot #4		Time slot #5		Time slot #6		Output cells
	X	Z	X	Z	X	Z	X	Z	X	Z	X	Z	
#0	[] 3	[] 0	[] 1	[] 1	[] 0	[] 2	[] 0	✓ [] 3	[] 0	[] 4	[] 1	[] 5	[] 3
#1	[] 1	[] 0	✓ [] 1	[] 0	[] 2	[] 0	[] 3	[] 0	✓ [] 4	[] 1	[] 5	[] 0	[] 1
#2	[] 5	[] 0	[] 2	[] 0	[] 3	[] 0	[] 4	[] 1	✗ [] 5	[] 0	[] 0	[] 1	[] 1
#3	[] 1	[] 0	[] 3	[] 0	[] 4	[] 1	[] 5	[] 0	[] 0	[] 1	✗ [] 1	[] 0	[] 2
#4	[] 3	[] 0	[] 4	[] 1	[] 5	[] 0	[] 0	[] 1	[] 1	[] 0	[] 2	[] 1	✗ [] 3
#5	[] 5	✓ [] 0	[] 5	[] 0	[] 0	[] 1	[] 1	[] 0	[] 2	[] 1	[] 3	[] 0	[] 4
													[] 5

✓ : allowed to send

✗ : not allowed to send

Fig. 5.9 The implementation of the ring reservation scheme.

performed, the cells reserved in the previous switch cycle are transmitted to the switch fabric.

As shown in Figure 5.9, at the first time slot, the output port addresses of cells from input ports 1 and 5 are matched, and checkmarks are used to indicate that the cells can be sent to these output ports. The token bits x_1 and x_5 are set to one to indicate that output ports 1 and 5 are already reserved. All the token bits are shifted up one bit, and the counter values are also modulo-increased by one for the second time slot. There are no matches found at the second and the third time slots. At the fourth time slot, the output port addresses of cells from input ports 0 and 2 are matched. Since output port 5 is already reserved for the cell in the previous time slot, which is indicated by the value of the token bit x_2 , the cell at input port 2 cannot be sent. Similarly, for the fifth and the sixth time slots, the cells at input ports 3 and 4 cannot be sent to output ports 1 and 3, respectively, since those output ports are already reserved in the previous time slots. At the end, cells from the input ports that are checked are the ones winning the contention.

In this example, since there are six input ports, the arbitration cycle can be completed within six time slots. This scheme uses the serial mechanism and, in general, the arbitration cycle can be done within an N -bit time slot, where N is the number of input/output ports of the switch. This will become the bottleneck when the number of ports of the switch is large. However, by arbitrarily setting the appropriate values for the counters prior to the arbitration, this scheme provides fairness among the input ports. Another advantage of this scheme is that it can be employed at the input of any type of switch fabric.

5.4 THE SUNSHINE SWITCH

The Sunshine switch uses the combination of a Batcher-sorting network and parallel banyan routing networks to provide more than one path to each destination. Figure 5.10 shows a block diagram of the architecture. The k

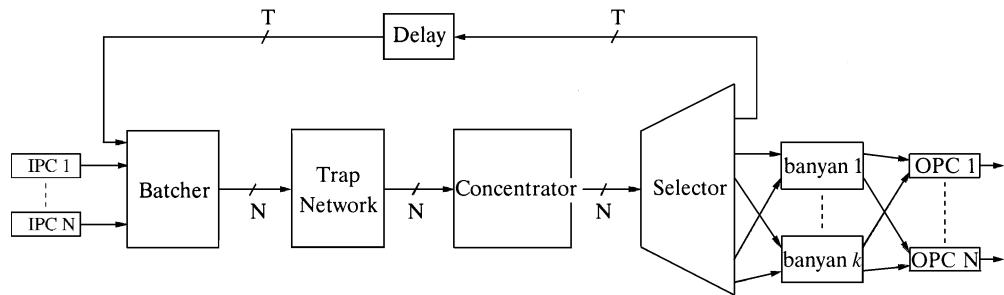


Fig. 5.10 Block diagram of the Sunshine switch.

parallel banyan routing networks provide k independent paths to each output. If more than k cells request a particular output during a time slot, then some excess cells overflow into a shared recirculating queue and then are resubmitted to the switch at dedicated input ports. The recirculating queue consists of T parallel loops and T dedicated inputs to the Batcher-sorting network. Each recirculating loop can hold one cell. A delay block is put within the loops to align recirculated cells with those new arrivals from the input port controllers (IPCs) in the next time slot. During each time slot, the Batcher network sorts newly arrived and recirculated cells in the order of destination address and priority. This enables the trap network to resolve output port contention by selecting the k highest priority cells for each destination address. Since there are k parallel banyan networks, each output can accept k cells in a time slot. When there are more than k cells destined for an output, the excess will enter the recirculating queue. The concentrator and the selector will direct the excess cells to the recirculating loops, while those cells selected for routing will be forwarded to the banyan networks.

Each cell is inserted with a control header at the input port controller. The header format is shown in Figure 5.11. It contains two control fields: a routing field and a priority field; both are ordered starting with the most significant bit. In the routing field, the first bit is a cell activity bit to indicate if the cell contains valid information ($A = 1$), or the cell is empty ($A = 0$). Then the destination address (DA) field identifying the desired output port follows. The priority field consists of the quality of service (QoS) indicator and the internal switch priority (SP). The QoS field distinguishes cells from higher-priority services such as circuit emulation and from lower-priority

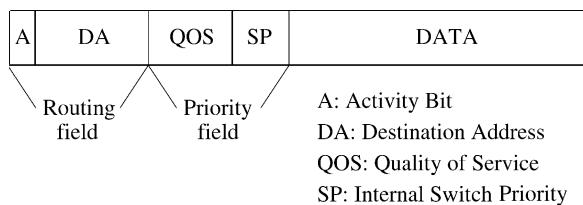


Fig. 5.11 The header format.

services such as connectionless service, and ensures that higher-priority cells will be routed before lower-priority cells when conflicts arise. The SP field is used internally by the switch to indicate the number of time slots that a cell has been delayed, and gives higher priority to recirculated cells. This causes cells from a given source to be routed in sequence.

When the cells are sorted, they are arranged in ascending order of destination address. The priority field, where a higher numerical value represents a higher priority level, appears as an extension of the routing field. This causes cells destined for the same output to be arranged in descending order of priority. In the trap network, the address of every cell is compared with that of the cell k positions above. If a cell has the same address as the cell k positions above, which indicates that there are at least k cells with higher priority, then the cell is marked to be recirculated, and its routing field is interchanged with the priority field because the priority field is more important for the subsequent operation of the concentration sorting network against the recirculation loss. Otherwise, the cell is one of the k (or less) highest-priority cells for the address, and is set to be routed.

In the Batcher concentration network, there are two groups of cells, one to be routed and the other to be recirculated; each is sorted into a contiguous list. Then the group of cells to be routed forms a list in ascending order of the destination address, to avoid subsequent internal blocking in the banyan networks. The group of cells to be recirculated is sorted into a separate contiguous list according to priority. However, the cells to be recirculated are arranged in the order of destination address. If the recirculating queue overflows, the cells destined for the higher-numbered outputs are more likely to be dropped than those for lower-numbered outputs.

Cells are then directed to the selector, which differentiates the two groups and directs them to k banyan networks and to T recirculators, respectively. Cells that enter the recirculators will have their routing and priority fields interchanged back into the original format. Their priority (SP) is incremented as they are recirculated.

The outputs of the selectors are spread among the k banyan networks by connecting every k th output to the same banyan network. This ensures every two cells destined for the same output are separated into different banyan networks. The cells in each banyan network still constitute a contiguous list destined for distinct outputs, which satisfies the nonblocking condition in a banyan network. Every cell then reaches the desired output of a banyan network, and all corresponding outputs are grouped together to an output queue in the output port controller (OPC).

5.5 DEFLECTION ROUTING

5.5.1 Tandem Banyan Switch

Figure 5.12 shows the tandem banyan switching fabric (TBSF) [19]. It consists of multiple banyan networks in series. When two cells contend at any node in

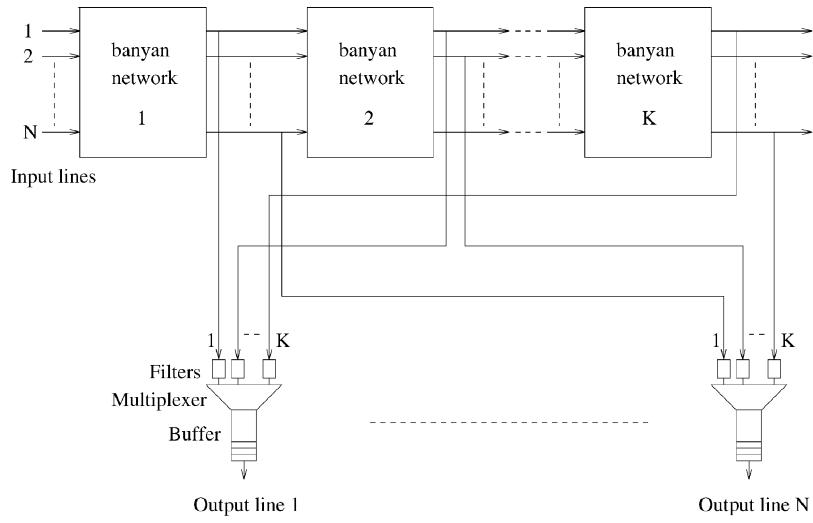


Fig. 5.12 Tandem banyan switching fabric.

a banyan network, one of them will just be deflected to the wrong output of the node and finally arrive at an incorrect destination of the banyan network. The deflected cell is then forwarded to the next banyan network. This process continues again and again until the cell reaches the desired output or it gets out of the last banyan network at an undesired output and is regarded as lost. Each output of every banyan network is connected to the corresponding output buffer. A cell is marked misrouted when it gets deflected in a banyan network, to distinguish it from the properly routed cells and avoid affecting their routing at later stages within the network. At outputs of each banyan network, the cells that have reached their destinations are extracted from the fabric and placed in output port buffers. Note that the load of successive banyan networks decreases and so does the likelihood of conflicts. With a sufficiently large number of banyan networks, it is possible to reduce the cell loss to desired levels. Numerical results show that each additional banyan network improves the cell loss probability by one order of magnitude.

The operation of the TBSF is as follows. A switching header is appended to each cell when it enters the switching fabric, and it comprises the following four fields:

- *Activity Bit a*: It indicates whether the slot contains a cell ($a = 1$) or is empty ($a = 0$).
- *Conflict Bit c*: It indicates whether the cell has already been misrouted at some previous stage of the present network ($c = 1$) or not ($c = 0$).
- *Priority Field P*: It is optional and is used if multiple priority is supported over the switch.
- *Address Field D*: It contains the destination addresses, d_1, d_2, \dots, d_n ($n = \log_2 N$).

The state of a switching element at stage s of a banyan network is primarily determined by three bits in each header of the two input cells; namely, a , c , and d_s . If multiple priority is supported, then the switch state also depends on the priority field P . The algorithm is as follows, where the bits are indexed by 1 and 2 corresponding to the two input cells:

1. If $a_1 = a_2 = 0$, then take no action, i.e., leave the switch in the present state.
2. If $a_1 = 1$ and $a_2 = 0$, then set the switch according to d_{s1} .
3. If $a_1 = 0$ and $a_2 = 1$, then set the switch according to d_{s2} .
4. If $a_1 = a_2 = 1$, then:
 - (a) If $c_1 = c_2 = 1$, then take no action.
 - (b) If $c_1 = 0$ and $c_2 = 1$, then set the switch according to d_{s1} .
 - (c) If $c_1 = 1$ and $c_2 = 0$, then set the switch according to d_{s2} .
 - (d) If $c_1 = c_2 = 0$, then:
 - i. If $P_1 > P_2$, then set the switch according to d_{s1} .
 - ii. If $P_1 < P_2$, then set the switch according to d_{s2} .
 - iii. If $P_1 = P_2$, then set the switch according to either d_{s1} or d_{s2} .
 - iv. If one of the cells has been misrouted, then set its conflict bit to 1.

In order to minimize the number of bits to be buffered at each stage to perform the above algorithm, and thus to minimize the latency incurred at each stage, the address bit is placed in the first position of the address field. This can be done by cyclically shifting the address field by one bit at each stage. It is then possible to keep the latency at each stage as low as 3 bit times without considering multiple-priority support, and to keep it constant over all stages.

It is simple to differentiate successful cells and deflected cells at outputs of each banyan network with the conflict bit: if $c = 0$, then the cell has been properly routed; if $c = 1$, then the cell has been misrouted. A cell with $c = 0$ is accepted by the output buffer, and is rejected by the next banyan network by setting the activity bit in that slot to 0. A cell with $c = 1$ is ignored by the output buffer, but is accepted by the following banyan network with its conflict bit reset to 0 for further routing.

All cells entering the tandem banyan switch fabric in the same time slot must be bit-synchronized throughout the entire fabric. If one ignores the propagation delay, then the delay for each cell in a banyan network is constant and is equal to n times the processing delay at a switching element, which is the time difference between two cells emerging from adjacent banyan networks. In order to have all cells from different banyan networks arrive at an output buffer at the same time, an appropriate delay element can be placed between each output and each banyan network.

In addition, the output buffer memory should have an output bandwidth equal to V bits/s and an input bandwidth equal to KV bits/s to accommodate as many as K cells arriving in the same slot.

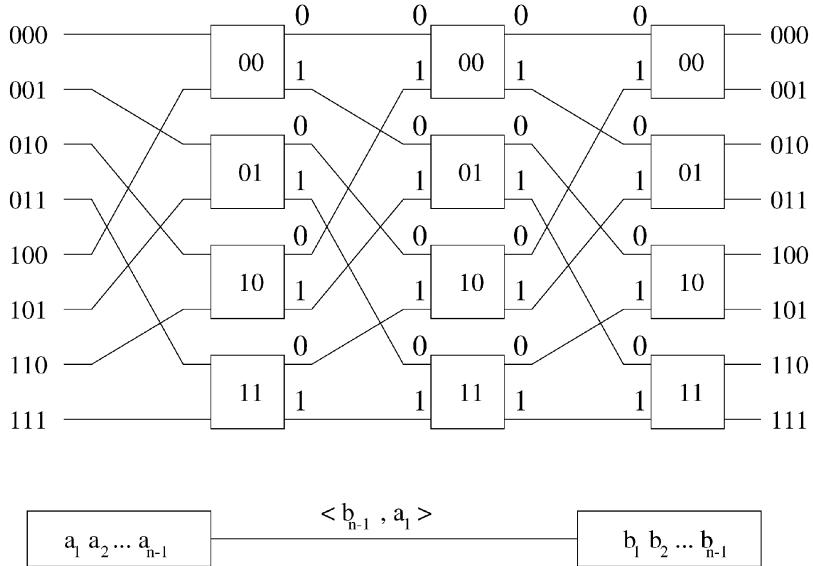


Fig. 5.13 An 8×8 shuffle-exchange network.

5.5.2 Shuffle-Exchange Network with Deflection Routing

Consider an $N \times N$ *shuffle-exchange network* (SN) [15] with $n = \log_2 N$ stages, each consisting of $N/2$ 2×2 switch elements. Figure 5.13 shows an 8×8 SN. Switch nodes in each stage are labeled by an $(n - 1)$ -bit binary number from top to bottom. The upper input (output) of a node is labeled 0, and the lower input (output) is labeled 1. A cell will be forwarded to output 0 (1) at stage i if the i th most significant bit of its destination address is 0 (1). The interconnection between two consecutive stages is called *shuffle exchange*. The output a_n of node $X = (a_1 a_2 \dots a_{n-1})$ is connected to the input a_1 of node $Y = (a_2 a_3 \dots a_n)$ of the subsequent stage. The link between node X and node Y is labeled $\langle a_n, a_1 \rangle$. The path of a cell from input to output is completely determined by the source address $S = s_1 \dots s_n$ and the destination address $D = d_1 \dots d_n$. It can be expressed symbolically as follows:

$$\begin{aligned}
 S &= s_1 \dots s_n \\
 &\xrightarrow{\langle -, s_1 \rangle} (s_2 \dots s_n) && \xrightarrow{\langle d_1, s_2 \rangle} (s_3 \dots s_n d_1) \\
 &\xrightarrow{\langle d_2, s_3 \rangle} \dots && \xrightarrow{\langle d_{i-1}, s_i \rangle} (s_{i+1} \dots s_n d_1 \dots d_{i-1}) \\
 &\xrightarrow{\langle d_i, s_{i+1} \rangle} \dots && \xrightarrow{\langle d_{n-1}, s_n \rangle} (d_1 \dots d_{n-1}) \\
 &\xrightarrow{\langle d_n, 0 \rangle} d_1 \dots d_n = D.
 \end{aligned}$$

The node sequence along the path is embedded in the binary string $s_2 \dots s_n d_1 \dots d_{n-1}$, represented by an $(n - 1)$ -bit window moving one bit per stage from left to right.

The state of a cell traveling in the SN can be represented by a pair (R, X) , where R is its current routing tag and X is the label of the node that the cell resides. At the first stage, the cell is in state $(d_n \dots d_1, s_2 \dots s_n)$. The state transition is determined by the self-routing algorithm as follows:

$$(r_1 \dots r_k, x_1 x_2 \dots x_{n-1}) \xrightarrow[\text{input label } x_n]{\text{exchange}} (r_1 \dots r_{k-1}, x_1 x_2 \dots x_{n-1}) \xrightarrow[\langle r_k, x_1 \rangle]{\text{shuffle}} (r_1 \dots r_{k-1}, x_2 \dots x_{n-1} r_k).$$

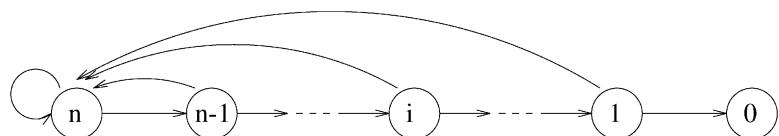
Notice that the routing bit used in the switching node is removed from the routing tag after each stage, before the node label is shuffled to the next stage. Finally, the cell will reach the state $(d_n d_1 \dots d_{n-1})$, from which the following 2×2 element will switch the cell to the destination.

When a contention occurs at a switch node, one of the cells will be successfully routed while the other one will be deflected to the wrong output. As a result, only the nondeflected cells can ever reach their desired outputs. The deflected cells can restart routing (with routing tag reset to $d_n \dots d_1$) again at the deflection point, and if the SN is extended to consist of more than n stages, those deflected cells can reach the destination at later stages. As some cells will reach their destinations after fewer stages than others, a multiplexer is needed to collect cells that reach physical links of the same logical address at different stages. A cell will eventually reach its destination address with good probability provided that the number of stages L is sufficiently large. If it cannot reach the destination after L stages, it is considered lost.

5.5.3 Dual Shuffle-Exchange Network with Error-Correcting Routing

The error-correcting SN is highly inefficient, especially when n is large. This is because routing of the cell must be restarted from the beginning whenever it is deflected. This is illustrated by the state-transition diagram in Figure 5.14(a), where the state is the distance or the number of stages away from destination. A desired network should be one with the state-transition diagram shown in Figure 5.14(b), in which the penalty is only one step backward.

An example is the *dual* SN, which consists of a SN and an *unshuffle-exchange network* (USN). An 8×8 USN is shown in Figure 5.15. It is the mirror image of the SN. Routing in successive stages is based on the least significant bit through the most significant bit. Using a numbering scheme



(a)



(b)

Fig. 5.14 (a) The state-transition diagram of a cell in the shuffle-exchange network, where the distance from the destination is the state. (b) One-step penalty-state transition diagram.

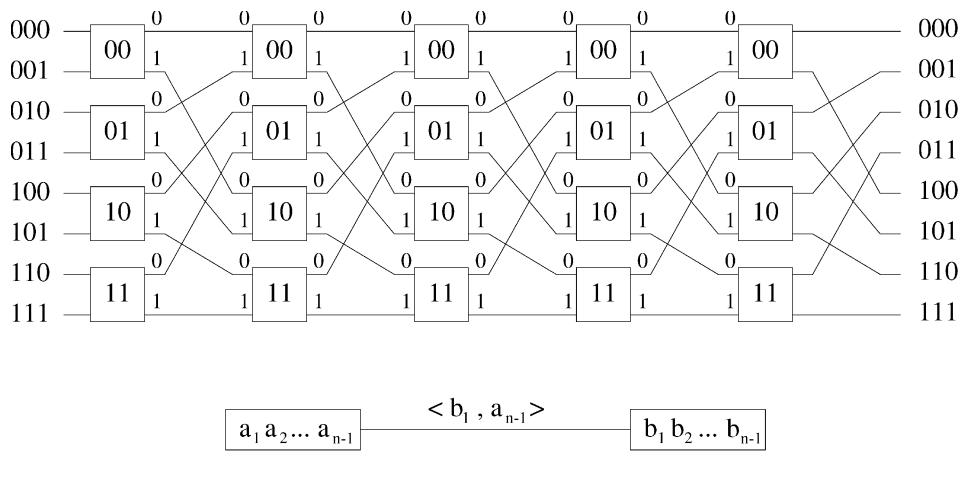


Fig. 5.15 An 8×8 unshuffle-exchange network with five stages.

similar to that in the SN, the path of a cell with source address $S = s_1 \dots s_n$ and destination address $D = d_1 \dots d_n$ can be expressed by

$$\begin{array}{ll}
 S = s_1 \dots s_n & \\
 \xrightarrow{\langle -, s_n \rangle} & (s_1 \dots s_{n-1}) \xrightarrow{\langle d_n, s_{n-1} \rangle} (d_n s_1 \dots s_{n-2}) \\
 \xrightarrow{\langle d_{n-1}, s_{n-2} \rangle} & \dots \xrightarrow{\langle d_{i+2}, s_{i+1} \rangle} (d_{i+2} \dots d_n s_1 \dots s_i) \\
 \xrightarrow{\langle d_{i+1}, s_i \rangle} & \dots \xrightarrow{\langle d_2, s_1 \rangle} (d_2 \dots d_n) \\
 \xrightarrow{\langle d_1, 0 \rangle} & d_1 \dots d_n = D.
 \end{array}$$

An $(n - 1)$ -bit window sliding on the binary string $d_2 \dots d_n s_1 \dots s_{n-1}$ one bit per stage from right to left exactly gives the sequence of nodes along the routing path. The initial state of the cell is $(d_1 \dots d_n, s_1 \dots s_{n-1})$, and the state transition is given by

$$\begin{array}{ccc}
 (r_1 \dots r_k, x_1 x_2 \dots x_{n-1}) & \xrightarrow{\text{exchange}} & (r_1 \dots r_{k-1}, x_1 x_2 \dots x_{n-1}) \\
 \text{input label } x_n & & \text{output label } r_k \\
 & \xrightarrow{\text{unshuffle}} & (r_1 \dots r_{k-1}, r_k x_1 \dots x_{n-2}).
 \end{array}$$

At the last stage, the cell is in state $(-d_1 d_2 \dots d_n)$ and reaches its destination.

Suppose an USN is overlaid on top of a SN, and each node in the USN is combined with its corresponding node in the SN so that a cell at any of the four inputs of the node can access any of the outputs of the node. The shuffle and the unshuffle interconnections between adjacent stages (nodes) compensate each other, so that the error caused by deflection in the SN can be corrected in the USN in only one step. See Figure 5.16. Cell A enters a SN from input 010 to output 101, and cell B , from input 100 to output 100. They collide at the second stage, when they both arrive at node 01 and request output 0. Suppose cell B wins the contention and cell A is deflected to node 11 in the third stage. Imagine cell A is moved to the companion node 11 in the corresponding USN, and is switched to output 0. Then it returns to node 01, the same node (label) where the error occurred, in two stages. At this point, the deflection error has been corrected and cell A can continue on its normal path in the SN. Intuitively, any incorrect routing operation is undone in the SN by a reverse routing operation in the USN.

The above procedure can be formulated more rigorously as follows. Consider a cell in state $(r_1 \dots r_k, x_1 \dots x_{n-1})$. The cell should be sent out on link $\langle r_k, x_1 \rangle$ in the SN. Suppose it is deflected to link $\langle \bar{r}_k, x_1 \rangle$ instead and reaches node $(x_2 \dots x_{n-1} \bar{r}_k)$ in the next stage. The error correction starts by attaching the bit x_1 to the routing tag instead of removing the bit r_k , so that

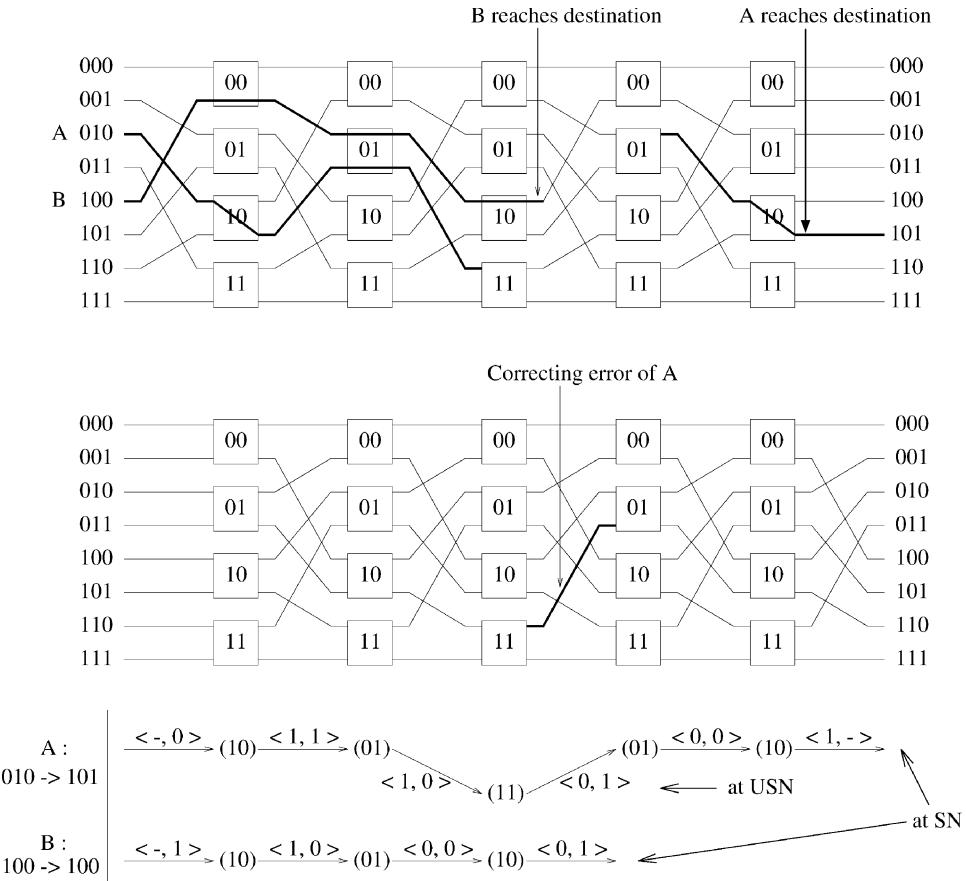
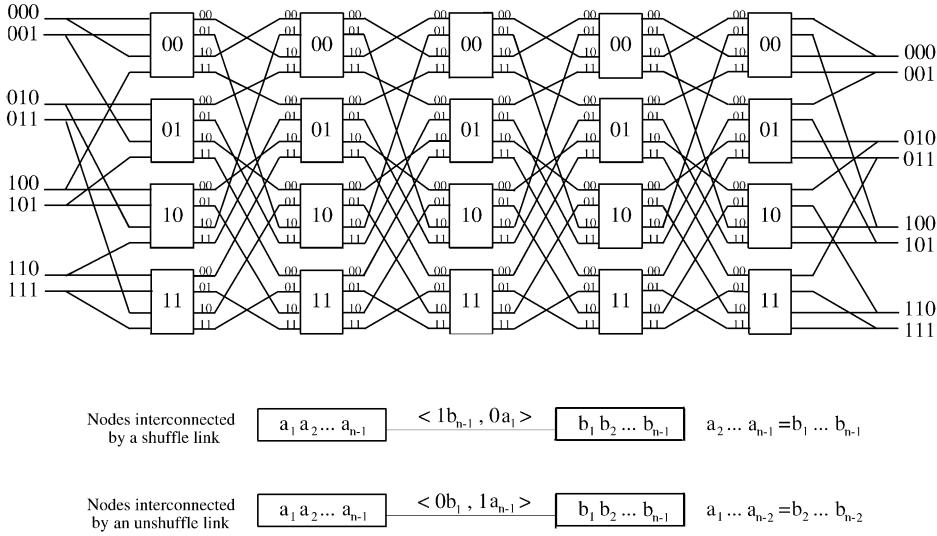


Fig. 5.16 A deflection error in the SN is corrected with the USN.

the state of the cell will be $(r_1 \dots r_k x_1, x_2 \dots x_{n-1} \bar{r}_k)$ in the next stage. Then the cell is moved to the companion node in the USN to correct the error. If the cell is routed successfully this time, it will be sent out on link $\langle x_1, \bar{r}_k \rangle$ and return to the previous state $(r_1 \dots r_k, x_1 \dots x_{n-1})$. Similarly, an error occurring in the USN can also be fixed in one step with the SN. In general, a cell in the SN may also be deflected to a link in the USN and vice versa, and consecutive deflections can occur. A simple algorithm to take these considerations into account is described in the following.

First of all, the companion 2×2 switch elements in the SN and in the USN are merged to form 4×4 switch elements to allow cells to be switched between the SN and the USN. Figure 5.17 shows a dual SN built with 4×4 switch elements. A new labeling scheme is used. The four inputs (outputs) of a switch node are labeled by 00, 01, 10, 11 from top to bottom. Outputs 00 and 01 are connected to the next stage according to an unshuffling pattern, while outputs 10 and 11 are connected to the next stage according to a

**Fig. 5.17** An 8×8 dual shuffle-exchange network.

shuffling pattern. On the other hand, inputs 00 and 01 are connected to the previous stage according to a shuffling pattern, while inputs 10 and 11 are connected to the previous stage according to an unshuffling pattern. A link with label $\langle 1a, 0b \rangle$ is an unshuffle link, and a link with label $\langle 0a, 1b \rangle$ is a shuffle link. Two nodes $(a_1 \dots a_{n-1})$ and $(b_1 \dots b_{n-1})$ are connected by an unshuffle link $\langle 0b_1, 1a_{n-1} \rangle$ if $a_1 \dots a_{n-2} = b_2 \dots b_{n-1}$, and by a shuffle link $\langle 1b_{n-1}, 0a_1 \rangle$ if $a_2 \dots a_{n-1} = b_1 \dots b_{n-2}$.

Since each switch node has four outputs, two routing bits are required to specify the desired output of a cell at each stage. A cell with destination $D = d_1 \dots d_n$ can be routed through either the USN or the SN. Accordingly, the initial routing tag of a cell is set to either $0d_1 \dots 0d_n$ (USN) or $1d_n \dots 1d_1$ (SN).

The state of a cell at any particular time is denoted by $(c_1 r_1 \dots c_k r_k, x_1 \dots x_{n-1})$. There are two possible regular transitions at a switch node; the cell will be sent out on an unshuffle link if $c_k = 0$ and a shuffle link if $c_k = 1$. The corresponding state transitions are given by

$$(c_1 r_1 \dots c_k r_k, x_1 \dots x_{n-1}) \begin{cases} \xrightarrow{\langle 0r_k, 1x_{n-1} \rangle} (c_1 r_1 \dots c_{k-1} r_{k-1}, r_k x_1 \dots x_{n-2}) & \text{if } c_k = 0, \\ \xrightarrow{\langle 1r_k, 0x_1 \rangle} (c_1 r_1 \dots c_{k-1} r_{k-1}, x_2 \dots x_{n-1} r_k) & \text{if } c_k = 1. \end{cases}$$

Without deflections, it is easy to see that a cell with the initial routing set to $0d_1 \dots 0d_n$ ($1d_n \dots 1d_1$) will stay in the USN (SN) links throughout the routing process until it reaches the desired destination at one of the USN (SN) links.

The routing direction is given as follows:

1. If output $c_k r_k$ is available and $k = 1$, the cell has reached its destination; output the cell before the next shuffle if $c = 1$, and after the next unshuffle if $c = 0$.
2. If output $c_k r_k$ is available and $k > 1$, remove the two least-significant bits from the routing tag and send the cell to the next stage.
3. If output $c_k r_k$ is unavailable and $k < n$, choose any other available outputs, attach the corresponding two bits for error correction to the routing tag, and send the cell to the next stage.
4. If output $c_k r_k$ is unavailable and $k = n$, reset the routing tag to its original value, either $0d_1 \dots 0d_n$ or $1d_n \dots 1d_1$; this prevents the length of the routing tag from growing in an unbounded fashion.

Figure 5.18 illustrates the complete error-correcting algorithm. For any node with label $(x_1 \dots x_{n-1})$, the error correcting tag of outputs 00 and 01 is

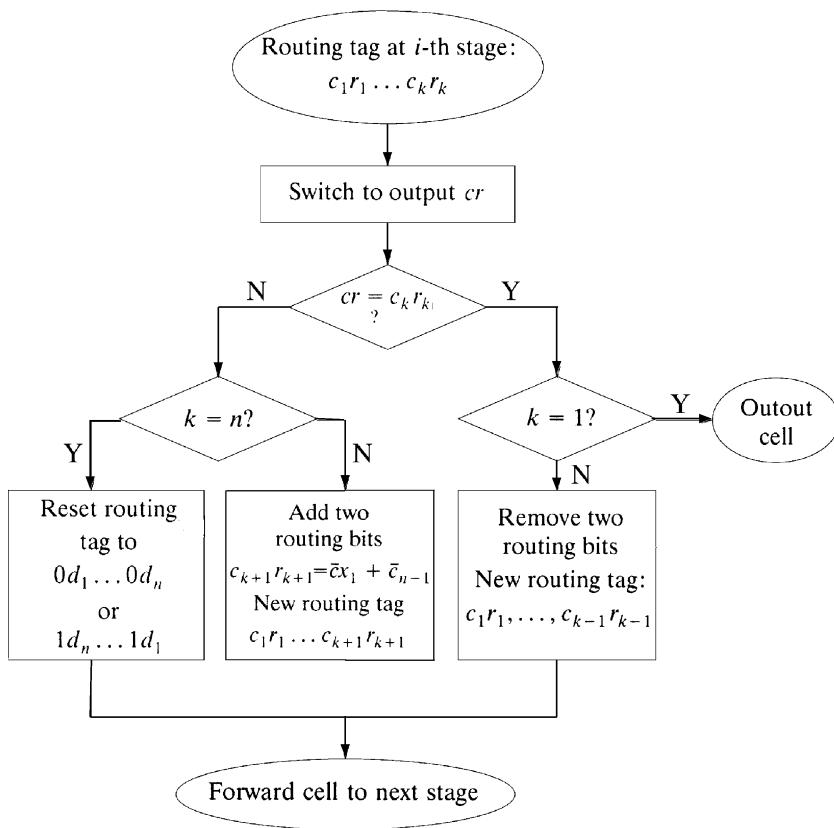
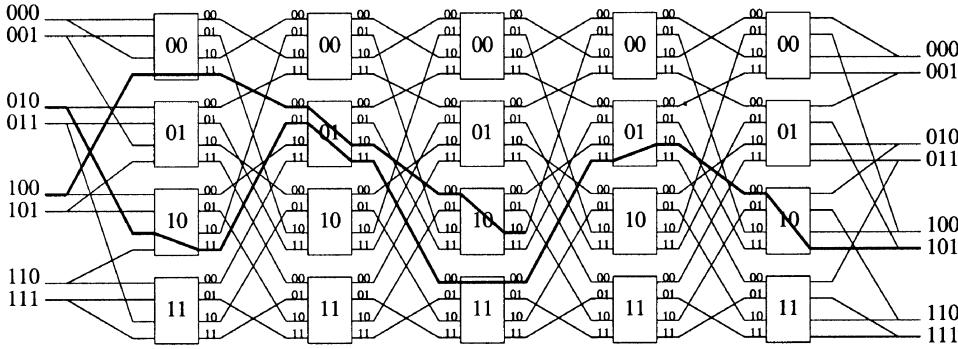


Fig. 5.18 The error-correcting routing algorithm.



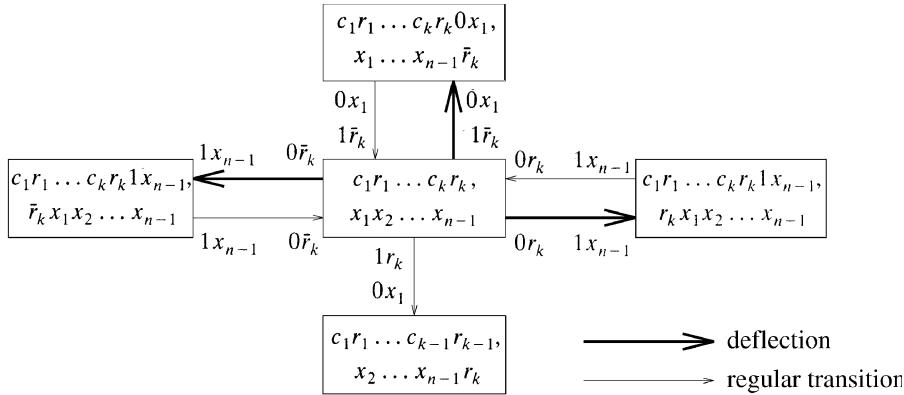
State transition of cells A and B

A: $010 \rightarrow (111011, 10) \rightarrow (1110, 01) \rightarrow (111000, 11) \rightarrow (1110, 01) \rightarrow (11, 10) \rightarrow 101$
B: $100 \rightarrow (101011, 00) \rightarrow (1010, 01) \rightarrow (10, 10) \rightarrow 100$

Fig. 5.19 An example of error-correcting routing in DSN.

$1x_{n-1}$, and the error-correcting tag of outputs 10 and 11 is $0x$. In either case, the error-correcting tag is just the second component $\bar{c}x$ in the link label $\langle cr, \bar{c}x \rangle$, where $x = x_{c+\bar{c}(n-1)}$, which is either x_1 or x_{n-1} according as $c = 1$ (SN) or $c = 0$ (USN). Therefore, a cell deflected to link $\langle cr, \bar{c}x \rangle$ will return to its previous state via link $\langle \bar{c}x, cr \rangle$ in the next stage. This point is illustrated in Figure 5.19 by the same example given in Figure 5.16.

This algorithm can implicitly handle successive deflections as shown by the finite-state machine representation of the algorithm in Figure 5.20. The state

Fig. 5.20 Finite-state machine representation of the error-correcting routing algorithm when $c_k = 1$.

transitions when deflection occurred are given by

$$(c_1r_1 \dots c_kr_k, x_1 \dots x_{n-1}) \begin{cases} \xrightarrow{\langle 0r, 1x_{n-1} \rangle} & (c_1r_1 \dots c_kr_k 1x_{n-1}, rx_1 \dots x_{n-2}) & \text{if } c_kr_k \neq 0r, \\ \xrightarrow{\langle 1r, 0x_1 \rangle} & (c_1r_1 \dots c_kr_k 0x_1, x_2 \dots x_{n-1}r) & \text{if } c_kr_k \neq 1r. \end{cases}$$

5.6 MULTICAST COPY NETWORKS

Figure 5.21 illustrates a serial combination [13] of a copy network and a point-to-point switch for supporting point-to-multipoint communications. The copy network replicates cells from various inputs simultaneously, and then copies of broadcast cells are routed to the final destination by the point-to-point switch.

A copy network consists of the following components and its basic structure is shown in Figure 5.22:

- The *running adder network* (RAN), which generates running sums of the *copy numbers*, specified in the headers of input cells.
- The *dummy address encoder* (DAE), which takes adjacent running sums to form a new header for each cell.
- The *broadcast banyan network* (BBN), which is a banyan network with broadcast switch nodes capable of replicating cells based on two-bit header information.
- The *trunk number translator* (TNT), which determines the outgoing trunk number for each cell copy.

The multicasting mechanism of the copy network lies in the header translations illustrated in Figure 5.23. First, the *numbers of copies* (CNs) specified in the cell headers are added up recursively over the running adder

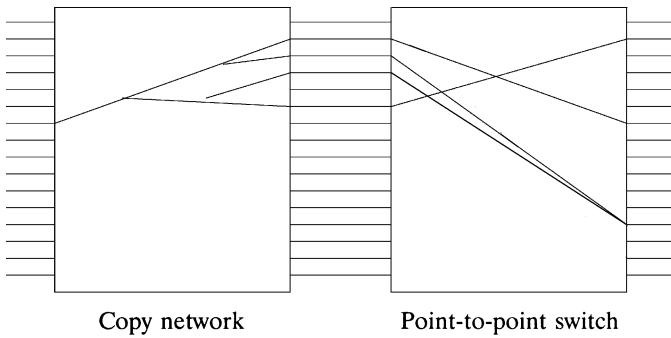


Fig. 5.21 A multicast cell switch consists of a copy network and a point-to-point switch.

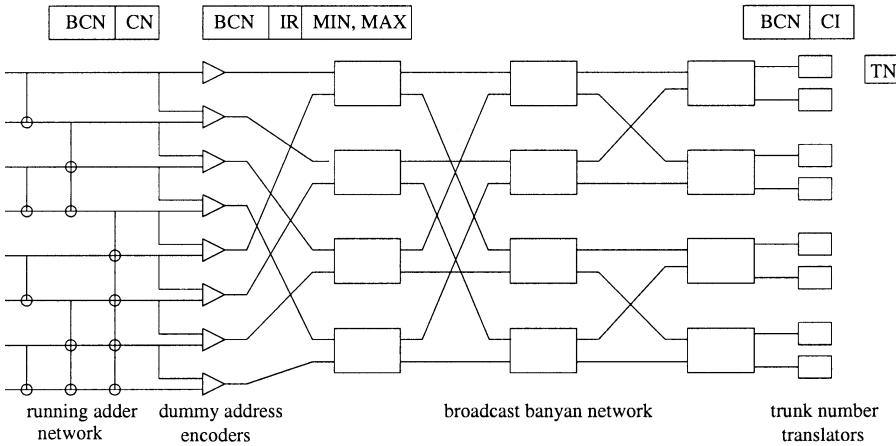


Fig. 5.22 The basic components in a nonblocking copy network.

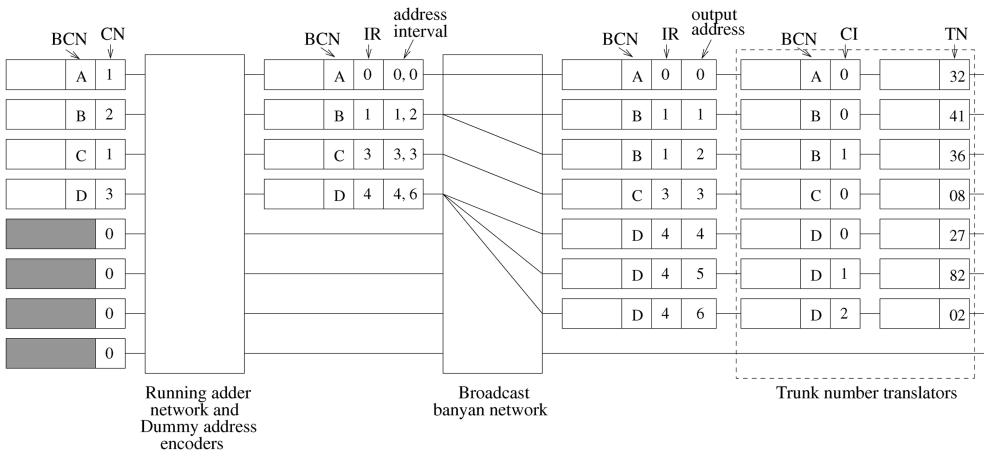


Fig. 5.23 Header translations in the copy network.

network. Based on the resulting sums, the dummy address encoders form new headers with two fields: a *dummy address interval* and an *index reference* (IR). The dummy address interval is formed by the adjacent running sums, namely, the *minimum* (MIN) and the *maximum* (MAX). The index reference is set equal to the minimum of the address interval, and is used later by the trunk number translators to determine the *copy index* (CI). The broadcast banyan network replicates cells according to a *Boolean interval splitting algorithm* based on the address interval in the new header. When a copy finally appears at the desired output, the TNT computes its CI from the output address and the index reference. The *broadcast channel number* (BCN) and the CI form a unique identifier pointing to a *trunk number* (TN), which is added to the cell header and used to route the cell to its final destination.

5.6.1 Broadcast Banyan Network

5.6.1.1 Generalized Self-Routing Algorithm A broadcast banyan network is a banyan network with switch nodes that are capable of replicating cells. A cell arriving at each node can be either routed to one of the output links, or replicated and sent out on both links. There are three possibilities and the uncertainty of making a decision is $\log_2 3 = 1.585$, which means that the minimum header information for a node is 2 bits.

Figure 5.24 illustrates a generalization of the 1-bit self-routing algorithm to the multi-bit case for a set of arbitrary N -bit destination addresses. When a cell arrives at a node in stage k , the cell routing is determined by the k th bits of all destination addresses in the header. If they are all 0 or all 1, then the cell will be sent out on link 0 or link 1, respectively. Otherwise, the cell and its copy are sent out on both links, and the destination addresses in the header are modified correspondingly to the two cell copies: the header of the cell copy sent out on link 0 or link 1 contains those addresses in the original header with the k th bit equal to 0 or 1, respectively.

Several problems may arise in implementing the generalized self-routing algorithm. First, a cell header contains a variable number of addresses and the switch nodes have to read all of them. Second, the cell header modification depends on the entire set of addresses, which is a processing burden on the switch nodes. Finally, the set of paths from any input to a set of outputs

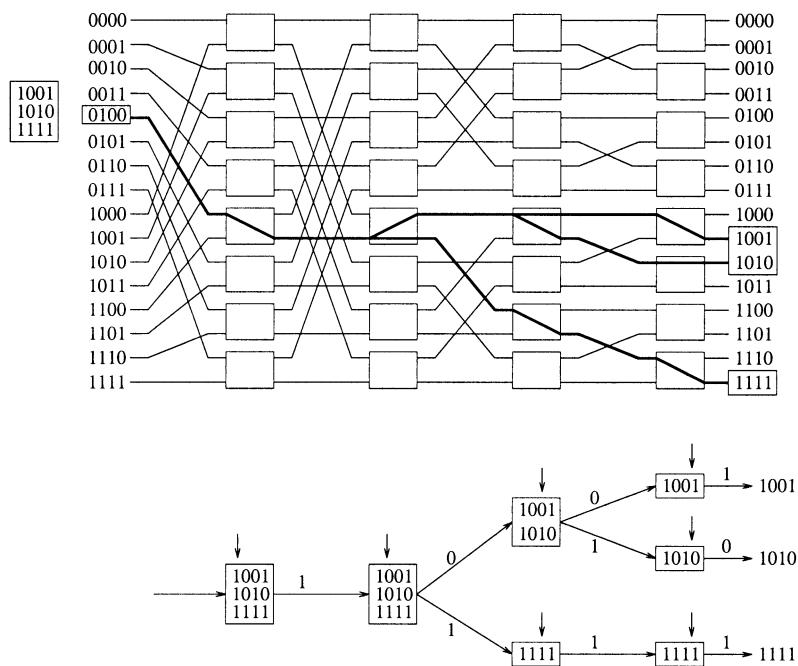


Fig. 5.24 An input–output tree generated by a generalized self-routing algorithm.

form a tree in the network. The trees generated by an arbitrary set of input cells are not link-independent in general, and the network is obviously blocking due to the irregularity of the set of actual destination addresses in the header of each cell. However, fictitious addresses instead of actual addresses can be used in the copy network, where cells are replicated but need not be routed to the actual destinations. The fictitious addresses for each cell then may be arranged to be contiguous so that an address interval consisting of the MIN and the MAX can represent the whole set of fictitious addresses. The address intervals of input cells can be specified to be monotonic to satisfy the nonblocking condition for the broadcast banyan network described below.

5.6.1.2 Boolean Interval Splitting Algorithm An *address interval* is a set of contiguous N -bit binary numbers, which can be represented by two numbers, namely, the *minimum* and the *maximum*. Suppose that a node at stage k receives a cell with the header containing an address interval specified by the two binary numbers $\min(k-1) = m_1 \dots m_N$ and $\max(k-1) = M_1 \dots M_N$, where the argument $k-1$ denotes the stage from which the cell came to stage k . The generalized self-routing algorithm gives the direction for cell routing as follows, and as illustrated in Figure 5.25:

- If $m_k = M_k = 0$ or $m_k = M_k = 1$, then send the cell out on link 0 or 1, respectively.

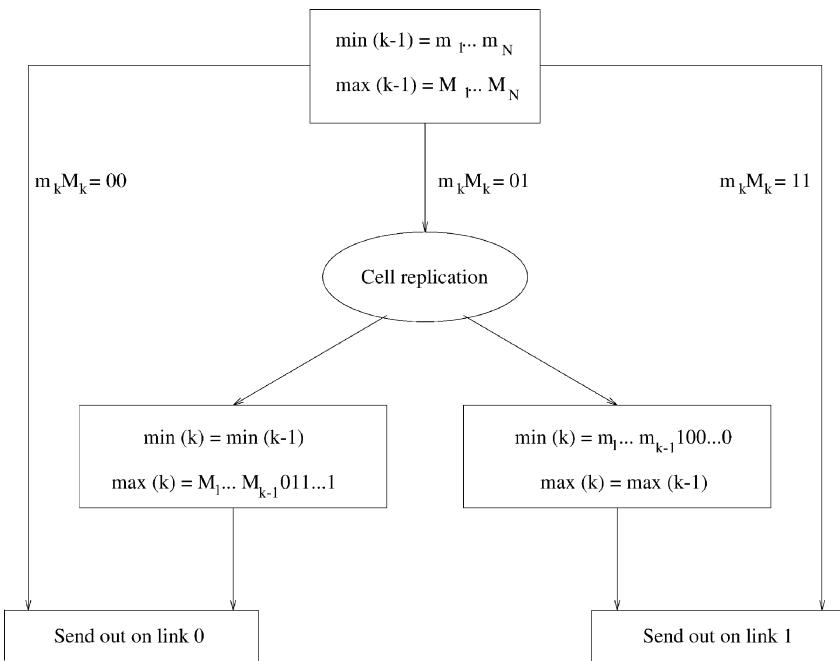


Fig. 5.25 The switch node logic at stage k of a broadcast banyan network.

- If $m_k = 0$ and $M_k = 1$, then replicate the cell, modify the headers of both copies (according to the scheme described below), and send each copy out on the corresponding link.

The modification of a cell header is simply splitting the original address interval into two subintervals, as expressed in the following recursion: For the cell sent out on link 0,

$$\begin{aligned}\min(k) &= \min(k-1) = sm_1 \dots m_N, \\ \max(k) &= M_1 \dots M_{k-1} 01 \dots 1,\end{aligned}$$

and for the cell sent out on link 1,

$$\begin{aligned}\min(k) &= m_1 \dots m_{k-1} 10 \dots 0, \\ \max(k) &= \max(k-1) = M_1 \dots M_N.\end{aligned}$$

Figure 5.26 illustrates the Boolean interval splitting algorithm. From the rules it is realized that $m_i = M_i$, $i = 1, \dots, k-1$, holds for every cell that arrives at stage k . The event $m_k = 1$ and $M_k = 0$ will never occur.

5.6.1.3 Nonblocking Condition of Broadcast Banyan Networks A broadcast banyan network is nonblocking if the active inputs x_1, \dots, x_k and the corresponding sets of outputs Y_1, \dots, Y_k satisfy the following:

1. Monotonicity: $Y_1 < Y_2 < \dots < Y_k$ or $Y_1 > Y_2 > \dots > Y_k$.
2. Concentration: Any input between two active inputs is also active.

The above inequality $Y_i < Y_j$ means that every output address in Y_i is less than any output address in Y_j . Figure 5.27 illustrates a nonblocking example with active inputs $x_1 = 7$, $x_2 = 8$, $x_3 = 9$, and corresponding outputs $Y_1 = \{1, 3\}$, $Y_2 = \{4, 5, 6\}$, $Y_3 = \{7, 8, 10, 13, 14\}$.

5.6.2 Encoding Process

The RAN, together with the DAEs, is used to arrange the destination addresses for each cell so that every eligible cell can be replicated appropriately in the broadcast banyan network without any conflicts. Cell replications in the broadcast banyan network are aided by two processes, an encoding process and a decoding process. The encoding process transforms the set of copy numbers, specified in the headers of incoming cells, into a set of monotone address intervals that form the cell headers in the broadcast banyan network. This process is carried out by a running adder network and a set of dummy address encoders. The decoding process determines the destinations of copies with the TNTs.

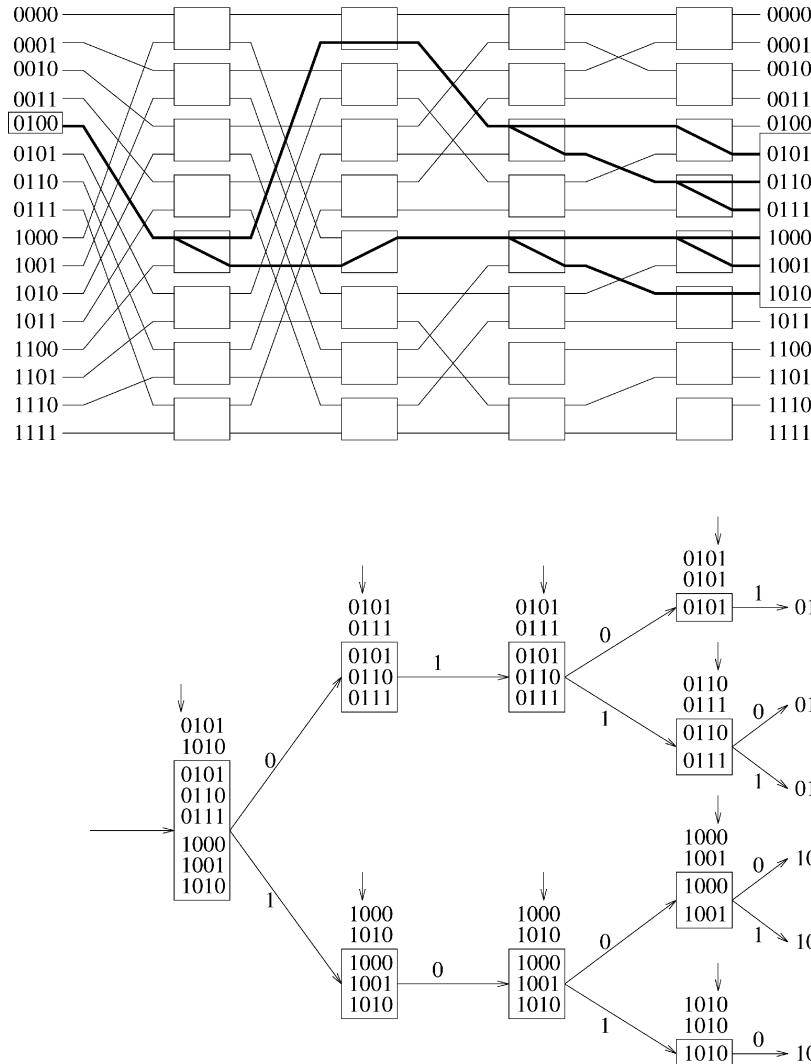


Fig. 5.26 The Boolean interval-splitting algorithm generates the tree while replicating a cell according to the address intervals.

The recursive structure of the $\log_2 N$ -stage running adder network is illustrated in Figure 5.28. The adder network consists of $(N/2)\log_2 N$ adders, each with two inputs and two outputs, where a vertical line denotes a pass. The east output is the sum of the west and the north inputs, while the south output just propagates the north input down. The running sums of CN's are then generated at each port after $\log_2 N$ stages, before the dummy address encoders form the new headers from adjacent running sums. The new header

$$\begin{aligned}
 x_1 &= 7 & Y_1 &= \{1, 3\} \\
 x_2 &= 8 & Y_2 &= \{4, 5, 6\} \\
 x_3 &= 9 & Y_3 &= \{7, 8, 10, 13, 14\}
 \end{aligned}$$

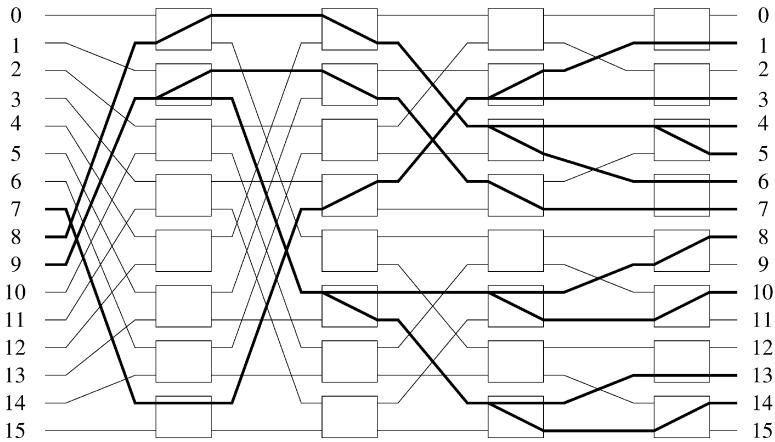


Fig. 5.27 An example to demonstrate the nonblocking condition of a broadcast banyan network.

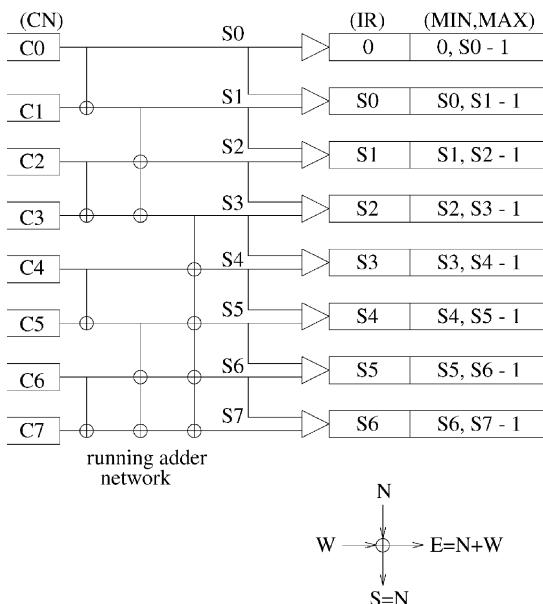


Fig. 5.28 A running adder network and dummy address encoders.

consists of two fields: one is the dummy address interval, represented by two $\log_2 N$ -bit binary numbers (the minimum and the maximum), and the other contains an index reference, which is equal to the minimum of the address interval. Note that the length of each interval is equal to the corresponding copy number in both addressing schemes.

Denoting by S_i the i th running sum, the sequence of dummy address intervals will be generated as follows:

$$(0, S_0 - 1), (S_0, S_1 - 1), \dots, (S_{N-2}, S_{N-1} - 1),$$

where the address is allocated beginning with 0. As shown in the previous section, this sequence satisfies the nonblocking condition over the broadcast banyan network.

5.6.3 Concentration

To satisfy the nonblocking condition of the broadcast banyan network, idle inputs between active inputs must be eliminated. This function should be performed before cells enter the broadcast banyan network, e.g., prior to the RAN or right after the DAE in Figure 5.22. A reverse banyan network is thus used to concentrate active inputs into a contiguous list. As illustrated in Figure 5.29, the routing address in the reverse banyan network is determined by the running sums over activity bits to produce a set of continuous monotonic addresses.

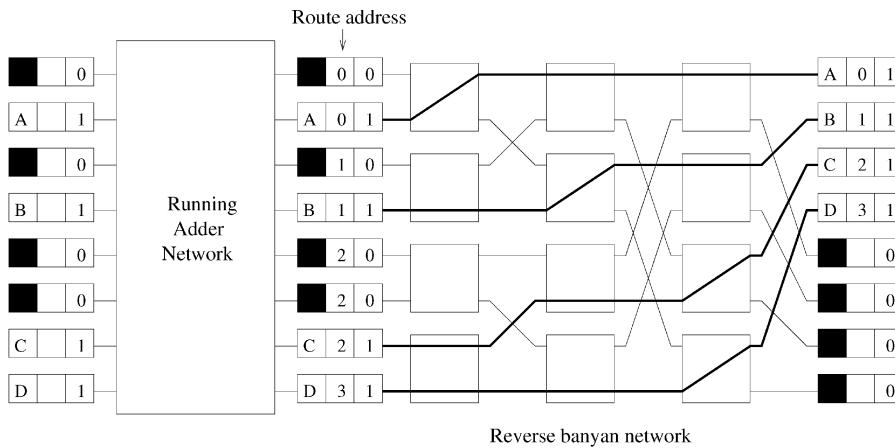


Fig. 5.29 An input concentrator consists of a running adder network and a reverse banyan network.

5.6.4 Decoding Process

When a cell emerges from the broadcast banyan network, the address interval in its header contains only one address, that is, according to the Boolean interval-splitting algorithm,

$$\min(\log_2 N) = \max(\log_2 N) = \text{output address}.$$

The cell copies belonging to the same broadcast channel should be distinguished by the CI, which is determined at the output of the broadcast banyan network (see Fig. 5.30) by,

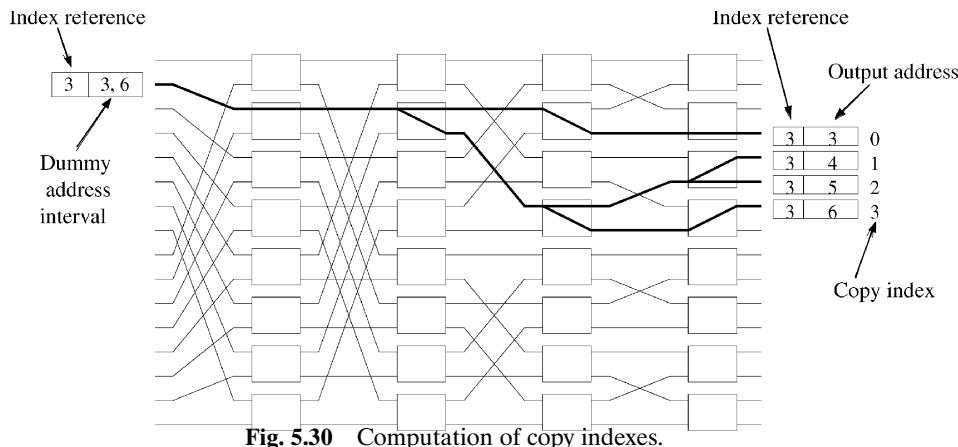
$$\text{CI} = \text{output address} - \text{index reference}.$$

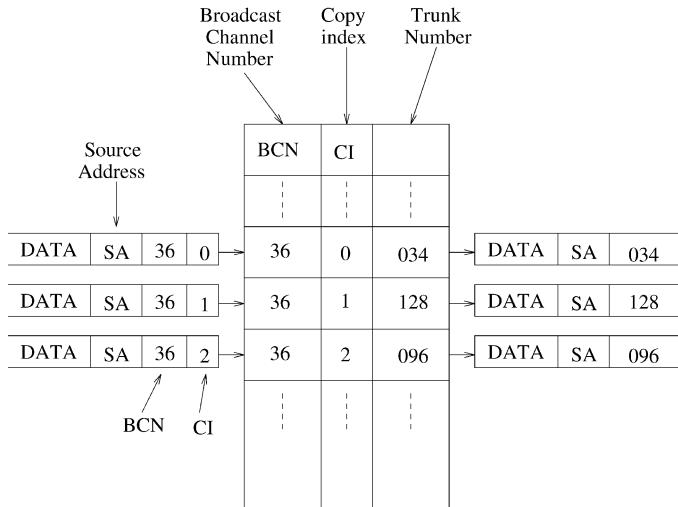
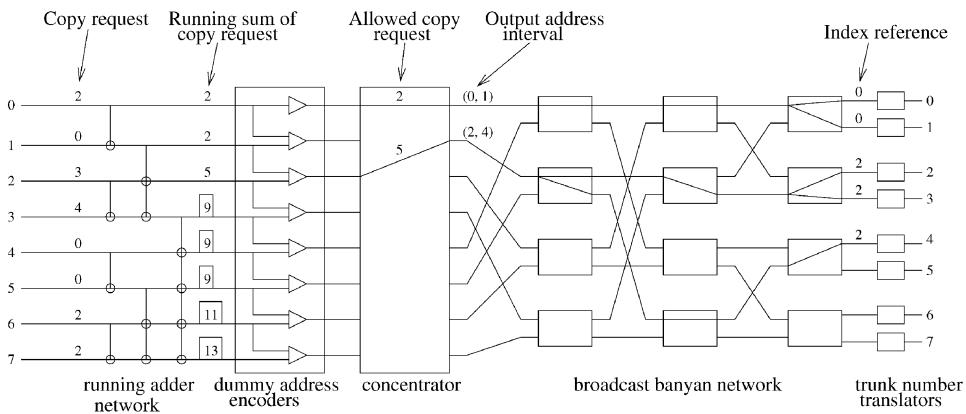
Recall that the index reference is initially set equal to the minimum of the address interval.

A TNT is used to assign the actual address to each cell copy so that it will be routed to its final destination in the succeeding point-to-point switch. TN assignment can be accomplished by a simple table lookup in which the identifier (searching key) consists of the BCN and the CI associated with each cell. When a TNT receives a cell copy, it first converts the output address and IR into the CI, and then replaces the BCN and CI with the corresponding TN in the translation table. The translation process is illustrated in Figure 5.31.

5.6.5 Overflow and Call Splitting

Overflow will occur in the RAN of the copy network when the total number of copy requests exceeds the capacity of the copy network. If partial service (also called *call splitting*) is not allowed in cell replication and a cell must



**Fig. 5.31** Trunk number translation by table lookup.**Fig. 5.32** An 8×8 nonblocking copy network without call splitting: Only five instead of eight cell copies are allowed in this time slot.

generate all its copies in a time slot, then the throughput may be degraded when overflow occurs. As illustrated in Figure 5.32, overflow occurs at port 3, and only five cell copies are allowed, although more than eight requests are available.

5.6.6 Overflow and Input Fairness

Overflow will also introduce unfairness among incoming cells, because the starting point of the RAN is fixed. Since the calculation of the running sum always starts from input port 0 in every time slot, lower-numbered input ports have higher service priorities than higher numbered ports.

This unfairness problem will be solved if the RAN is redesigned to calculate the running sums cyclically starting from any input port, and the starting point of computing the running sums in every time slot is determined adaptively by the overflow condition in the previous time slot. Such a *cyclic RAN* (CRAN) is illustrated in Figure 5.33. The current starting point is port 3, call splitting is performed at port 6, and the new starting point in the next slot will be port 6. The negative index reference -3 , provided by the DAE, implies that the copy request from port 3 is a residual one, and three copies have been generated in the previous time slot.

5.6.6.1 A. Cyclic Running Adder Network Figure 5.34 shows the structure of an 8×8 CRAN. The associated cell header format consists of three fields: starting indicator (SI), running sum (RS), and routing address (RA). Only one port, which is the starting point, has a nonzero SI initially. The RS field is initially set to the number of copies requested by the input cell. The RA field is initially set to 1 if the port is active; otherwise it is set to 0. At the output of the RAN, the RA field will carry the running sum over activity bits, to be used as the routing address in the following concentrator.

A set of cyclic passing paths is implemented in each stage of the CRAN, so that recursive computation of running sums can be done cyclically. In order to emulate the actual running sum computation from a starting point, however, some passing paths should be virtually cut, as illustrated in Figure 5.34, which is equivalent to having the shaded nodes ignore their links while computing the running sums. These nodes are preceded by a cell header with the SI field equal to 1, as it is propagated from the starting point over the CRAN. The header modification in a node is summarized in Figure 5.35.

The next starting point will remain the same unless overflow occurs, in which case the first port facing the overflow will be the next starting point. If

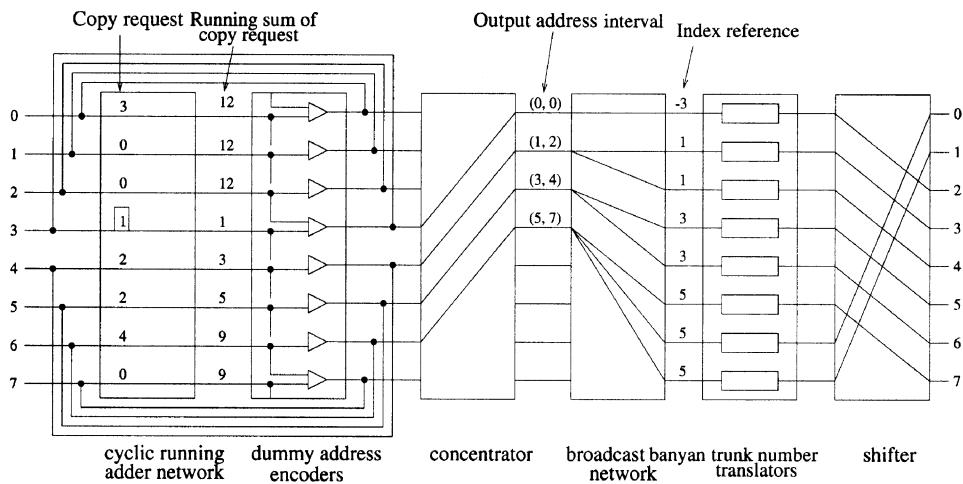


Fig. 5.33 A CRAN in an 8×8 copy network.

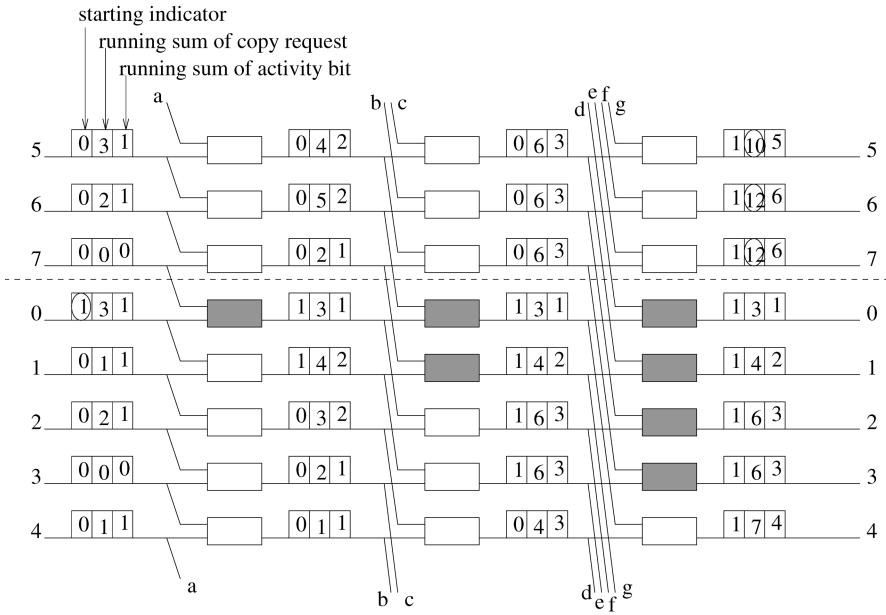


Fig. 5.34 An 8×8 cyclic running adder network.

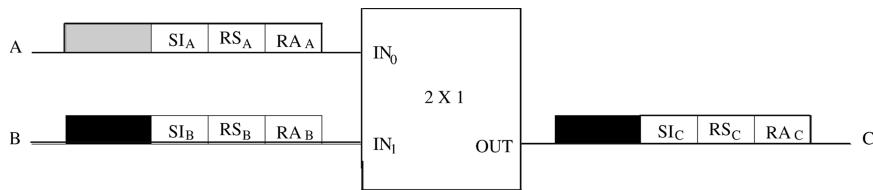


Fig. 5.35 The operation of a node in a CRAN.

we denote the starting point as port 0, and number the other ports from 1 to $N - 1$ in a cyclic manner, then the SI bit that indicates the next starting point is updated with adjacent RS fields at each port as follows:

$$SI_0 = \begin{cases} 1 & \text{if } RS_{N-1} \leq N, \\ 0 & \text{otherwise.} \end{cases}$$

and

$$SI_i = \begin{cases} 1 & \text{if } RS_{i-1} \leq N \text{ and } RS_i > N, \\ 0 & \text{otherwise,} \end{cases}$$

where $i = 1, 2, \dots, N - 1$.

In order to support call splitting, every input port should know how many copies are served in each time slot. This piece of information is called the starting copy number (SCN). A set of feedback loops is then established to send this information back to input ports after it is determined with adjacent

running sums as follows:

$$SCN_0 = RS_0,$$

and

$$SCN_i = \begin{cases} \min(N - RS_{i-1}, RS_i - RS_{i-1}) & \text{if } RS_{i-1} < N \\ 0 & \text{otherwise} \end{cases}$$

5.6.6.2 Concentration The starting point in a CRAN may not be port 0, and the resulting sequence of routing addresses in the RBN may not be continuous monotone. As illustrated in Figure 5.36, internal collisions may occur in the RBN.

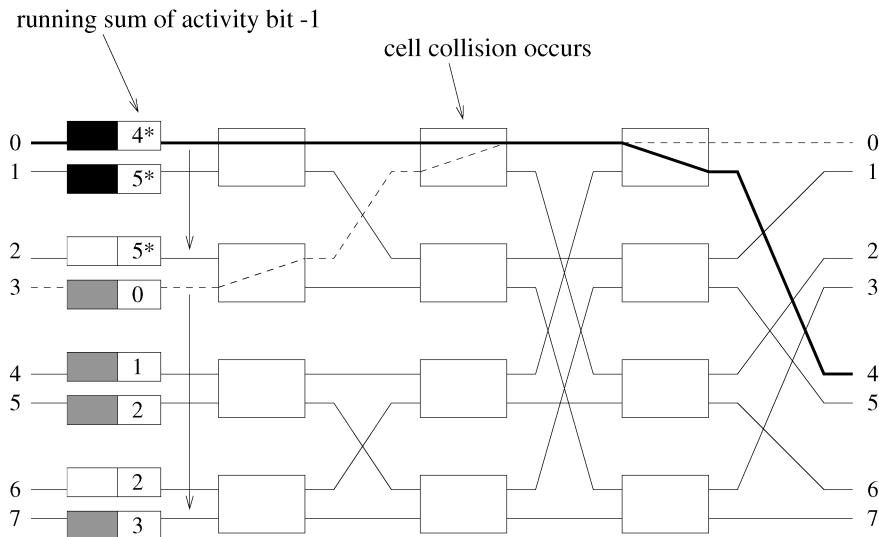


Fig. 5.36 Cyclic monotone addresses give rise to cell collisions in a reverse banyan network. Port 2 and port 6 are idle.

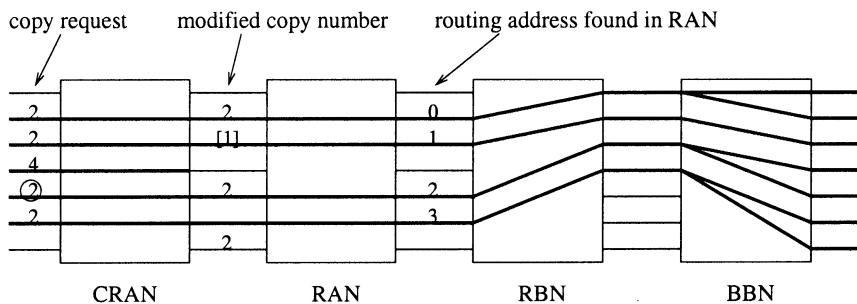


Fig. 5.37 An additional RAN is used to concentrate active cells. The starting point is marked by encircling its copy request.

This problem can be solved if an additional RAN with fixed starting point (port 0) is added in front of the RBN. As shown in Figure 5.37, this additional RAN will recalculate the running sums of RAs so that the resulting sequence of RAs becomes continuous monotone.

REFERENCES

1. H. Ahmadi and W. E. Denzel, "A survey of modern high-performance switching techniques," *IEEE J. Select. Areas Commun.*, vol. 7, no. 7, pp. 1091–1103, Sep. 1989.
2. P. Baran, "On distributed communications networks," *IEEE Trans. Commun.*, vol. 12, pp. 1–9, 1964.
3. K. E. Batcher, "Sorting networks and their application," *Proc. Spring Joint Comput. Conf.*, AFIPS, pp. 307–314, 1968.
4. V. E. Benes, "Optimal rearrangeable multistage connecting networks," *Bell Syst. Tech. J.*, vol. 43, pp. 1641–1656, Jul. 1964.
5. B. Bingham and H. Bussey, "Reservation-based contention resolution mechanism for Batcher–banyan packet switches," *Electron. Lett.*, vol. 24, no. 13, pp. 772–773, Jun. 1988.
6. J. W. Byun, "The design and analysis of an ATM multicast switch with adaptive traffic controller," *IEEE/ACM Trans. Networking*, vol. 2, no. 3, pp. 288–298, Jun. 1994.
7. C. Clos, "A study of non-blocking switching network," *Bell Syst. Tech. J.*, vol. 32, pp. 404–426, Mar. 1953.
8. J. N. Giacopelli, W. D. Sincoskie, and M. Littlewood, "Sunshine: A high performance self routing broadband packet switch architecture," *Proc. Int. Switching Symp.*, Jun. 1990.
9. L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," *Proc. 1st Annu. Int. Symp. Comput. Architecture*, pp. 21–28, Dec. 1973.
10. A. Huang and S. Knauer, "Starlite: A wideband digital switch," *Proc. IEEE Globecom '84*, pp. 121–125, Dec. 1984.
11. Y. N. J. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Select. Areas Commun.*, vol. 5, no. 8, pp. 1264–1273, Oct. 1987.
12. J. J. Kulzer and W. A. Montgomery, "Statistical switching architecture for future services," *Proc. ISS '84*, Florence, Italy, pp. 22A.4.1–4.6, May 1984.
13. T. T. Lee, "Nonblocking copy networks for multicast packet switching," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1455–1467, Dec. 1988.
14. T. T. Lee and S. C. Liew, "Broadband packet switches based on dilated interconnection networks," *IEEE Trans. Commun.*, vol. 42, Feb. 1994.
15. S. C. Liew and T. T. Lee, "NlogN dual shuffle-exchange network with error-correcting routing," *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 754–766, Feb./Mar./Apr. 1994.
16. S. C. Liew and T. T. Lee, "Principles of broadband switching and networking (Draft 3)," Chinese Hong Kong University, 1995.

17. R. J. McMillan, "A survey of interconnection networks," *Proc. IEEE Globecom '84*, pp. 105–113, Dec. 1984.
18. F. A. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks," *Proc. IEEE*, vol. 78, no. 1, pp. 133–167, Jan. 1990.
19. F. A. Tobagi and T. Kwok, "The tandem banyan switching fabric: a simple high-performance fast packet switch," *Proc. IEEE Infocom '91*, pp. 1245–1253, 1991.
20. J. S. Turner and L. F. Wyatt, "A packet network architecture for integrated services," *Proc. IEEE Globecom '83*, pp. 2.1.1–2.1.6, Nov. 1983.
21. J. S. Turner, "Design of a broadcast packet switching network," *Proc. IEEE Infocom '86*, pp. 667–675, 1986.
22. "Design of an integrated service packet network," *IEEE J. Select. Areas Commun.*, Nov. 1986.
23. J. S. Turner, "New directions in communications (or which way to the information age?),"*IEEE Trans. Commun. Mag.*, vol. 24, pp. 8–15, Oct. 1986.
24. J. S. Turner, "A practical version of Lee's multicast switch architecture," *IEEE Trans. Commun.*, vol. 41, no. 8, pp. 1166–1169, Aug. 1993.

CHAPTER 6

KNOCKOUT-BASED SWITCHES

As shown in Chapter 2, output buffer switches (including the shared-memory switches) provide the best delay–throughput performance. The problem of the output-buffered switches is that their capacity is limited by the memory speed. Consider the case of an ATM switch with 100 ports. We should ask ourselves what is the probability of all 100 cells arriving at the same output port in the same time slot. If the probability is very low, why do we need to have the output buffer able to receive all 100 cells at the same slot? A group of researchers in Bell Labs in the late 1980s tried to solve this problem by limiting the number of cells that can arrive at an output port in each time slot and thus the speed requirement of the memory at the output ports. Excess cells are discarded by the switch fabric. The concept is called the *knockout principle*. The question is how many cells should be delivered to the output port in each time slot. If it is too many, memory speed may be the bottleneck. If too few, the cell loss rate in the switch fabric may be too high to be acceptable. For a given cell loss rate, this number can be determined. The number is found to be 12 for a cell loss rate of 10^{-10} , independent of the switch size.

This result seems very encouraging in that the memory speed is no longer the bottleneck for the output-buffered switch. However, there are no commercial switches implemented with the knockout principle. This is because the results obtained assume that input traffic distributions from different inputs are uncorrelated, which may be unrealistic in the real world. Secondly, people are not comfortable with the idea that cells are discarded by the switch fabric. Usually, cells are discarded when a buffer is filled or exceeds some predetermined threshold.

Although the knockout principle has not been used in real switching systems, it has attracted many researchers in the past, and various architectures based on it have been proposed. Some of them are discussed in this chapter. Section 6.1 describes the knockout principle and an implementation and architecture of a knockout switch. Section 6.2 describes a useful and powerful concept, channel grouping, to save routing links in the switch fabric. A generalized knockout principle that extends the original one by integrating the channel grouping concept is described. Section 6.3 describes a two-stage multicast output-buffered switch that is based on the generalized knockout principle. Section 6.4 describes a fault-tolerant multicast output-buffered ATM switch. Section 6.5 is an appendix that shows the derivation of some equations used in this chapter.

6.1 SINGLE-STAGE KNOCKOUT SWITCH

6.1.1 Basic Architecture

The knockout switch [28] is illustrated in Figure 6.1. It is composed of a completely broadcasting interconnection fabric and N bus interfaces. The interconnection fabric for the knockout switch has two basic characteristics: (1) each input has a separate broadcast bus, and (2) each output has access to all broadcast buses and thus all input cells.

With each input having a direct path to every output, no switch blocking occurs within the interconnection fabric. The only congestion in the switch takes place at the interface to each output, where cells can arrive simultaneously on different inputs destined for the same output. The switch architecture is modular in that the N broadcast buses can reside on an equipment backplane with the circuitry for each of the N input–output pairs placed on a single plug-in circuit card.

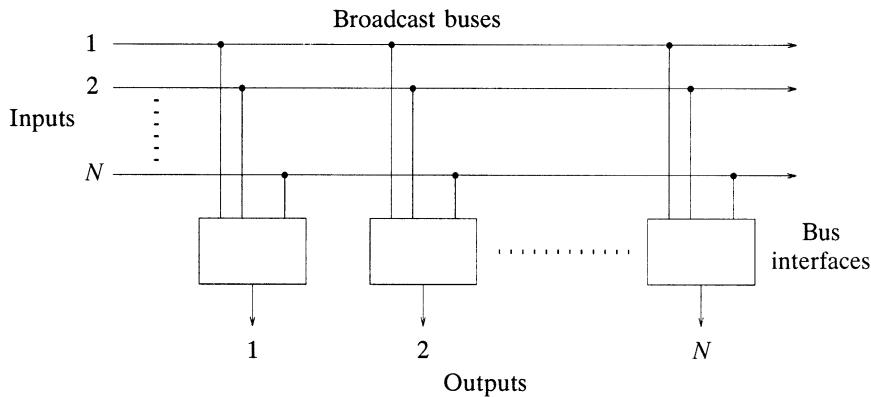


Fig. 6.1 Knockout switch interconnection fabric.

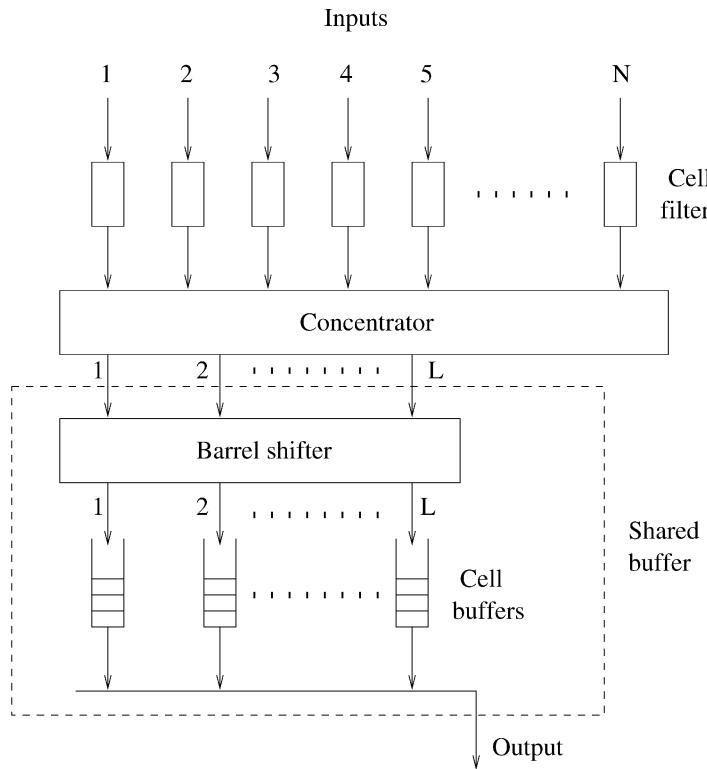


Fig. 6.2 Knockout switch bus interface. (©1987 IEEE.)

Figure 6.2 illustrates the architecture of the bus interface associated with each output of the switch. The bus interface has three major components. At the top there are a row of N cell filters where the address of every cell is examined, with cells addressed to the output allowed to pass on to the concentrator and all others blocked. The concentrator then achieves an N -to- L ($L \ll N$) concentration of the input lines, and up to L cells in each time slot will emerge at outputs of the concentrator. These L concentrator outputs then enter a shared buffer, which is composed of a barrel shifter and L separate FIFO buffers. The shared buffer allows complete sharing of the L FIFO buffers and provides the equivalent of a single queue with L inputs and one output, each operated under a FIFO queuing discipline. The operation of the barrel shifter is shown in Figure 6.3. At time T , cells A , B , C arrive and are stored in the top three FIFO buffers. At time $T + 1$, cells D to J arrive and begin to be stored from the fourth FIFO in a round-robin manner. The number of positions that the barrel shifter shifts is equal to the sum of arriving cells mod L .

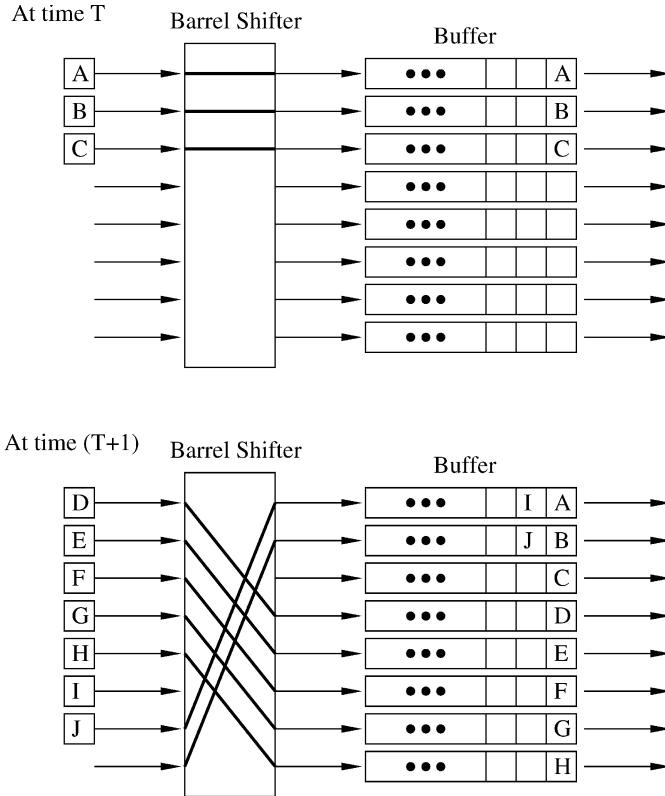


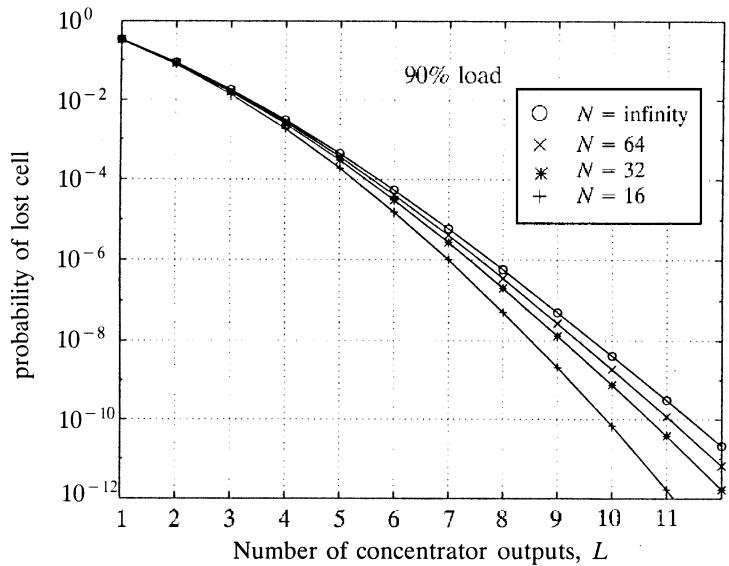
Fig. 6.3 Operation of a barrel shifter.

6.1.2 Knockout Concentration Principle

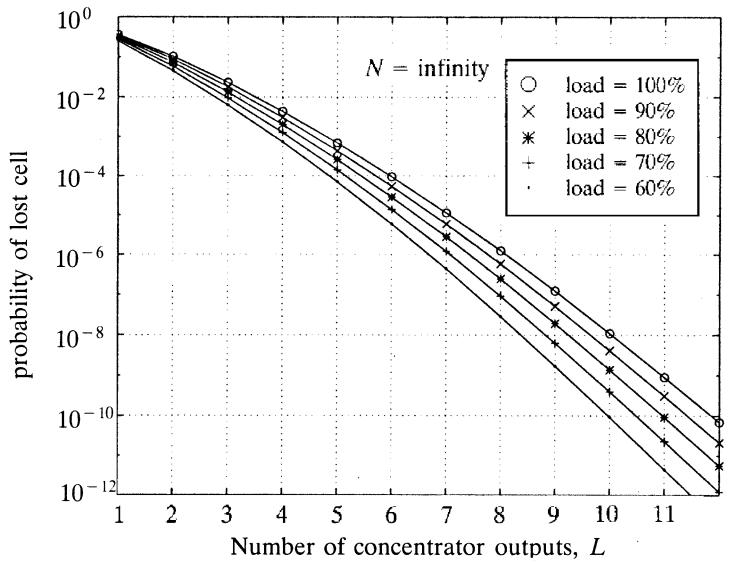
All cells passing through the cell filters enter the concentrator, with an N -to- L concentration. If there are $k \leq L$ cells arriving in a time slot for a given output, these k cells will emerge from the concentrator on outputs 1 to k after getting out of the concentrator. If $k > L$, then all L outputs of the concentrator will have cells, and $k - L$ cells will be dropped (i.e., lost) within the concentrator.

The cell loss probability is evaluated as follows. It is assumed that, in every time slot, there is a fixed and independent probability ρ that a cell arrives at an input. Every cell is equally likely destined for each output. Denote by P_k the probability of k cells arriving in a time slot all destined for the same output, which is binomially distributed as follows:

$$P_k = \binom{N}{k} \left(\frac{\rho}{N} \right)^k \left(1 - \frac{\rho}{N} \right)^{N-k}, \quad k = 0, 1, \dots, N. \quad (6.1)$$



(a) with various switch sizes



(b) with various loads

Fig. 6.4 Concentrator cell loss performance.

It then follows that the probability of a cell being dropped in a concentrator with N inputs and L outputs is given by

$$\Pr[\text{cell loss}] = \frac{1}{\rho} \sum_{k=L+1}^N (k-L) \binom{N}{k} \cdot \left(\frac{\rho}{N}\right)^k \left(1 - \frac{\rho}{N}\right)^{N-k}. \quad (6.2)$$

Taking the limit as $N \rightarrow \infty$, and with some manipulations,

$$\Pr[\text{cell loss}] = \left(1 - \frac{L}{\rho}\right) \left(1 - \sum_{k=0}^L \frac{\rho^k e^{-\rho}}{k!}\right) + \frac{\rho^L e^{-\rho}}{L!} \quad (6.3)$$

Using (6.2) and (6.3), Figure 6.4(a) shows a plot of the cell loss probability versus L , the number of outputs on the concentrator, for $\rho = 0.9$ and $N = 16, 32, 64, \infty$. Note that a concentrator with only eight outputs achieves a cell loss probability less than 10^{-6} for arbitrarily large N . This is comparable to the probability of losing a 500-bit cell from transmission errors with a bit error rate of 10^{-9} . Also note from Figure 6.4(a) that each additional output added to the concentrator beyond eight results in an order of magnitude decrease in the cell loss probability. Hence, independent of the number of inputs (N), a concentrator with 12 outputs will have a cell loss probability less than 10^{-10} . Figure 6.4(b) illustrates, for $N \rightarrow \infty$, that the required number of concentrator outputs is not particularly sensitive to the load on the switch, up to and including a load of 100%. It is also important to note that, assuming independent cell arrivals on each input, the simple, homogeneous model used in the analysis corresponds to the worst case, making the cell loss probability performance results shown in Figure 6.4 upper bounds on any set of heterogeneous arrival statistics [10].

6.1.3 Construction of the Concentrator

The basic building block of the concentrator is a simple 2×2 contention switch shown in Figure 6.5(a). The two inputs contend for the *winner* output according to their activity bits. If only one input has an arriving cell (indicated by an activity bit = 1), it is routed to the winner (left) output. If both inputs have arriving cells, one input is routed to the winner output and the other input to the loser output. If both inputs have no arriving cells, we do not care, except that the activity bit for both should remain at logic 0 at the switch outputs.

The above requirements are met by a switch with the two states shown in Figure 6.5(b). The switch examines the activity bit of the left input only. If the activity bit is a 1, the left input is routed to the winner output and the right input is routed to the loser output. If the activity bit is a 0, the right input is routed to the winner output, and no path is provided through the switch for the left input. Such a switch can be realized with as few as 16 gates, and

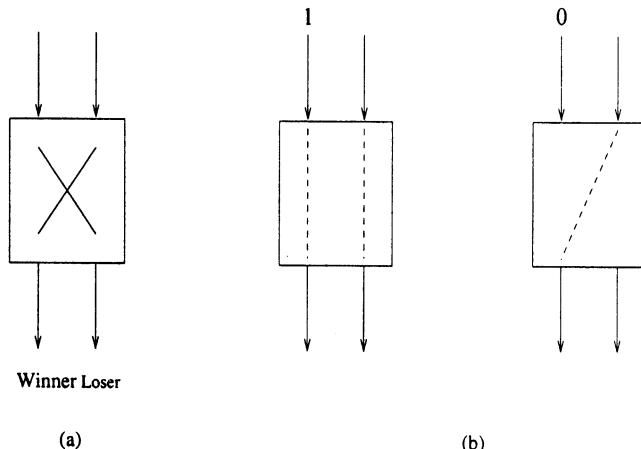


Fig. 6.5 (a) The 2×2 contention switch. (b) States of the 2×2 contention switch.

having a latency of at most one bit. Note that priority is given to the cell on the left input to the 2×2 switch element. To avoid this, the switch element can be designed so that it alternates between selecting the left and right inputs as winners when there is a cell arriving on both inputs in the same time slot. However, suppose the priority structure of the 2×2 switch element were maintained and (as described below) the concentrator were designed so that one input, say the N th, always received the lowest priority for exiting a concentrator output. The cell loss probability for this worst case input, as $N \rightarrow \infty$, is given by

$$\Pr[\text{cell loss for the worst case input}] = 1 - \sum_{k=0}^{L-1} \frac{\rho^k e^{-\rho}}{k!}. \quad (6.4)$$

The above equation is obtained by considering there are k cells destined to the same output port from the first $N - 1$ inputs, where

$$P_k = \binom{N-1}{k} \left(\frac{\rho}{N-1} \right)^k \left(1 - \frac{\rho}{N-1} \right)^{N-1-k}, \quad k = 0, 1, \dots, N-1. \quad (6.5)$$

As $N \rightarrow \infty$, $P_k = (\rho^k e^{-\rho})/k!$. Cells at the N th input will be transmitted to the output if the number of cells from the first $N - 1$ inputs destined for the same output port is less than or equal to $L - 1$. The entire summation in (6.4) is the probability that the cell from the N th input will not be lost. Comparing the results of (6.4) with the cell loss probability averaged over all inputs, as given by (6.3) and shown in Figure 6.4(b), it is found that the worst case cell loss probability is about a factor of 10 greater than the average. This greater cell loss probability, however, can be easily compensated for by adding an additional output to the concentrator.

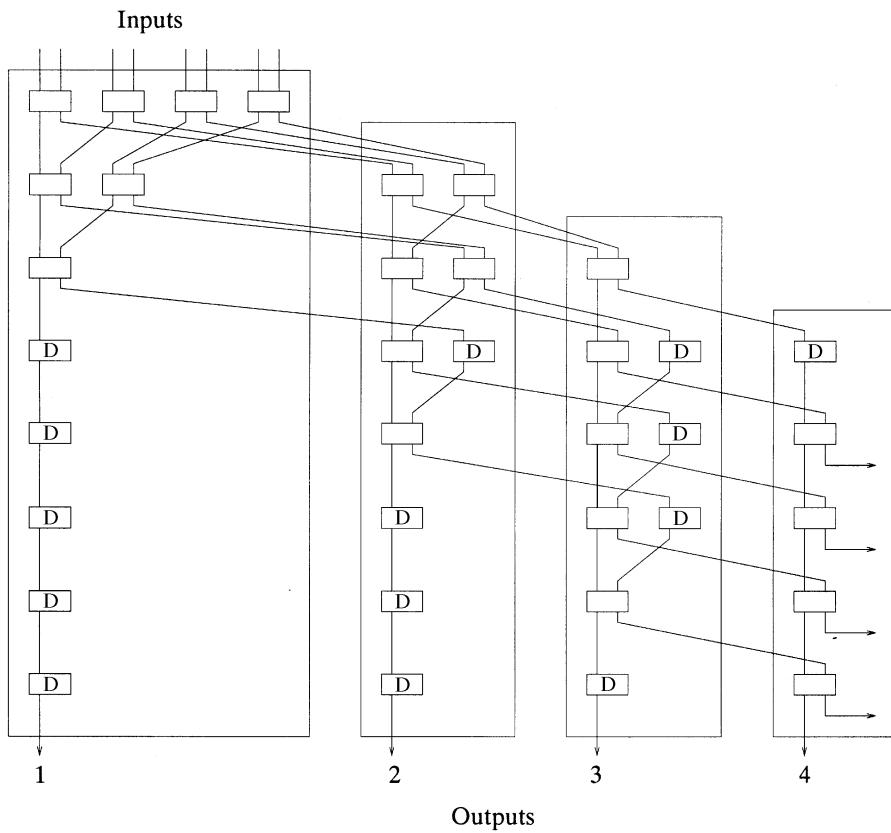


Fig. 6.6 An eight-input to four-output concentrator. (©1987 IEEE.)

Figure 6.6 shows the design of an eight-input four-output concentrator composed of these simple 2×2 switch elements and single-input, single-output 1-bit delay elements (marked *D*). At the input to the concentrator (upper left side of Figure 6.6), the N outputs from the cell filters are paired and enter a row of $N/2$ switch elements. One may view this first stage of switching as the first round of a tournament with N players, where the winner of each match emerges from the left side of the 2×2 switch element and the loser emerges from the right side. The $N/2$ winners from the first round advance to the second round, where they compete in pairs as before, using a row of $N/4$ switch elements. The winners in the second round advance to the third round, and this continues until two compete for the championship: that is, the right to exit the first output of the concentrator. Note that if there is at least one cell arriving on an input to the concentrator, a cell will exit the first output of the concentrator.

A tournament with only a single tree-structured competition leading to a single winner is sometimes referred to as a single-knockout tournament: lose

one match and you are knocked out of the tournament. In a double-knockout tournament, the $N - 1$ losers from the first section of competition compete in a second section, which produces a second-place finisher (i.e., a second output for the concentrator) and $N - 2$ losers. As Figure 6.6 illustrates, the losers from the first section can begin competing in the second section before the competition is finished in the first. Whenever there are an odd number of players in a round, one player must wait and compete in a later round in the section. In the concentrator, a simple delay element serves this function.

For a concentrator with N inputs and L outputs, there are L sections of competition, one for each output. A cell entering the concentrator is given L opportunities to exit through a concentrator output. In other words, a cell losing L times is knocked out of the competition and is discarded by the concentrator. In all cases, however, some cells are lost only if more than L cells arrive in any one time slot. As we have seen, for $L \geq 8$, this is a low-probability event.

For $N \gg L$, each section of the concentrator contains approximately N switch elements for a total concentrator complexity of $16NL$ gates. For $N = 32$ and $L = 8$, this corresponds to a relatively modest 4000 gates. Once a concentrator microcircuit is fabricated, Figure 6.7 illustrates how several identical chips can be interconnected to form a larger concentrator. The loss probability performance of the two-stage concentrator is the same as for the single-stage concentrator. In general, a K^JL -input, L -output concentrator can be formed by interconnecting $J + 1$ rows of KL -to- L concentrator chips in a treelike structure, with the i th row (counting from the bottom) containing K^{i-1} chips. For the example illustrated in Figure 6.7, $L = 8$, $K = 4$, and $J = 2$.

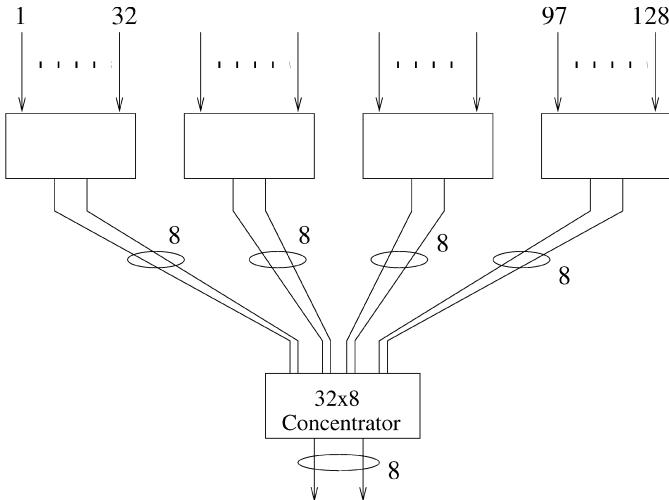


Fig. 6.7 The 128-to-8 concentrator constructed from 32-to-8 concentrator chips.

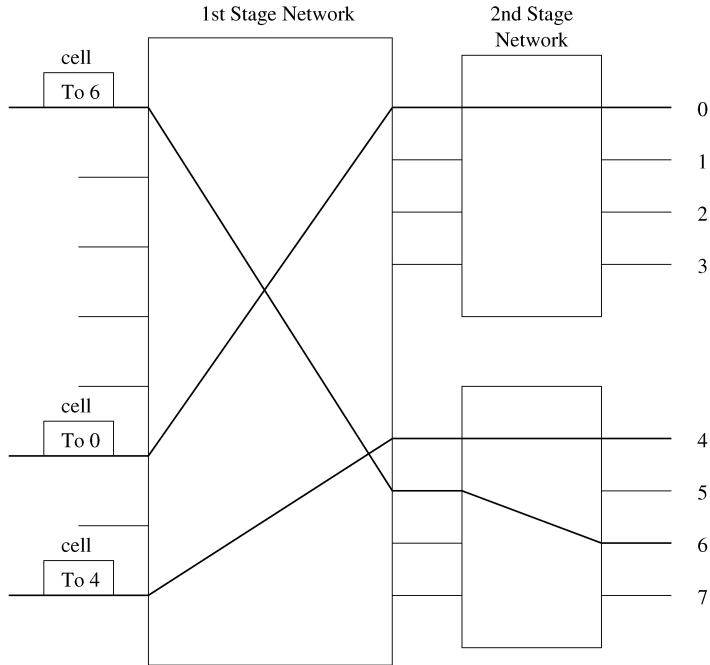


Fig. 6.8 Illustration of channel grouping principle.

6.2 CHANNEL GROUPING PRINCIPLE

The construction of a two-stage modular network is mostly based on the channel grouping principle [24] to separate the second stage from the first stage. With a group of outputs treated identically in the first stage, a cell destined for an output of this group can be routed to any output of the group before being forwarded to the desired output in the second stage. For instance, as shown in Figure 6.8, a cell at the top input destined for output 6 appears at the second input of the second group, while another input cell destined for output 0 appears at the first input of the first group. The first-stage network routes cells to proper output groups, and the second-stage network further routes cells to proper output ports. This smooths the problem of output contentions and thus achieves better performance-complexity tradeoff for the first stage switch. More theoretical evaluations are provided as follows.

6.2.1 Maximum Throughput

This subsection focuses on the switch structure shown in Figure 6.9. An output group consists of M output ports and corresponds to an output address for the first-stage network. A cell can access any of the M corresponding output ports of the first-stage network. In any given time slot, at

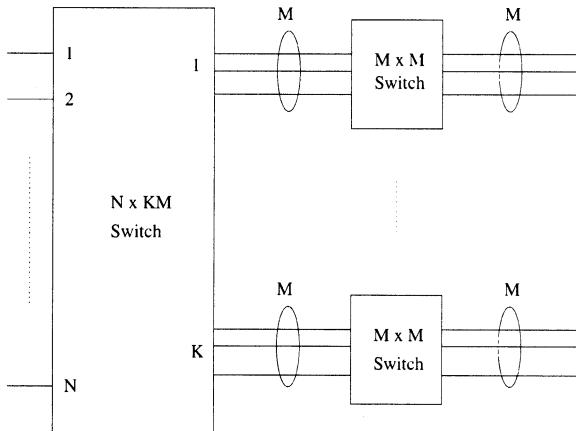


Fig. 6.9 An asymmetric switch with line expansion ratio KM/N .

most M cells can be cleared from a particular output group, one cell on each output port.

The maximum throughput of an input-buffered switch is limited by head-of-line blocking. The symmetric case is evaluated in [8], and the maximum throughput is 0.586. A similar approach could be taken for the asymmetric case to a point where the solution could be found by numerical analysis [20].

Table 6.1 lists the maximum throughput per input for various values of M and K/N [20]. The column in which $K/N = 1$ corresponds to the special cases studied in [8] and [23]. For a given M , the maximum throughput increases with K/N because the load on each output group decreases with increasing K/N . For a given K/N , the maximum throughput increases with M because each output group has more output ports for clearing cells.

Table 6.2 lists the maximum throughput as a function of the line expansion ratio (the ratio of the number of output ports to the number of input ports), KM/N . Notice that for a given line expansion ratio, the maximum throughput increases with M . Channel grouping has a stronger effect on throughput for smaller KM/N than for larger KM/N . This is because for large KM/N , and

TABLE 6.1 Maximum Throughput with K/N Kept Constant While $K, N \rightarrow \infty$

M	Maximum Throughput								
	$K/N = \frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	4	8	16
1	0.061	0.117	0.219	0.382	0.586	0.764	0.877	0.938	0.969
2	0.121	0.233	0.426	0.686	0.885	0.966	0.991	0.998	0.999
4	0.241	0.457	0.768	0.959	0.996	1.000	1.000	1.000	
8	0.476	0.831	0.991	1.000	1.000				
16	0.878	0.999	1.000						

TABLE 6.2 Maximum Throughput with KM/N Kept Constant While $KM, N \rightarrow \infty$

M	KM/N =	Maximum Throughput					
		1	2	4	8	16	32
1		0.586	0.764	0.877	0.938	0.969	0.984
2		0.686	0.885	0.966	0.991	0.998	0.999
4		0.768	0.959	0.996	1.000	1.000	1.000
8		0.831	0.991	1.000			
16		0.878	0.999				
32		0.912	1.000				
64		0.937					
128		0.955					
256		0.968					
512		0.978					
1024		0.984					

$M = 1$, the line expansion has already alleviated much of the throughput limitation due to head-of-line blocking.

6.2.2 Generalized Knockout Principle

This section generalizes the knockout concentration loss calculation to a group of outputs [7, 3]. Consider an $N \times N$ switch with two-stage routing networks, as shown in Figure 6.10. A group of M outputs at the second stage share LM routing links from the first-stage network. The probability that an input cell is destined for this group of outputs is simply $M\rho/N$. If only up to LM cells are allowed to pass through to the group of outputs, where L is

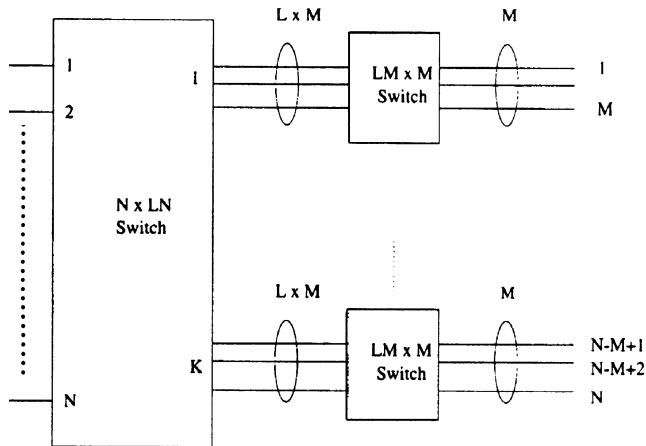


Fig. 6.10 An $N \times N$ switch with group expansion ratio L .

called group expansion ratio, then

$$\Pr[\text{cell loss}] = \frac{1}{M\rho} \sum_{k=LM+1}^N (k-LM) \binom{N}{k} \left(\frac{M\rho}{N}\right)^k \cdot \left(1 - \frac{M\rho}{N}\right)^{N-k}. \quad (6.6)$$

As $N \rightarrow \infty$,

$$\Pr[\text{cell loss}] = \left(1 - \frac{L}{\rho}\right) \left(1 - \sum_{k=0}^{LM} \frac{(M\rho)^k e^{-M\rho}}{k!}\right) + \frac{(M\rho)^{LM} e^{-M\rho}}{(LM)!}. \quad (6.7)$$

The derivation of the above equation can be found in the appendix of this chapter (Sec. 6.5).

As an example, we set $M = 16$ and plot in Figure 6.11 the cell loss probability (6.7) as a function of LM under various loads. Note that $LM = 33$ is large enough to keep the cell loss probability below 10^{-6} for a 90% load. In contrast, if the group outputs had been treated individually, the value of LM would have been 128 (8×16) for the same cell loss performance. The advantage from grouping outputs is shown in Figure 6.12 as the group expansion ratio L vs. a practical range of M under different cell loss criteria. For a cell loss probability of 10^{-8} , note that L decreases rapidly from 8 to less than 2.5 for group sizes M larger than 16; a similar trend is evident for other cell loss probabilities.

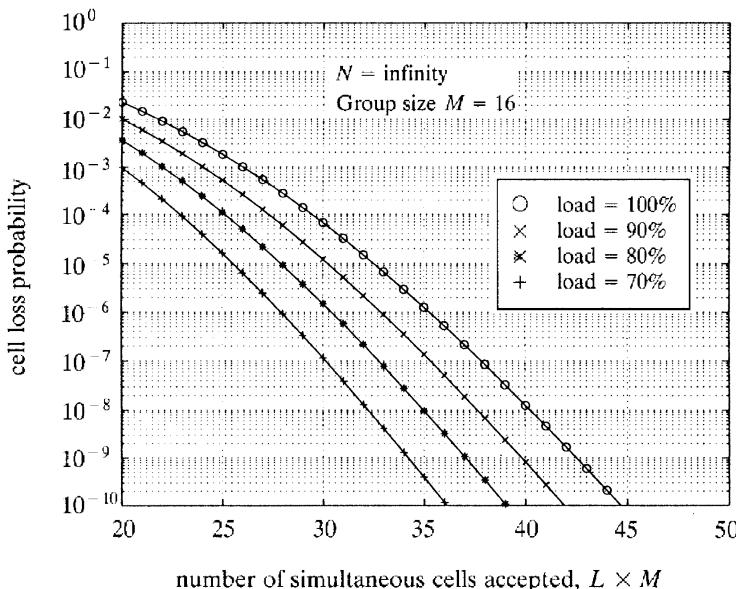


Fig. 6.11 Cell loss probability when using the generalized knockout principle.

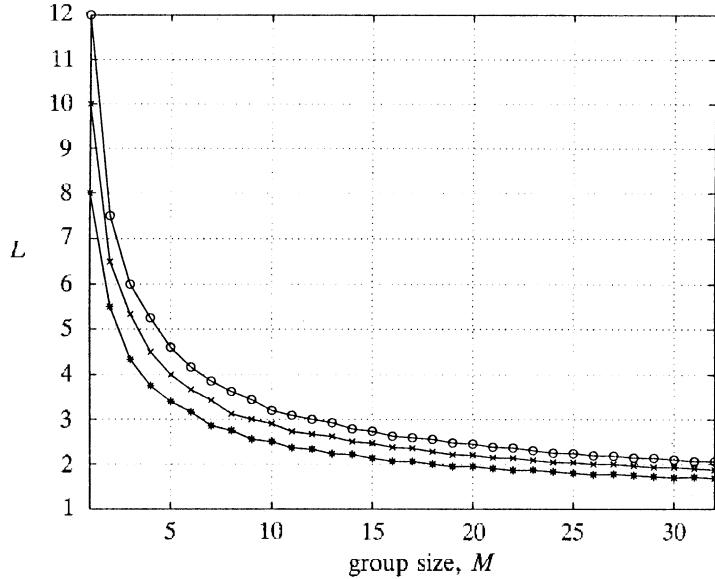


Fig. 6.12 Ratio of the number of simultaneous cells accepted to group size for various cell loss probabilities.

6.3 A TWO-STAGE MULTICAST OUTPUT-BUFFERED ATM SWITCH

6.3.1 Two-Stage Configuration

Figure 6.13 shows a two-stage structure of the multicast output buffered ATM switch (MOBAS) that adopts the generalized knockout principle described above. As a result, the complexity of interconnection wires and building elements can be reduced significantly, almost one order of magnitude [3]. The switch consists of input port controllers (IPCs), multicast grouping networks (MGN1, MGN2), multicast translation tables (MTTs), and output port controllers (OPCs). The IPCs terminate incoming cells, look up necessary information in translation tables, and attach the information (e.g., multicast patterns and priority bits) to the front of the cells so that cells can be properly routed in the MGNs. The MGNs replicate multicast cells based on their multicast patterns and send one copy to each output group. The MTTs facilitate the multicast cell routing in the MGN2. The OPCs temporarily store multiple arriving cells destined for that output port in an output buffer, generate multiple copies for multicast cells with a cell duplicator (CD), assign a new virtual channel identifier (VCI) obtained from a translation table to each copy, convert the internal cell format to the standardized ATM cell format, and finally send the cells to the next switching node or the final destination.

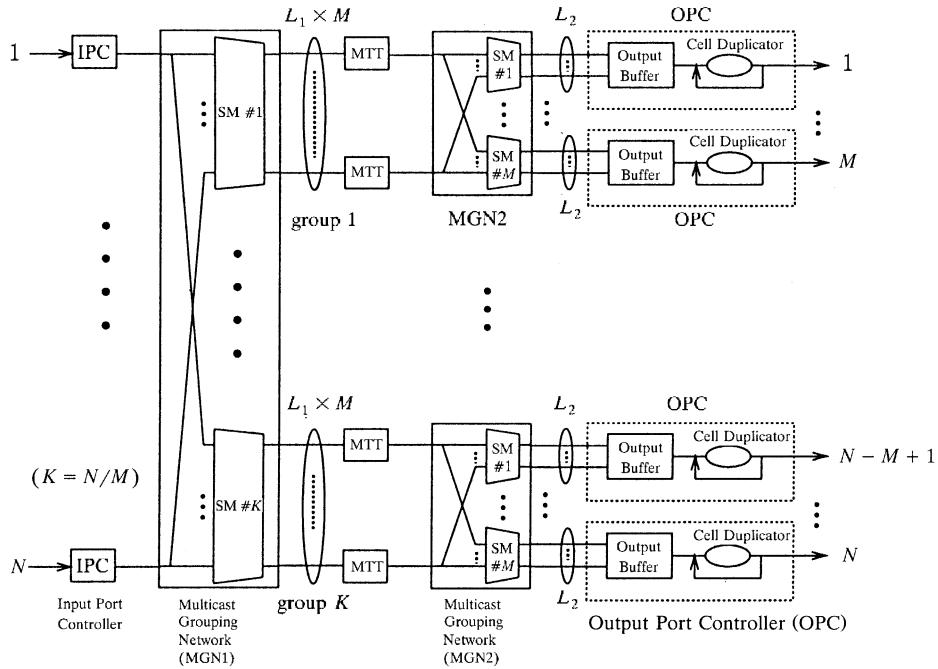


Fig. 6.13 The architecture of a multicast output buffered ATM switch. (©1995 IEEE.)

Let us first consider the unicast situation. As shown in Figure 6.13, every M output ports are bundled in a group, and there are a total of K groups ($K = N/M$) for a switch size of N inputs and N outputs. Due to cell contention, $L_1 M$ routing links are provided to each group of M output ports. If there are more than $L_1 M$ cells in one cell time slot destined for the same output group, the excess cells will be discarded and lost. However, we can engineer L_1 (or called group expansion ratio) so that the probability of cell loss due to the competition for the $L_1 M$ links is lower than that due to the buffer overflow at the output port or bit errors occurring in the cell header. The performance study in Section 6.2.2 shows that the larger the M is, the smaller L_1 needs to be to achieve the same cell loss probability. For instance, for a group size of one output port, which is the case in the second stage (MGN2), L_2 needs to be at least 12 to have a cell loss probability of 10^{-10} . But for a group size of 32 output ports, which is the case in the first stage (MGN1), L_1 just needs to be 2 to have the same cell loss probability. Cells from input ports are properly routed in MGN1 to one of the K groups; they are then further routed to a proper output port through the MGN2. Up to L_2 cells can arrive simultaneously at each output port. An output buffer is used to store these cells and send out one cell in each cell time slot. Cells

that originate from the same traffic source can be arbitrarily routed onto any of the $L_1 M$ routing links, and their cell sequence is still retained.

Now let us consider a multicast situation where a cell is replicated into multiple copies in MGN1, MGN2, or both, and these copies are sent to multiple outputs. Figure 6.14 shows an example to illustrate how a cell is replicated in the MGNs and duplicated in the CD. Suppose a cell arrives at an input port i and is to be multicast to four output ports: 1, M , $M + 1$, and N . The cell is first broadcast to all K groups in MGN1, but only groups, 1, 2, and K accept the cell. Note that only one copy of the cell will appear in each group, and the replicated cell can appear at any of the $L_1 M$ links. The copy of the cell at the output of group 1 is again replicated into two copies at MGN2. In total four replicated cells are created after MGN2. When each replicated cell arrives at the OPC, it can be further duplicated into multiple copies by the CD as needed. Each duplicated copy at the OPC is updated with a new VCI obtained from a translation table at the OPC before it is sent to the network. For instance, two copies are generated at output port 1, and three copies at output port $M + 1$. The reason for using the CD is to reduce the output port buffer size by storing only one copy of the multicast cell at each output port instead of storing multiple copies that are generated from the same traffic source and multicast to multiple virtual circuits on an output port. Also note that since there are no buffers in either MGN1 or MGN2, the

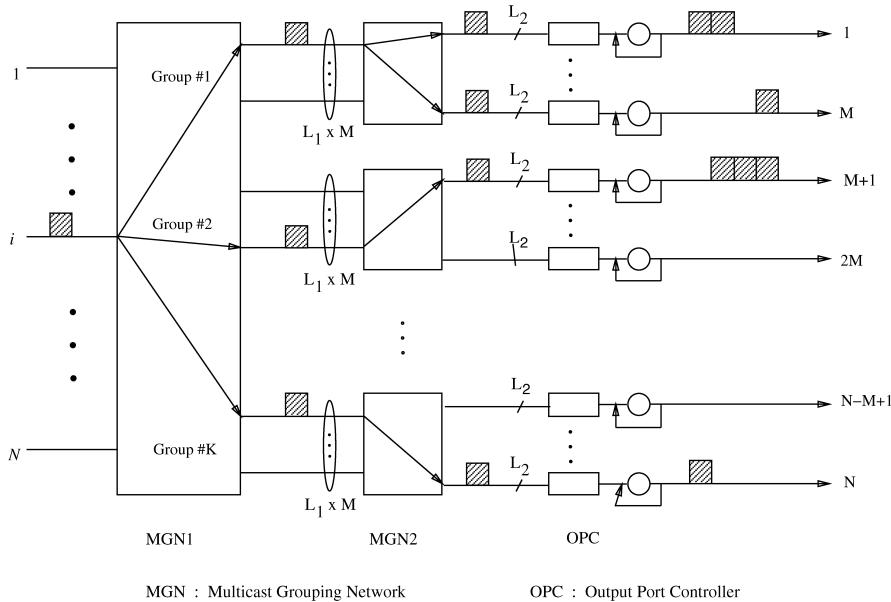


Fig. 6.14 An example of replicating cells for a multicast connection in the MOBAS.
(©1995 IEEE.)

replicated cells from either MGN are aligned in time. However, the final duplicated cells at the output ports may not be aligned in time, because they may have different queueing delays in the output buffers.

6.3.2 Multicast Grouping Network

Figure 6.15 shows a modular structure for the MGN at the first or the second stage. The MGN consists of K switch modules for the first stage or M for the second stage. Each switch module contains a switch element (SWE) array, a number of multicast pattern maskers (MPMs), and an address broadcaster (AB). The AB generates dummy cells that have the same destined address as the output. This enables cell switching to be done in a distributed manner and permits the SWE not to store output group address information, which simplifies the circuit of the SWE significantly and results in higher VLSI integration density. Since the structure and operation for MGN1 and MGN2 are identical, let us just describe MGN1.

Each switch module in MGN1 has N horizontal input lines and $L_1 M$ vertical routing links, where $M = N/K$. These routing links are shared by the cells that are destined for the same output group of a switch module. Each

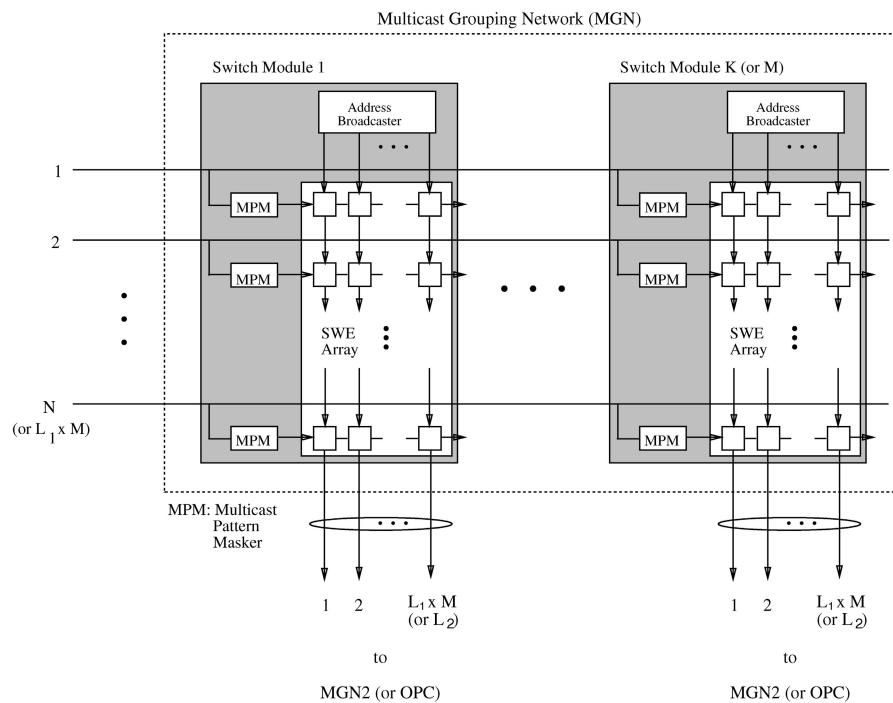


Fig. 6.15 The multicast grouping network. (©1995 IEEE.)

input line is connected to all switch modules, allowing a cell from each input line to be broadcast to all K switch modules.

The routing information carried in front of each arriving cell is a multicast pattern, which is a bit map of all the outputs in the MGN. Each bit indicates if the cell is to be sent to the associated output group. For instance, let us consider a multicast switch with 1024 inputs and 1024 outputs and the numbers of groups in MGN1 and MGN2, K and M , both chosen to be 32. Thus, the multicast pattern in both MGN1 and MGN2 has 32 bits. For a unicast cell, the multicast pattern is basically a flattened output address (i.e., a decoded output address) in which only one bit is set to 1 and all other 31 bits are set to 0. For a multicast cell, there are more than one bit in the multicast pattern set to 1. For instance, if a cell X is multicast to switch modules i and j , the i th and j th bits in the multicast pattern are set to 1.

The MPM performs a logic AND function for the multicast pattern with a fixed 32-bit pattern in which only the i th bit, corresponding to switch module i , is set to 1 and all other 31 bits are set to 0. So, after cell X passes through the MPM in switch module i , its multicast pattern becomes a flattened output address where only the i th bit is set to 1.

Each empty cell that is transmitted from the AB has attached to it, in front, a flattened output address with only one bit set to 1. For example, empty cells from the AB in switch module i have only the i th bit set to 1 in their flattened address. Cells from horizontal inputs will be properly routed to different switch modules based on the result of matching their multicast patterns with empty cells' flattened addresses. For cell X , since its i th and j th bits in the multicast pattern are both set to 1, it matches with the flattened addresses of empty cells from the ABs in switch modules i and j . Thus, cell X will be routed to the output of these two switch modules.

The SWE has two states, cross state and toggled state, as shown in Figure 6.16. The state of the SWE depends on the comparison result of the flattened addresses and the priority fields in cell headers. The priority is used for cell contention resolution. Normally, the SWE is at cross state, i.e., cells from the north side are routed to the south side, and cells from the west side are routed to the east side. When the flattened address of the cell from the west (FA_w) is matched with the flattened address of the cell from the north (FA_n), and when the west's priority level (P_w) is higher than the north's (P_n), the SWE's state is toggled: The cell from the west side is routed to the south side, and the cell from the north is routed to the east. In other words, any unmatched or lower-priority (including the same priority) cells from the west side are always routed to the east side. Each SWE introduces a 1-bit delay as the bit stream of a cell passes it in either direction. Cells from MPMs and AB are skewed by one bit before they are sent to each SWE array, due to the timing alignment requirement.

Figure 6.17 shows an example of how cells are routed in a switch module. Cells U, V, W, X, Y , and Z arrive at inputs 1 to 6, respectively, and are to be routed in switch module 3. In the cell header, there is a 3-bit multicast

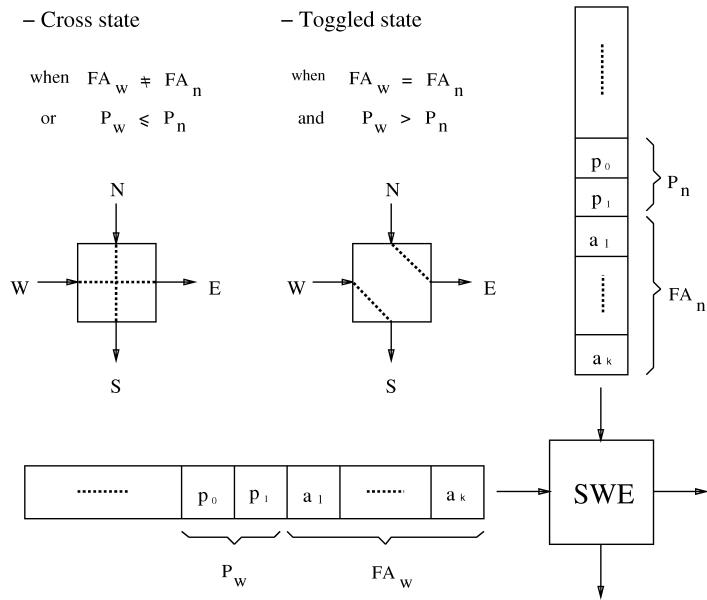


Fig. 6.16 Switching condition of the switch element (SWE).

pattern ($m_3 \ m_2 \ m_1$) and a 2-bit priority field ($p_1 \ p_0$). If a cell is to be sent to an output of this switch module, its m_3 bit will be set to 1. Among these six cells, cells U, V , and X are for unicast, where only one bit in the multicast pattern is set to 1. The other three cells are for multicast, where more than one bit in the multicast pattern is set to 1. It is assumed that a smaller priority value has a higher priority level. For instance, cell Z has the highest priority level (00) and empty cells transmitted from the address broadcaster have the lowest priority level (11). The MPM performs a logic AND function for each cell's multicast pattern with a fixed pattern of 100. For instance, after cell W passes through the MPM, its multicast pattern (110) becomes 100 ($a_3 \ a_2 \ a_1$), which has only one bit set to 1 and is taken as the flattened address. When cells are routed in the SWE array, their routing paths are determined by the state of SWEs, which are controlled according to the rules in Figure 6.16. Since cells V and X are not destined for this group, the SWEs they pass remain in a cross state. Consequently, they are routed to the right side of the module and are discarded. Since there are only three routing links in this example, while there are four cells destined to this switch module, the one with the lowest priority (i.e., cell U) loses contention to the other three and is discarded.

Since the crossbar structure inherits the characteristics of identical and short interconnection wires between switch elements, the timing alignment for the signals at each SWE is much easier than that of other types of

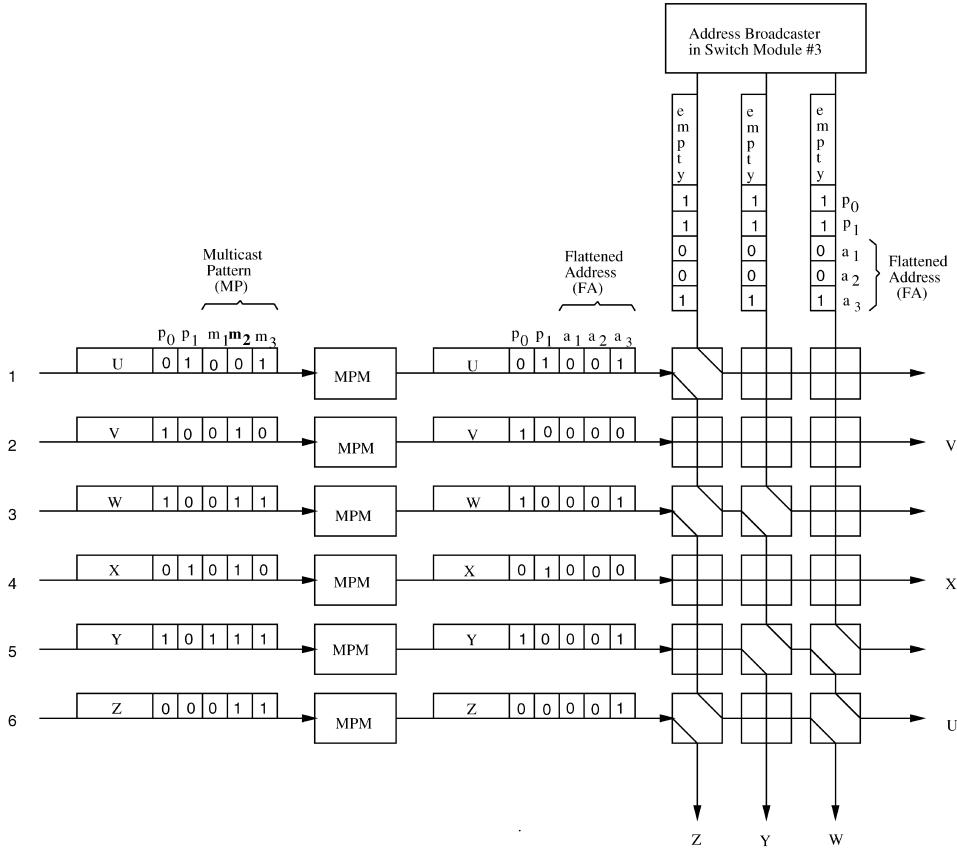
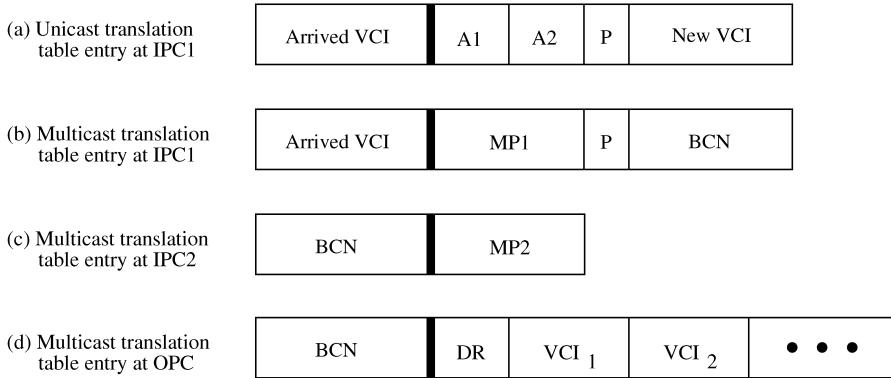


Fig. 6.17 An example of routing a multicast cell.

interconnection network, such as the binary network, the Clos network, and so on. The unequal length of the interconnection wires increases the difficulty of synchronizing the signals, and, consequently limits the switch fabric's size, as with the Batcher-banyan switch. The SWEs in the switch modules only communicate locally with their neighbors, as do the chips that contain a two-dimensional SWE array. The switch chips do not need to drive long wires to other chips on the same printed circuit board. Note that synchronization of data signals at each SWE is only required in each individual switch module, not in the entire switch fabric.

6.3.3 Translation Tables

The tables in the IPC, MTT, and the OPC (Fig. 6.18) contain information necessary to properly route cells in the switch modules of MGN1 and MGN2, and to translate old VCI values into new VCI values. As mentioned above, cell routing in the MGNs depends on the multicast pattern and priority



VCI : Virtual channel identifier

A1 : Output address of MGN1

MP1 : Multicast pattern in MGN1

P : Priority for cell contention

BCN : Broadcast channel number

A2 : Output address of MGN2

MP2 : Multicast pattern in MGN2

DR : Number of duplication requests

Fig. 6.18 Translation tables in the IPC, MTT, and OPC.

values that are attached in front of the incoming cell. To reduce the translation table's complexity, the table contents and the information attached to the front of cells are different for the unicast and multicast calls.

In a point-to-point ATM switch, an incoming cell's VCI on an input line can be identical with other cells' VCIs on other input lines. Since the VCI translation table is associated with the IPC on each input line, the same VCI values can be repeatedly used for different virtual circuits on different input lines without causing any ambiguity. But cells that are from different virtual circuits and destined for the same output port need different translated VCIs.

In a multicast switch, since a cell is replicated into multiple copies that are likely to be transmitted on the same routing link inside a switch fabric, the switch must use another identifier, the broadcast channel number (BCN), to uniquely identify each multicast connection. In other words, the BCNs of multicast connections have to be different from each other, which is unlike the unicast case, where a VCI value can be repeatedly used for different connections at different input lines. The BCN can either be assigned during call setup or be defined as a combination of the input port number and the VPI/VCI value.

For the unicast situation, upon a cell's arrival, its VCI value is used as an index to access the necessary information in the IPC's translation table, such as the output addresses in MGN1 and MGN2, a contention priority value, and a new VCI, as shown in Figure 6.19(a). The output address of MGN1,

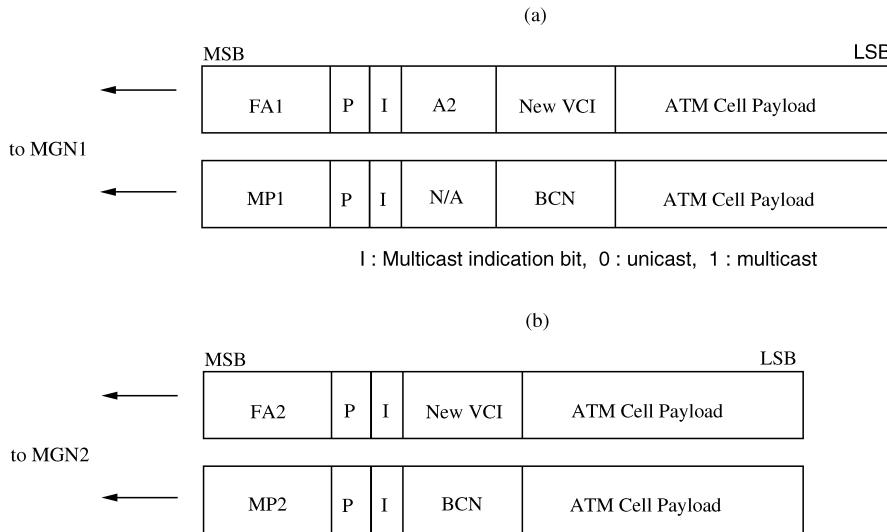


Fig. 6.19 Unicast and multicast cell formats in (a) MGN1, (b) MGN2.

A_1 , is first decoded into a flattened address, which has K bits and is put into the MP_1 field in the cell header, as shown in Figure 6.19(a). The MP_1 and P are used for routing cells in the MGN1, and A_2 is used for routing cells in the MGN2. The I bit is the multicast indication bit, which is set to 0 for unicast and 1 for multicast. When a unicast cell arrives at the MTT, the A_2 field is simply decoded into a flattened address and put into the MP_2 field as the routing information in MGN2. Thus, no translation table in the MTT is needed for unicast cells. Note that A_2 is not decoded into a flattened address until it arrives at the MTT. This saves some bits in the cell header (e.g., 27 bits for the above example of a 1024×1024 switch) and thus reduces the required operation speed in the switch fabric. The unicast cell routing format in MGN2 is shown in Figure 6.19(b).

For the multicast situation, besides the routing information of MP_1 , MP_2 , and P , the BCN is used to identify cells that are routed to the same output group of a switch module. Similarly to the unicast case, an incoming cell's VCI is first used to look up the information in the translation table in the IPC, as shown in Figure 6.18(b). After a cell has been routed through MGN1, MP_1 is no longer used. Instead, the BCN is used to look up the next routing information, MP_2 , in the MTT, as shown in Figure 6.18(c). The cell formats for a multicast call in MGN1 and MGN2 are shown in Figure 6.19(a) and (b). The BCN is further used at the OPC to obtain a new VCI for each duplicated copy that is generated by a cell duplicator at the OPC. The entry for the multicast translation table in the OPC is shown in Figure 6.18(d).

Note that the MTTs that are connected to the same MGN2 contain identical information, because the copy of a multicast cell can appear randomly at any one of L_1M output links to MGN2. As compared to those in [18, 26], the translation table size in the MTT is much smaller. This is because the copy of a multicast cell can only appear at one of L_1M output links in the MOBAS, vs. N links in [18, 26], resulting in fewer table entries in the MTT. In addition, since the VCI values of replicated copies are not stored in the MTT's, the content of each table entry in the MTT is also less.

6.3.4 Multicast Knockout Principle

A new multicast knockout principle, an extension of generalized knockout principle, has been applied to the two-stage MOBAS to provide multicasting capability. Since the switch module (SM) in the MGN performs a concentration function (e.g., N to L_1M), it is also called a concentrator.

6.3.4.1 Cell Loss Rate in MGN1 In the analysis, it is assumed that the traffic at each input port of the MOBAS is independent of that at the other inputs, and replicated cells are uniformly delivered to all output groups. The average cell arrival rate, ρ , is the probability that a cell arrives at an input port in a given cell time slot. It is assumed that the average cell replication in MGN1 is $E[F_1]$, the average cell replication in MGN2 is $E[F_2]$, the average cell duplication in the OPC is $E[D]$, and the random variables F_1 , F_2 , and D are independent of each other.

Every incoming cell is broadcast to all concentrators (SMs), and is properly filtered at each concentrator according to the multicast pattern in the cell header. The average cell arrival rate, p , at each input of a concentrator is

$$p = \frac{\rho E[F_1]}{K},$$

where K ($= N/M$) is the number of concentrators in MGN1. The probability (A_k) that k cells are destined for a specific concentrator of MGN1 in a given time slot is

$$\begin{aligned} A_k &= \binom{N}{k} p^k (1-p)^{N-k} \\ &= \binom{N}{k} \left(\frac{\rho E[F_1] \cdot M}{N} \right)^k \left(1 - \frac{\rho E[F_1] \cdot M}{N} \right)^{N-k}, \quad 0 \leq k \leq N, \end{aligned} \quad (6.8)$$

where $\rho E[F_1] \cdot M/N$ is the probability of a cell arriving at the input of a specific concentrator in MGN1.

As $N \rightarrow \infty$, (6.8) becomes

$$A_k = e^{-\rho E[F_1] \cdot M} \frac{(\rho E[F_1] \cdot M)^k}{k!} \quad (6.9)$$

where ρ should satisfy the following condition for a stable system:

$$\rho E[F_1] E[F_2] E[D] < 1.$$

Since there are only $L_1 M$ routing links available for each output group, if more than $L_1 M$ cells are destined for this output group in a cell time slot, excess cells will be discarded and lost. The cell loss rate in MGN1, P_1 , is

$$P_1 = \frac{\sum_{k=L_1 M+1}^N (k - L_1 M) A_k E[F_2] E[D]}{N p E[F_2] E[D]} \quad (6.10)$$

$$= \frac{1}{\rho E[F_1] \cdot M} \sum_{k=L_1 M+1}^N (k - L_1 M) \times \binom{N}{k} \left(\frac{\rho E[F_1] \cdot M}{N} \right)^k \left(1 - \frac{\rho E[F_1] \cdot M}{N} \right)^{N-k}. \quad (6.11)$$

Both the denominator and the numerator in (6.10) include the factor $E[F_2] E[D]$ to allow for the cell replication in MGN2 and the OPC. In other words, a cell lost in MGN1 could be a cell that would have been replicated in MGN2 and the OPC. The denominator in (6.10), $N p E[F_2] E[D]$, is the average number of cells effectively arriving at a specific concentrator during one cell time, and the numerator in (6.10), $\sum_{k=L_1 M+1}^N (k - L_1 M) A_k E[F_2] E[D]$, is the average number of cells effectively lost in the specific concentrator.

As $N \rightarrow \infty$, (6.11) becomes

$$\begin{aligned} P_1 &= \left(1 - \frac{L_1 M}{N p} \right) \left(1 - \sum_{k=0}^{L_1 M} \frac{(N p)^k e^{-N p}}{k!} \right) + \frac{(N p)^{L_1 M} e^{-N p}}{(L_1 M)!} \\ &= \left(1 - \frac{L_1 M}{\rho E[F_1] M} \right) \left(1 - \sum_{k=0}^{L_1 M} \frac{(\rho E[F_1] \cdot M)^k e^{-\rho E[F_1] \cdot M}}{k!} \right) \\ &\quad + \frac{(\rho E[F_1] \cdot M)^{L_1 M} e^{-\rho E[F_1] \cdot M}}{(L_1 M)!} \end{aligned} \quad (6.12)$$

Note that (6.12) is similar to the equation for the generalized knockout principle, except that the parameters in the two equations are slightly different due to the cell replication in MGN1 and MGN2 and to the cell duplication in the OPC.

Figure 6.20 shows the plots of the cell loss probability at MGN1 vs. L_1 for various fanout values and an offered load ($= \rho E[F_1]E[F_2]E[D]$) of 0.9 at each output port. Again it shows that as M increases (i.e., more outputs are sharing their routing links), the required L_1 value decreases for a given cell loss rate. The requirements on the switch design parameters M and L_1 are more stringent in the unicast case than in the multicast. Since the load on MGN1 decreases as the product of the average fanouts in MGN2 and the

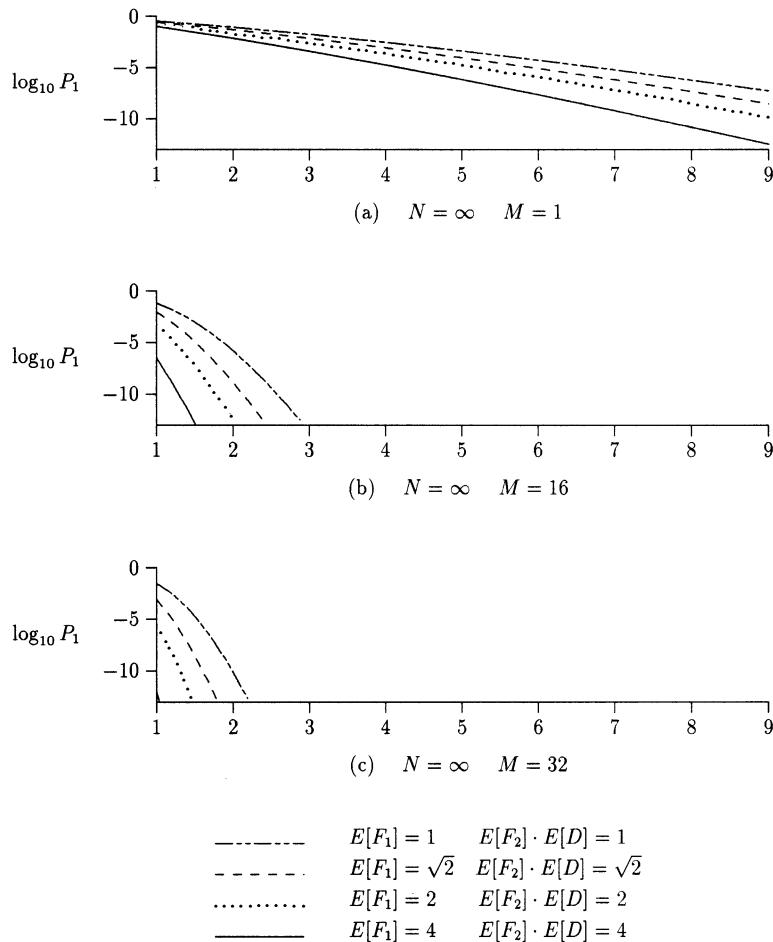


Fig. 6.20 Cell loss probability vs. the group expansion ratio L_1 in MGN1.

OPC increases, the cell loss rate of MGN1 in the multicast case is lower than that in the unicast case. The replicated cells from a multicast call will never contend with each other for the same output group (concentrator), since the MOBAS replicates at most one cell for each output group. In other words, an MGN1 that is designed to meet the performance requirement for unicast calls will also meet the one for calls.

6.3.4.2 Cell Loss Rate in MGN2 Special attention is required when analyzing the cell loss rate in MGN2, because the cell arrival pattern at the inputs of MGN2 is determined by the number of cells passing through the corresponding concentrator in MGN1. If there are l ($l \leq L_1 M$) cells arriving at MGN2, these cells will appear at the upper l consecutive inputs of MGN2. If more than $L_1 M$ cells are destined for MGN2, only $L_1 M$ cells will arrive at MGN2's inputs (one cell per input port), while excessive cells are discarded in MGN1.

The cell loss rate of a concentrator in MGN2 is assumed to be P_2 , and the probability that l cells arrive at a specific concentrator in MGN2 to be B_l . Both P_2 and B_l depend on the average number of cells arriving at the inputs of MGN2 (i.e., the number of cells passing through the corresponding concentrator in MGN1). This implies that P_2 is a function of A_k in (6.8). In order to calculate P_2 , the probability of j cells arriving at MGN2, denoted as I_j , is found to be

$$I_j = \begin{cases} A_j & \text{for } j < L_1 M, \\ \sum_{k=L_1 M}^N A_k & \text{for } j = L_1 M. \end{cases}$$

If j ($j \leq L_1 M$) cells arrive at MGN2, they will appear at the upper j consecutive inputs of MGN2. Since how and where the cells appear at the MGN2's inputs does not affect the cell loss performance, the analysis can be simplified by assuming that a cell can appear at any input of the MGN2.

Let us denote by $B_{l|j}$ the probability that l cells arrive at the inputs of a specific concentrator in MGN2 for given j cells arrived at MGN2. Then,

$$B_{l|j} = \binom{j}{l} q^l (1-q)^{j-l}, \quad 0 \leq j \leq L_1 M, \quad 0 \leq l \leq j,$$

where q is equal to $E[F_2]/M$ under the assumption that replicated cells are uniformly delivered to the M concentrators in MGN2.

If no more than L_2 cells arrive at MGN2 ($0 \leq j \leq L_2$), no cell will be discarded in MGN2, because each concentrator can accept up to L_2 cells during one cell time slot. If more than L_2 cells arrive at MGN2 ($L_2 \leq j \leq L_1 M$), cell loss will occur in each concentrator with a certain probability.

Since replicated cells are assumed to be uniformly distributed to all M outputs in MGN2, the probability B_l that l cells are destined for a specific output port of MGN2 in a cell time slot is

$$B_l = \sum_{j=l}^{L_1 M} B_{l|j} I_j.$$

The cell loss rate in MGN2, P_2 , is

$$\begin{aligned} P_2 &= \frac{\sum_{l=L_2+1}^{L_1 M} (l - L_2) B_l E[D]}{N \cdot \frac{\rho E[F_1]}{K} (1 - P_1) E[F_2] \cdot E[D]} \\ &= \frac{\sum_{l=L_2+1}^{L_1 M} (l - L_2) B_l}{\rho E[F_1] \cdot (1 - P_1) E[F_2]}. \end{aligned} \quad (6.13)$$

Here $N \cdot \rho E[F_1]/K$ is the average number of cells destined for a concentrator in MGN1 from the inputs of the MOBAS, and $N \cdot (\rho E[F_1]/K)(1 - P_1)$ is the average number of cells that have survived in this concentrator, which in turn becomes the average number of cells arriving at the corresponding MGN2. Thus, the denominator in (6.13), $N \cdot (\rho E[F_1]/K)(1 - P_1) E[F_2]E[D]/M$, is the average number of cells effectively arriving at a specific output port. The numerator in (6.13), $\sum_{l=L_2+1}^{L_1 M} (l - L_2) B_l E[D]$, is the average number of cells effectively lost in a specific concentrator in MGN2, because the lost cell can be a cell that will be duplicated in the OPC.

Figure 6.21 shows the plots of the cell loss probability at MGN2 vs. L_2 for various average duplication values and an effective offered load ($= \rho E[F_1]E[F_2]E[D]$) of 0.9. The average fanout on MGN1 is assumed to be 1.0 ($E[F_1] = 1.0$), the group size to be 32 ($M = 32$), and the expansion ratio to be 2.0 ($L_1 = 2.0$). Since the traffic load on MGN2 decreases as the average cell duplication ($E[D]$) increases, the cell loss rate in MGN2 decreases as $E[D]$ increases. Therefore, the switch design parameter L_2 is more stringently restricted in the unicast case than in the multicast. Consequently, if MGN2 is designed to meet the performance requirement for unicast calls, it will also satisfy multicast calls' performance requirement.

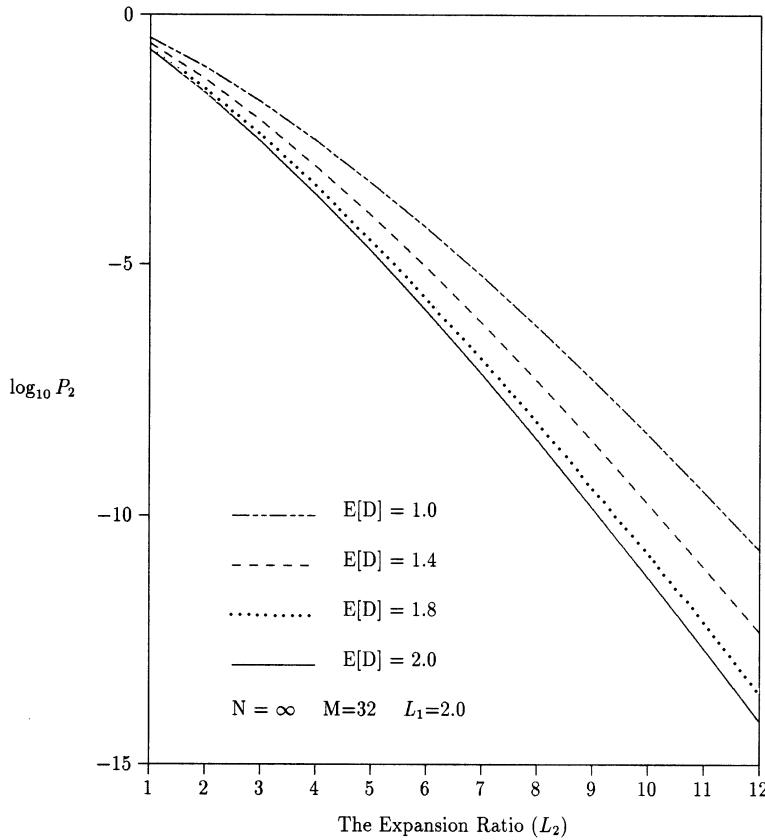


Fig. 6.21 Cell loss probability vs. group expansion ratio L_2 in MGN2.

6.3.4.3 Total Cell Loss Rate in the MOBAS The total cell loss rate in the MOBAS is

$$\begin{aligned}
 P_T &= \frac{\text{Avg. no. of cells effectively lost in both MGN1 and MGN2}}{\text{Avg. no. of cells effectively offered to MOBAS}} \\
 &= \frac{1}{N \cdot \rho E[F_1]E[F_2]E[D]} \left(\frac{N}{M} \sum_{k=L_1M+1}^N (k - L_1M) A_k E[F_2] E[D] \right. \\
 &\quad \left. + N \sum_{l=L_2+1}^{L_1M} (l - L_2) B_l E[D] \right). \quad (6.14)
 \end{aligned}$$

The first term of the numerator in (6.14), $(N/M)\sum_{k=L_1M+1}^N (k - L_1M) A_k E[F_2] E[D]$, is the average number of cells effectively lost in MGN1. The second term of the numerator in (6.14), $N\sum_{l=L_2+1}^{L_1M} (l - L_2) B_l E[D]$, is the average number of cells effectively lost in all of MGN2. The denominator in

(6.14), $N \cdot \rho E[F_1]E[F_2]E[D]$, is the average number of cells effectively offered to the MOBAS.

In (6.14) L_1 and L_2 should be in a certain range to guarantee the required cell loss performance in the MOBAS. For example, in order to get the cell loss rate of 10^{-10} in the MOBAS with a very large size ($N \geq 1024$) and a group size of $M (= 32)$, L_1 and L_2 should be greater than 2 and 12, respectively. If either of them is less than the required number, the total cell loss rate in the MOBAS cannot be guaranteed, since the term of the numerator of (6.14), that is a function of the smaller number, dominates the total cell loss rate in the MOBAS.

6.4 A FAULT-TOLERANT MULTICAST OUTPUT-BUFFERED ATM SWITCH

Fault tolerance is an important issue when designing ATM switches for reliable communications. In particular, as line rates increase to 2.5 or 10 Gbit/s, the failure of a single switch element in the switch fabric will disrupt the service of all connections on the link that passes through the failure point. Several interconnection networks, along with some techniques, including time redundancy and the space redundancy, have been proposed to achieve network fault tolerance [1, 11, 27, 12, 15, 29, 17]. The time redundancy technique uses a multiple-pass routing strategy to obtain dynamic full access between inputs and outputs. More specifically, it permits each input to be accessible to any output in a finite number of recirculations through a switching network. The space redundancy technique provides multiple paths from each input to all outputs by using redundant hardware (additional switching elements, links, or stages) [16]. This additional hardware increases the system complexity and complicates the operations of the switch.

Because of the property of a unique routing path between any input and output pair in binary self-routing networks, it is difficult to achieve fault tolerance without adding additional switching planes, stages, or switching elements. As shown in Figure 6.13, there are multiple paths between each input and output pair. Therefore, it is inherently robust against faulty switching elements and internal link failures. This section presents different techniques of fault diagnosis (including fault detection and location) and system reconfiguration (by isolating faulty switching elements) for the MOBAS [5]. The regular structure of the MOBAS also simplifies fault diagnosis and system reconfiguration when a fault occurs.

6.4.1 Fault Model of Switch Element

A fault in the SWE can happen in the control logic circuit or on the data link of the SWE. In the former case, the SWE will remain in either a cross state or a toggle state; these faults are called cross-stuck or toggle-stuck, respectively. If a data link is short-circuited or open-circuited, the output of the SWE is either at a high or low state; it is then called stuck-at-one (s-a-1) or

stuck-at-zero (s-a-0). Cells passing through the faulty links are always corrupted. The link failure can occur at either a vertical or a horizontal link; it is then called vertical-stuck or horizontal-stuck, respectively. Let $SWE(i, j)$ represent the switch element located at the i th row and the j th column in the SWE array. Denote by $CS(i, j)$, $TS(i, j)$, $VS(i, j)$, and $HS(i, j)$ a cross-stuck, toggle-stuck, vertical-stuck, and horizontal-stuck fault at $SWE(i, j)$, respectively.

In the following, we only discuss single-fault failures, since they are much more likely to happen than multiple-fault failures in integrated circuits [2].

6.4.1.1 Cross-Stuck (CS) Fault Figure 6.22 shows the cell routing in an SWE array with across-stuck $SWE(4, 3)$ [the shaded one, denoted $CS(4, 3)$], where a cell from the north is always routed to the south, and a cell from the west to the east. If the CS fault occurs in the last column of the SWE array, as shown in Figure 6.22, cell loss performance is degraded. For instance, cell

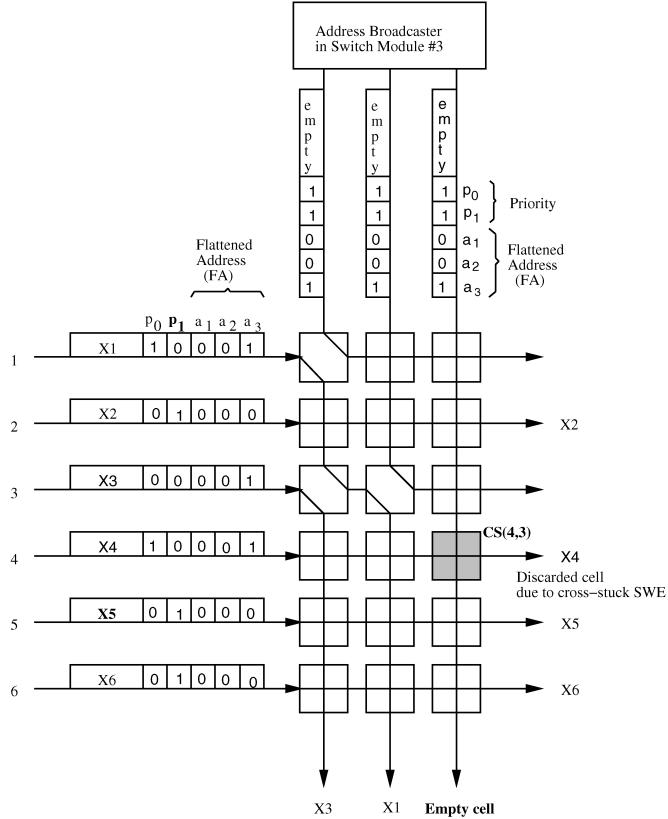


Fig. 6.22 An example of cell loss due to a cross-stuck fault at $SWE(4, 3)$, denoted $CS(4, 3)$. (©1994 IEEE.)

X_4 is discarded because of the cross-stuck SWE(4, 3). If there are only three cells from the first to the fourth input destined for this output group, one of these three will be discarded at SWE(4, 3), which contributes to cell loss performance degradation.

However, if X_4 is not the third cell destined for this group, or there are more than three (the number of output links) cells destined for this group, the CS in the last column does not contribute to cell loss performance degradation.

6.4.1.2 Toggle-Stuck (TS) Fault Figure 6.23 shows how a toggle-stuck SWE(4, 2) affects the cell routing in the SWE array. Unlike a cross-stuck fault, a toggle-stuck fault may result in cells being misrouted. If the west input of the toggle-stuck SWE in Figure 6.23 does not have a cell destined for this output, an output link is wasted because the link is mistakenly occupied by this input. Therefore, every input, except this input, may experi-

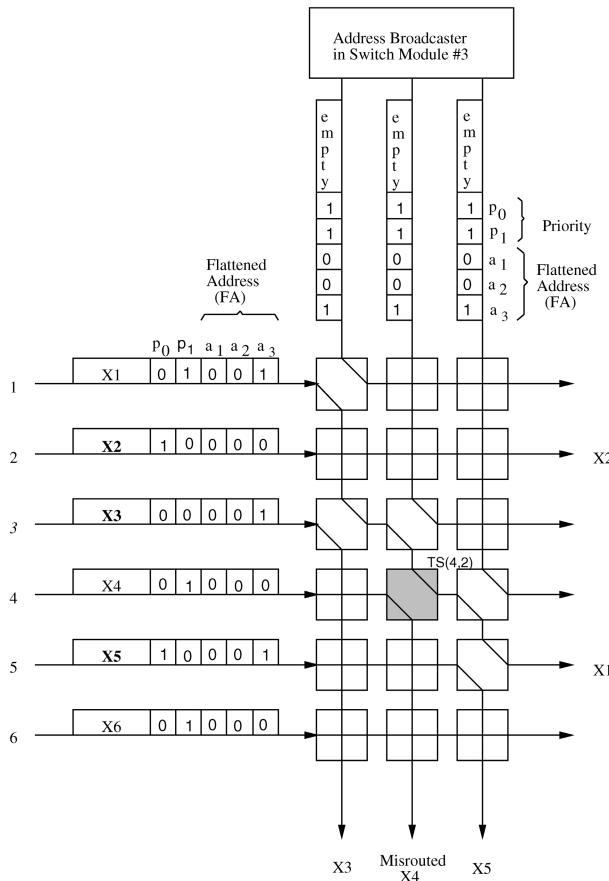


Fig. 6.23 An example of cell loss due to the toggle-stuck fault at SWE(4, 2), denoted TS(4, 2). (©1994 IEEE.)

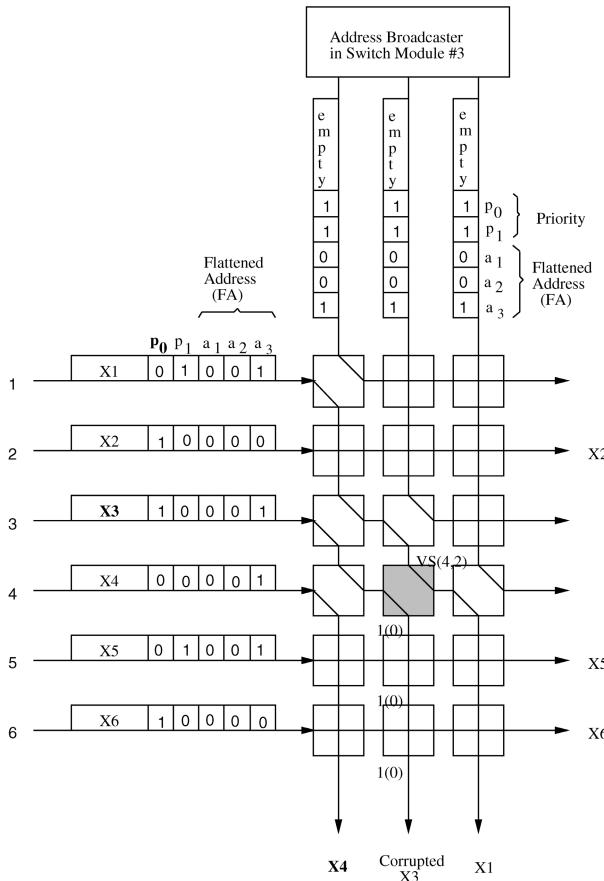


Fig. 6.24 A cell loss example due to the vertical stuck-at-1 or -0 fault at SWE(4, 2), denoted VS(4, 2). (©1994 IEEE.)

ence a cell loss performance degradation. For example, in Figure 6.23, cell X1 is discarded due to the fault of TS(4, 2).

6.4.1.3 Vertical/Horizontal-Stuck (VS/HS) Fault Figure 6.24 shows the cell routing in the SWE array when a fault of stuck-at-one (or -zero) is found at the vertical link of SWE(4, 2). Figure 6.25 shows a fault at the horizontal link. Cells passing through this faulty SWE are always corrupted at either the vertical or the horizontal link. This might cause data retransmission and increase the network load and the congestion probability.

6.4.2 Fault Detection

To detect a fault, it is assumed that the MTTs, the OPCs, and the horizontal outputs of the SWE array (where cells are discarded) have simple logic circuits, called fault detectors (FDs), which are capable of detecting any

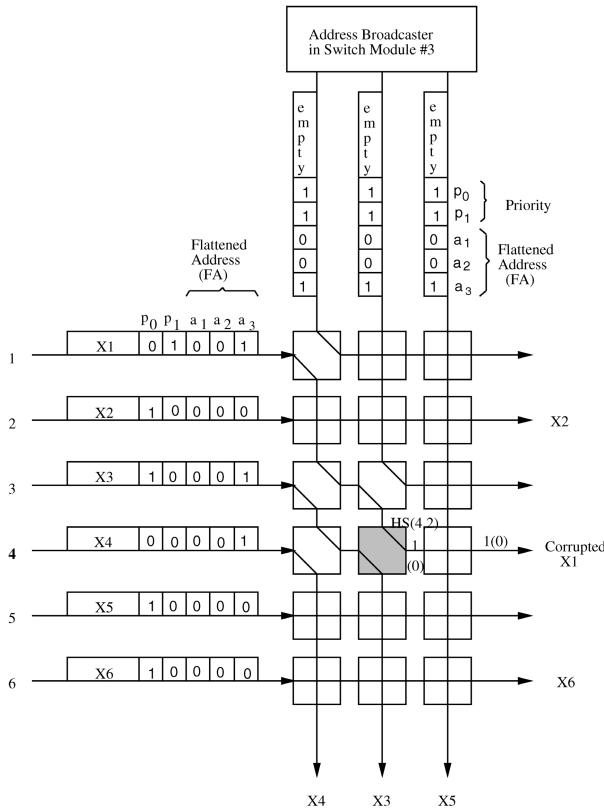


Fig. 6.25 A cell loss example due to a horizontal stuck-at-1 or -0 fault at SWE(4, 2), denoted HS(4, 2). (©1994 IEEE.)

misrouted cells in the switch modules and thus providing on-line fault detection capability. In addition, the MPMs and the ABs are assumed to be able to generate test cells to locate faulty SWEs once a fault is detected by the FDs.

The fault diagnosis and system reconfiguration can be carried out at any time when needed by retaining users' cells at the IPCs and feeding test cells to the switch fabric. Because the test can be completed within a couple of cell slots, the overhead will not affect the switch's normal operations.

The FDs in MTTs or OPCs of the k th SM examine the k th bit of a flattened address (zero or one) to determine a misrouted cell. They also examine the source address in the test cell to locate a faulty SWE.

6.4.2.1 Cross-Stuck and Toggle-Stuck Fault Detection A TS fault can be detected by monitoring cells routed to MTTs or the OPCs. If a cell routed to an MTT or OPC has a FA of all zeros, it is considered a misrouted cell. This is because any cell with all zeros in the FA should not be routed to the

south. As shown in Figure 6.23, when cell X4 is misrouted to the second output, the associated FD detects this misrouted cell because its FA is all 0s.

While the TS fault can be easily detected online, the CS fault cannot be detected online. This fault will not contribute to cell-loss performance degradation if it is not located in the last column. Even if it is in the last column, we do not care about the CS fault, since this kind of sticking is exactly the action we will take for fault isolation (details are explained later).

6.4.2.2 Vertical-Stuck and Horizontal-Stuck Fault Detection VS and HS faults can be easily detected by checking if the outgoing signal from the SWE array is always one or zero. For instance, as shown in Figure 6.24, the signal from the second output link is always one (or zero), which indicates that one of the SWEs in the second column must be a VS. Once a VS fault occurs, e.g., at SWE(4, 2), all the SWEs below the faulty one remain at the cross state, so that the s-2-0 and s-2-1 faults will display at the south output of the faulty column. Cells that reach the south outputs are either valid cells from the user or empty cells from the ABs. They do not have the pattern of all zeros or all ones in their headers (consisting of the FA and priority fields). So, if a cell appears at a south output and has a pattern of all zeros or all ones, it must be a VS fault.

Figure 6.25 shows that the signal from the fourth discarding output is always one (or zero), and that there must be an SWE HS fault in the fourth row. Once a HS fault occurs, e.g., at SWE(4, 2), all the SWEs on the right of the faulty one remain in the cross state, so that the s-a-0 and s-a-1 faults will display at the east output of the faulty row. Cells that reach the east outputs can be valid cells from the user, empty cells from the ABs, or idle cells from the IPCs. Let us assume the address field of the idle cells is set to all zeros while the priority field is set to all ones (corresponding to the lowest priority). So, if a cell appears at an east output and has a pattern of all zeros or all ones, it must be a HS fault.

6.4.3 Fault Location and Reconfiguration

Once a fault is detected, we need more information to locate the fault and reconfigure the switching network. The fault detection itself provides partial information about the location of the faulty SWE (e.g., either the row or column identification). In order to properly reconfigure the network, we need to exactly locate a HS SWE and a TS SWE. For the VS fault, we just need to know the column information. A test cell, consisting of a FA field, a new priority field, and an input source address field, is generated by the MPM or the AB to locate the fault. Its FA field has the same length as the regular cell format in the MOBAS. The new priority field is $\lceil \log_2 2L_1 M \rceil$ bits long in MGN1 and $\lceil \log_2 2L_2 \rceil$ bits in MGN2. The input source address field has $\lceil \log_2 N \rceil$ bits in MGN1 and $\lceil \log_2 L_1 M \rceil$ bits in MGN2.

6.4.3.1 Toggle-Stuck and Cross-Stuck Cases By performing fault detection for the TS, the location of the faulty column can be identified. In order to locate the faulty row, we define a location test (called the TS test) for a TS fault. Figure 6.26 shows an example of the TS test. Cells coming from MPMs are forced to have a different FA than those from the AB; thus, all SWEs in the SWE array are set to a cross state. As a result, fault-free condition cells from the MPMs all appear at the east side of the SWE array, while those from the AB appear at the south side. If there is a TS at SWE(i, j), cells from the i th MPM will be delivered to the j th output link, while cells from the j th column of the AB are routed to the i th discarding output.

Once we have exactly located the TS fault, say at (i, j) , we can reconfigure the SWE array by setting SWE(i, j) to a cross state, as shown in Figure 6.27.

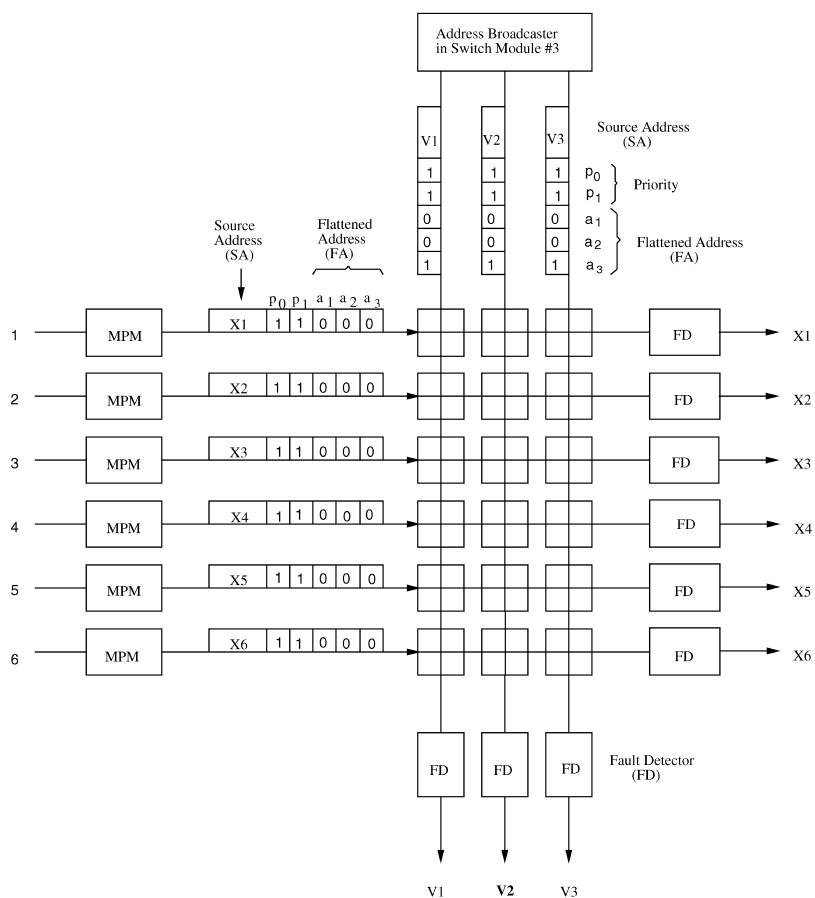


Fig. 6.26 Fault location test for a toggle-stuck SWE by forcing all SWEs to a cross state.

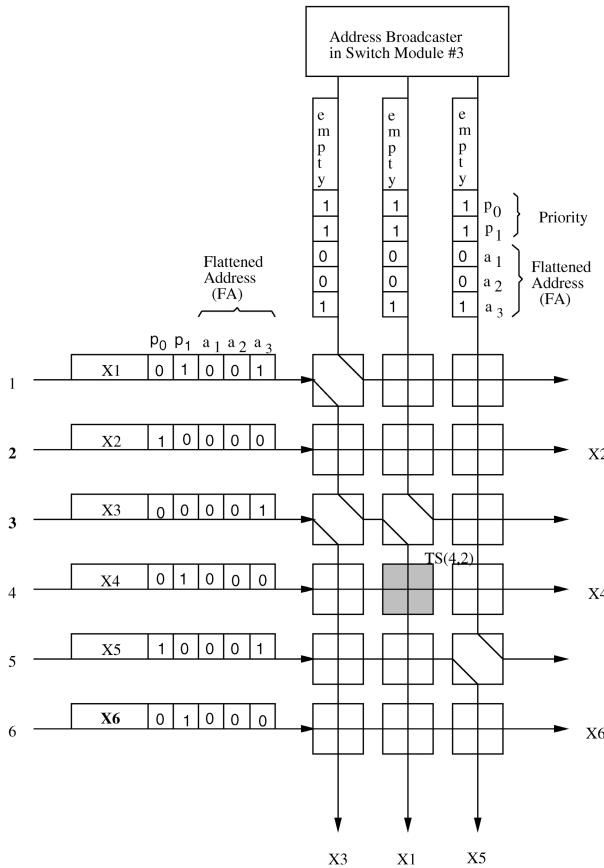


Fig. 6.27 Reconfiguration by forcing a toggle-stuck SWE(4, 2) to a cross state.

This reconfigured SWE array has the same cell-loss performance as the SWE array with a CS fault at (i, j) .

As mentioned before, a CS fault cannot be detected online. However, since a CS situation can be considered a reconfiguration of TS faults, we do not need to identify the location of the CS fault (although it can be detected and located with a few steps of offline tests as described in the following).

If we add one more bit to the priority field and modify the setting of the priority fields of the test cells, we can offline detect and locate a CS fault in the SWE array. The method of locating a CS fault is to force all the SWEs in the shaded square block of SWEs in Figure 6.28 to a toggle state while the rest of the SWEs are set to a cross state. If there exists a CS SWE in the square block, we will be able to identify its location by monitoring the outputs. By moving the square block around in the SWE array and repeating the test procedure, we can determine if there is a CS fault and locate its position (if any).

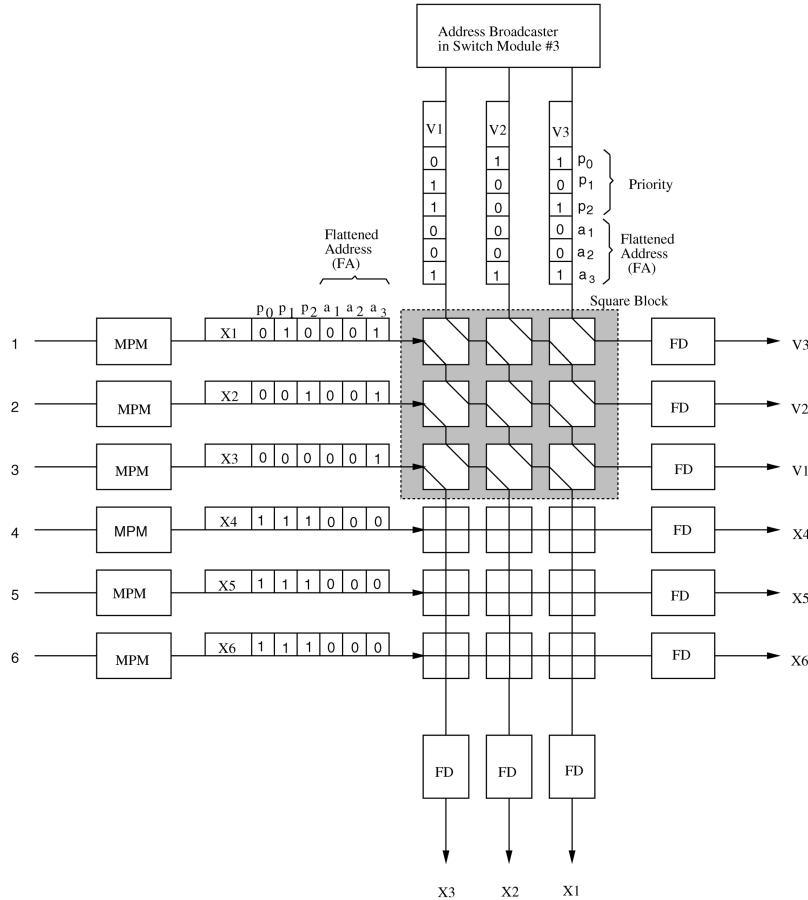


Fig. 6.28 Fault location test for a cross-stuck SWE by moving the square block around in the SWE array.

Figure 6.28 shows an example of the fault location test for a CS SWE (CS test). This test can also be used to locate a HS SWE. To diagnose the SWEs in the three uppermost rows, the priorities of the test cells, X3, X2, X1, V1, V2, and V3, are arranged in descending order. These cells' FAs are set to be identical, but the FAs of test cells X4, X5, and X6 are set to be different. The CS test forces all SWEs in the square block (3×3 in this example) to a toggle state and all other SWEs to a cross state. If there exists a CS in this square block, the output pattern will be different from the expected one. For offline testing of a switch module, at least $\lceil N/L_1M \rceil$ tests are required in MGN1, and $\lceil L_1M/L_2 \rceil$ tests in MGN2.

6.4.3.2 Vertical-Stuck and Horizontal-Stuck Cases If a VS or an HS fault exists in the SWE array, cells that pass through the faulty SWE will be corrupted as shown in Figure 6.24 and Figure 6.25. Once a VS fault is

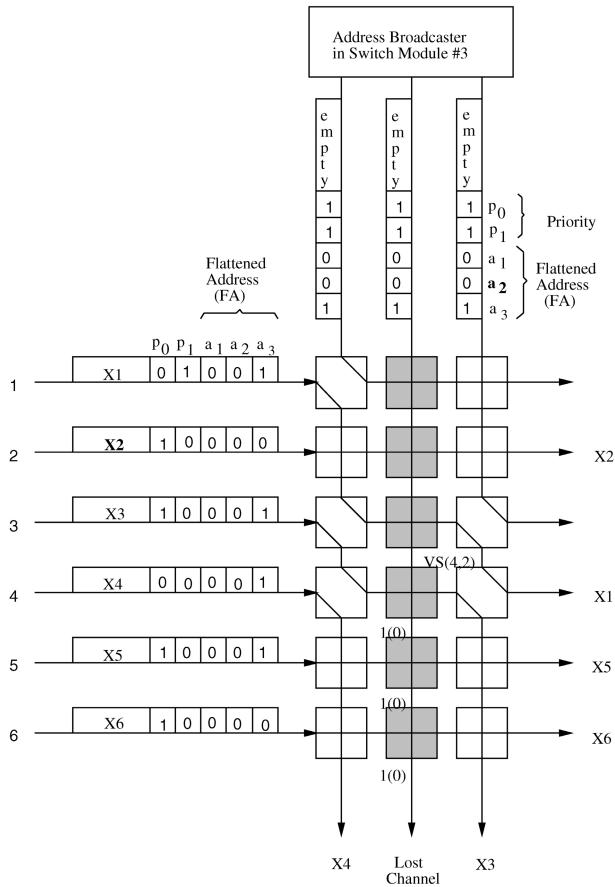


Fig. 6.29 Reconfiguration by isolating a faulty column caused by a vertical = stuck SWE(4, 2).

detected by the FD at MTTs or OPCs, the information about the faulty column is known immediately. This information is sufficient to reconfigure the switch network. For example, if the j th column contains a VS ($j = 2$ in Figure 6.24), all the SWEs in the j th column are forced to a cross state, so that the j th column is isolated. For instance, Figure 6.29 shows the reconfiguration with the second column isolated. Of course, the cell loss performance is degraded because of the reduction of available routing links.

Special attention is needed for a HS fault, because we must locate exactly the faulty SWE prior to proper reconfiguration. Once a HS fault is detected, we have the information about the faulty row automatically. What we need is the column information. In order to locate the fault, we define a fault location test (HS test) for it.

The HS test sets the test cells' FA, priority, and source address to appropriate values such that all SWEs on the faulty row are forced to a

toggle state. Let us consider an HS test for the i th row of the SWE array. The FA of test cells from all MPMs except the i th MPM are set to all zeros (for mismatching purposes), while the test cell's FA from the i th MPM is set to be identical to that of the test cells from the AB. All test cells' source addresses are set to their row numbers or column numbers, depending on whether they are fed to the SWE array horizontally or vertically. Priorities of the test cell from the i th MPM and broadcast test cells from AB are set in descending order. Normally, the priority of the test cell from the i th MPM is the highest, the broadcast test cell from the first column is the second highest, the broadcast test cell from the second column is the third highest, and so on.

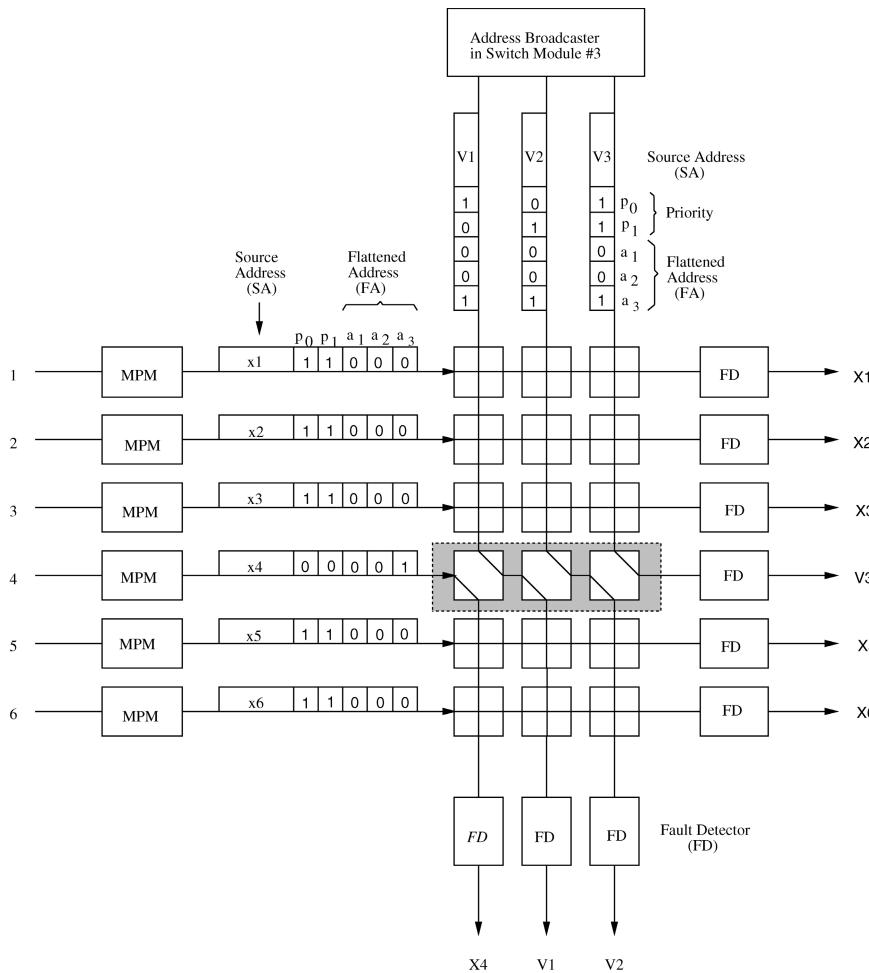


Fig. 6.30 Fault location test for a horizontal-stuck SWE by forcing the SWEs in the faulty row to a toggle state.

With this arrangement, the cell from the i th MPM will appear at the first output link, while the broadcast cell from the r th AB will appear at the $(r + 1)$ th output link or the i th discarding output. The FDs at the output links examine the source address SA field to locate the faulty column that contains the HS SWE. If the j th column contains a HS SWE, the test cell from the i th MPM will appear at the first output link, and the test cell V_r broadcast from the r th AB ($1 \leq r \leq j - 1$) will appear at the $(r + 1)$ th output link. The test cell V_j broadcast from the j th AB will be corrupted and appear at the i th discarding output. The test cell, V_{j+s} broadcast from the $(j + s)$ th AB ($1 \leq s \leq L_1 M - j$) will appear at the $(j + s)$ th output link. If a HS SWE is on the last column, the test cell from the r th AB will appear at the $(r + 1)$ th output link, and the test cell broadcast from the rightmost AB appears at the i th discarding output and is corrupted.

Figure 6.30 shows a test example where a HS SWE exists in the fourth row and all SWEs in the fourth row are forced to a toggle state to locate the HS

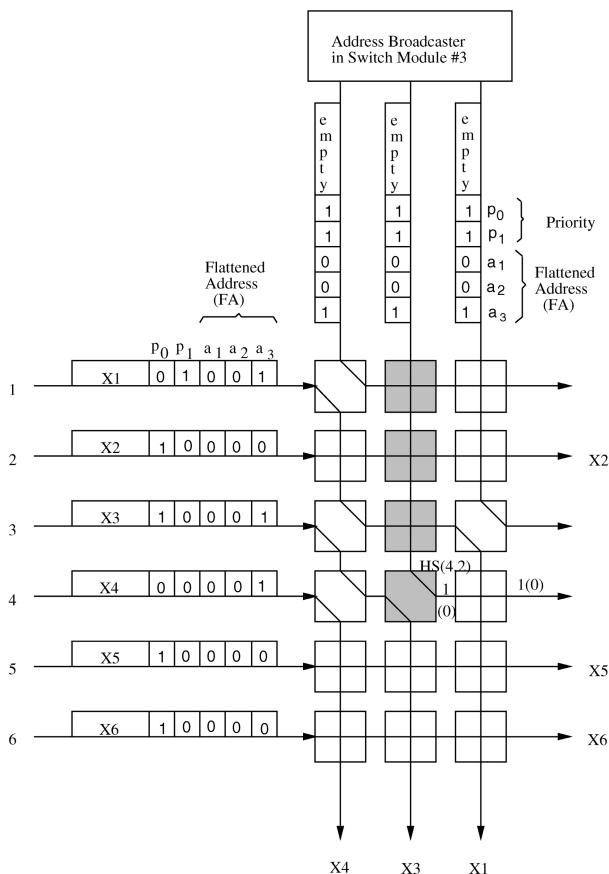


Fig. 6.31 Reconfiguration by isolating the upper column of the horizontal-stuck SWE(4, 2).

SWE. For example, if SWE(4, 2) is HS, cell V3 (instead of cell V2) will appear at output link 3; cell V2, which is corrupted, will appear at the 4th discarding output.

After locating the HS fault, say at SWE(i, j), we can reconfigure the SWE array by forcing SWE(i, j) to a toggle state and the switch elements that are in the same column and are above the faulty SWE [SWE(k, j), $1 \leq k \leq i - 1$] to a cross state, so that cells will not be affected by the HS SWE(i, j). Figure 6.31 shows an example of the reconfiguration for a HS at SWE(4, 2).

6.4.4 Performance Analysis of Reconfigured Switch Module

In this subsection, we show the graceful cell loss performance degradation of a SM under faulty conditions. The cell loss rate of each input under different faults is calculated. For simplicity, it is assumed that the SM has an $N \times L$ SWE array, that all incoming cells have the same priority and have a higher priority than broadcast cells from the AB, and that the traffic at each input port of the MOBAS is independent of that at the other inputs. The average cell arrival rate, ρ , is the probability that a cell arrives at an input port in a given time slot. It is also assumed that cells are uniformly delivered to all output ports and that there is only point-to-point communication, which gives the worst-case performance analysis. As a result, the traffic at each input of the SM has an identical Bernoulli distribution with ρ/N as its average cell arrival rate.

$P_L(k)$ is defined as the cell loss rate of the k th input with an expansion ratio L under a fault-free condition. The cell loss rates of the uppermost L inputs are zero, meaning that cells from these inputs are always delivered successfully, or

$$P_L(k) = 0, \quad 1 \leq k \leq L.$$

The cell loss rates of the remaining inputs, from input $L + 1$ to input N , are

$$\begin{aligned} P_L(k) &= \Pr \left\{ \begin{array}{l|l} \text{at least } L \text{ inputs above} & \text{the } k \text{th input has} \\ \text{the } k \text{th have active cells} & \text{an active cell} \end{array} \right\} \\ &= \Pr \left\{ \begin{array}{l} \text{at least } L \text{ active cells arrive} \\ \text{at inputs 1 to } k - 1 \end{array} \right\} \\ &= \sum_{i=L}^{k-1} \binom{k-1}{i} \left(\frac{\rho}{N} \right)^i \left(1 - \frac{\rho}{N} \right)^{k-1-i} \\ &= 1 - \sum_{i=0}^{L-1} \binom{k-1}{i} \left(\frac{\rho}{N} \right)^i \left(1 - \frac{\rho}{N} \right)^{k-1-i}, \quad L + 1 \leq k \leq N. \end{aligned}$$

$P'_L(k)$ is defined as the cell loss rate of the k th input with an expansion ratio L under a faulty SWE. Note that the cell loss rate discussed here is not for the entire MOBAS, but only for the SM that has a faulty SWE.

6.4.4.1 Cross-Stuck and Toggle-Stuck Cases As discussed before, a TS fault can be isolated by setting the TS SWE to a cross state, which resembles the occurrence of the CS fault at this SWE (see Fig. 6.27). Thus, the effect on cell loss performance of the CS and TS faults is the same. We will analyze only the cell loss performance of the SM with a CS SWE.

If the CS fault of $\text{SWE}(i, j)$ is not in the last column ($j \neq L$), there is no performance degradation. When it is in the last column ($j = L$), the CS SWE does not affect the cell loss performance of the inputs except for the one in the same row of the faulty SWE.

For a fault $\text{CS}(i, L)$ ($i \geq L$), only one input arrival pattern contributes to the cell loss performance degradation. That is when there are exactly L cells destined for this output port. Among these cells are $L - 1$ cells from inputs 1 to $i - 1$; one cell from input i , and no cells from the inputs $i + 1$ to N ($L \leq i \leq N$). The average number of additionally lost cells, $\text{AL}(i)$, due to the faulty $\text{SWE}(i, j)$ depends on the fault row position i :

$$\begin{aligned} \text{AL}(i) &= \binom{i-1}{L-1} \left(\frac{\rho}{N}\right)^{L-1} \left(1 - \frac{\rho}{N}\right)^{i-L} \frac{\rho}{N} \binom{N-i}{0} \left(\frac{\rho}{N}\right)^0 \left(1 - \frac{\rho}{N}\right)^{N-i} \\ &= \binom{i-1}{L-1} \left(\frac{\rho}{N}\right)^L \left(1 - \frac{\rho}{N}\right)^{N-L}, \quad L+1 \leq i \leq N. \end{aligned}$$

Note that a faulty $\text{SWE}(i, L)$ ($i \leq L - 1$) does not introduce any cell loss performance degradation. Under the cross-state reconfiguration for a faulty $\text{SWE}(i, L)$, the cell loss rate of each input is as follows:

$$P'_L(k) = \begin{cases} P_L(k), & 1 \leq k \leq i-1, \\ P_{L-1}(k), & k = i, \\ P_L(k) - P(A), & i+1 \leq k \leq N. \end{cases}$$

where $P(A)$ is the probability of the event A ,

$$\begin{aligned} A &= \left\{ \begin{array}{l|l} \text{the } k\text{th input's active cell} & L \text{ inputs, located above the} \\ \text{is routed successfully} & k\text{th input, have active cells} \end{array} \right\} \\ P(A) &= \binom{i-1}{L-1} \left(\frac{\rho}{N}\right)^{L-1} \left(1 - \frac{\rho}{N}\right)^{i-L} \frac{\rho}{N} \binom{k-i}{0} \left(\frac{\rho}{N}\right)^0 \left(1 - \frac{\rho}{N}\right)^{k-i} \\ &= \binom{i-1}{L-1} \left(\frac{\rho}{N}\right)^L \left(1 - \frac{\rho}{N}\right)^{k-L}, \quad i < k \leq N. \end{aligned}$$

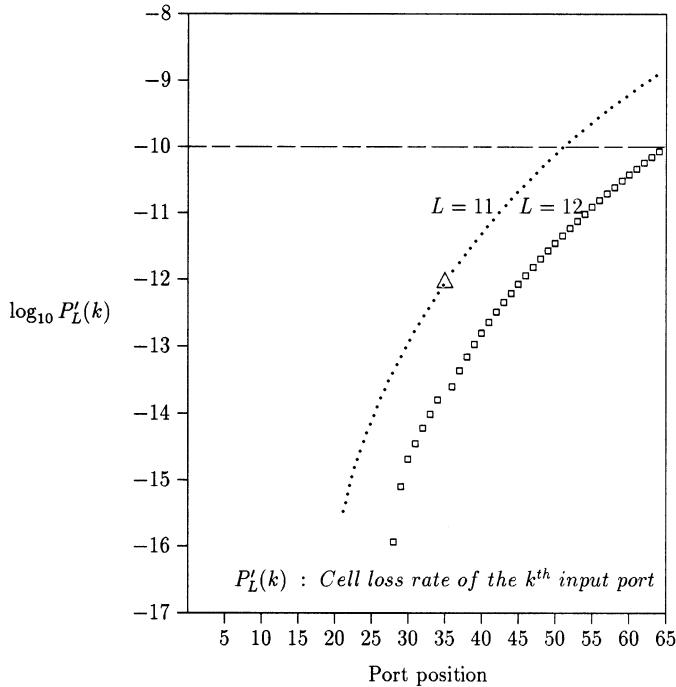


Fig. 6.32 Cell loss rate of each input port after cross-reconfiguration. (©1994 IEEE.)

The probability $P(A)$ is considered as the cell loss rate improvement of the k th input

Figure 6.32 shows the cell loss rate of each input after the SWE array is reconfigured. Here, the MOBAS is assumed to be a single stage with a size of 64×64 and an expansion ratio L of 12. The faulty SWE is assumed to be SWE(35, 12). For those inputs above the faulty input (35), the cell loss rates are the same as in the fault-free case, while the faulty input's cell loss rate is equal to that of the $L = 11$ fault-free case. The inputs below the faulty one (from 36 to 64) have almost the same cell loss rates as in the fault-free case, since the improvement $P(A)$ is very small.

6.4.4.2 Vertical-Stuck Case When there is a VS SWE in the j th column, all the SWEs in that column are forced to a cross state to isolate the j th column, as shown in Figure 6.29 ($j = 2$). This column isolation prevents cells from being corrupted, but reduces available routing links from L to $L - 1$. Therefore, after the reconfiguration, the cell loss rate of each input is slightly increased to the following:

$$P'_L(k) = P_{L-1}(k), \quad 1 \leq k \leq N.$$

Note that the effect on the cell loss performance degradation is independent of the faulty link's position. The cell loss rates after the reconfiguration are

equal to those of the fault-free $L = 11$ case, regardless of the row position of the faulty link.

6.4.4.3 Horizontal-Stuck SWE Case Figure 6.31 shows a reconfiguration example of HS SWE(4, 2). Let us consider an $\text{HS}(i, j)$ case: the cell loss rates of all inputs above the i th one are equal to those of the fault-free ($L - 1$) case, while the i th input has no cell loss. And the cell loss rates of all inputs below the i th one are almost equal to those of the fault-free ($L - 1$) case. Incoming cells from all inputs above the i th one can only be routed to $L - 1$ output links, and the i th input always occupies one of j output links (from the first to the j th).

Special attention is needed when we consider the cell loss rates of the inputs below the i th row. The cell loss rate of the k th input ($i < k \leq N$) depends only on $k - 2$ inputs, which are all above the k th one and exclude the i th input. Thus, the cell loss rate of the k th input is equal to that of the $(k - 1)$ th input in the case where the switch module is fault-free and its expansion ratio is $L - 1$:

$$P'_L(k) = \begin{cases} P_{L-1}(k), & 1 \leq k < i - 1, \\ 0, & k = i, \\ P_{L-1}(k-1), & i + 1 \leq k \leq N. \end{cases}$$

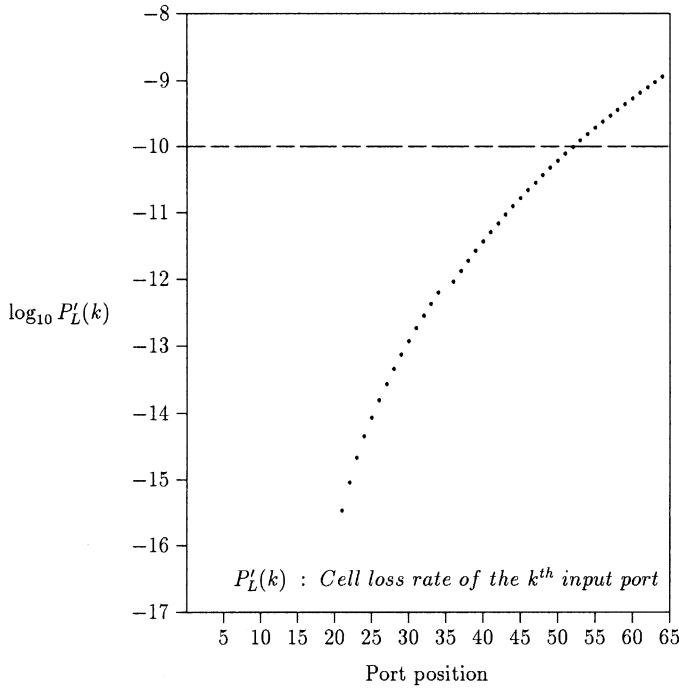


Fig. 6.33 Cell loss rate of each input port after partial column isolation.

Figure 6.33 shows the cell loss rates of inputs of the switch module with a HS SWE. The cell loss rates of all inputs above the faulty one ($i = 35$) are equal to those of fault-free $L = 11$ case, while the faulty input has no cell loss. The cell loss rates of the inputs below the faulty one are almost equal to those of the fault-free $L = 11$ case, because one of j output links is always occupied by the cells from the faulty input, and they are independent of the faulty input's cell arrival pattern.

From the above analysis it can be concluded that under the single-fault condition, the worst cell loss performance degradation corresponds to the loss of one output link. Note that these cell loss rates are not the average rate of the entire MOBAS but that of the SM with a faulty SWE. When an input experiences cell loss performance degradation due to a faulty SWE, it corresponds to one output link loss in the worst case. Thus, if we add one more column of SWEs for each switch module, which corresponds to less than 2% hardware overhead in MGN1 and less than 9% hardware overhead in MGN2, under any kind of single-fault condition, the MOBAS can still have as good cell loss performance as the originally designed, fault-free system.

6.5 APPENDIX

Let us consider an ATM switch as shown in Figure 6.10, and assume that cells arrive independently from different input ports and are uniformly delivered to all output ports. The variables used are defined as follows.

N : the number of a switch's input ports or output ports

M : the number of output ports that are in the same group

L : the group expansion ratio

ρ : the offered load of each input port, or the average number of cells that arrive at the input port in each cell time slot

ρ/N : the average number of cells from each input port destined for an output port in each cell time slot

$\rho M/N$: the number of cells from each input port destined for an output group in each cell time slot

P_k : the probability that k cells arrive at an output group in each cell time slot

λ : the average number of cells from all input ports that are destined for an output group in each cell time slot

λ' : the average number for cells from all input ports that have arrived at an output group in each cell time slot

Then P_k is given by the following binomial probability:

$$P_k = \binom{N}{k} \left(\frac{\rho M}{N} \right)^k \left(1 - \frac{\rho M}{N} \right)^{N-k}, \quad k = 0, 1, \dots, N,$$

and we have

$$\begin{aligned} \lambda &= \sum_{k=1}^N k P_k = N \left(\frac{\rho M}{N} \right) = \rho M, \\ \lambda' &= \sum_{k=1}^{LM} k P_k + \sum_{k=LM+1}^N (LM) P_k \\ &= \lambda - \sum_{k=LM+1}^N k P_k + \sum_{k=LM+1}^N (LM) P_k \\ &= \lambda - \sum_{k=LM+1}^N (k - LM) P_k. \end{aligned}$$

Since at most LM cells are sent to each output group in each cell time slot, the excess cells will be discarded and lost. The cell loss probability is

$$\begin{aligned} P(\text{cell loss}) &= \frac{\lambda - \lambda'}{\lambda} \\ &= \frac{1}{\lambda} \sum_{k=LM+1}^N (k - LM) P_k \\ &= \frac{1}{\rho M} \sum_{k=LM+1}^N (k - LM) \binom{N}{k} \left(\frac{\rho M}{N} \right)^k \left(1 - \frac{\rho M}{N} \right)^{N-k}. \quad (6.15) \end{aligned}$$

As $N \rightarrow \infty$,

$$\begin{aligned} P_k &= \frac{(\rho M)^k e^{-\rho M}}{k!}, \\ P(\text{cell loss}) &= \frac{1}{\rho M} \sum_{k=LM+1}^{\infty} (k - LM) \frac{(\rho M)^k e^{-\rho M}}{k!} \\ &= \sum_{k=LM+1}^{\infty} \frac{(k - LM)}{\rho M} \frac{(\rho M)^k e^{-\rho M}}{k!} \end{aligned}$$

$$\begin{aligned}
&= \sum_{k=LM+1}^{\infty} \frac{(\rho M)^{k-1} e^{-\rho M}}{(k-1)!} - \frac{L}{\rho} \sum_{k=LM+1}^{\infty} \frac{(\rho M)^k e^{-\rho M}}{k!} \\
&= \sum_{k=LM}^{\infty} \frac{(\rho M)^k e^{-\rho M}}{k!} - \frac{L}{\rho} \sum_{k=LM+1}^{\infty} \frac{(\rho M)^k e^{-\rho M}}{k!} \\
&= \frac{(\rho M)^{LM} e^{-\rho M}}{(LM)!} + \sum_{k=LM+1}^{\infty} \frac{(\rho M)^k e^{-\rho M}}{k!} \\
&\quad - \frac{L}{\rho} \sum_{k=LM+1}^{\infty} \frac{(\rho M)^k e^{-\rho M}}{k!} \\
&= \left(1 - \frac{L}{\rho}\right) \left(\sum_{k=LM+1}^{\infty} \frac{(\rho M)^k e^{-\rho M}}{k!} \right) + \frac{(\rho M)^{LM} e^{-\rho M}}{(LM)!} \\
&= \left(1 - \frac{L}{\rho}\right) \left(1 - \sum_{k=0}^{LM} \frac{(\rho M)^k e^{-\rho M}}{k!}\right) + \frac{(\rho M)^{LM} e^{-\rho M}}{(LM)!}. \quad (6.16)
\end{aligned}$$

REFERENCES

1. G. B. Adams III, D. P. Agrawal, and H. J. Siegel, "A survey and comparison of fault-tolerant multistage interconnection networks," *IEEE Computer Mag.*, vol. 36, pp. 14–27, Jun. 1987.
2. D. P. Agrawal, "Testing and fault tolerance of multistage interconnection networks," *IEEE Computer*, vol. 15, pp. 41–53, Apr. 1982.
3. H. J. Chao, "A recursive modular terabit/second ATM switch," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1161–1172, Oct. 1991.
4. H. J. Chao and B. S. Choe, "Design and analysis of a large-scale multicast output buffered ATM switch," *IEEE/ACM Trans. Networking*, vol. 3, no. 2, pp. 126–138, Apr. 1995.
5. B. S. Choe and H. J. Chao, "Fault-tolerance of a large-scale multicast output buffered ATM switch," *IEEE Proc. INFOCOM '94*, Toronto, Canada, Jun. 1994.
6. B. S. Choe and H. J. Chao, "Fault tolerance of a large-scale multicast output buffered ATM switch," *Proc. IEEE INFOCOM '94*, Toronto, Canada, pp. 1456–1464, Jun. 1994.
7. K. Y. Eng, M. J. Karol, and Y.-S. Yeh, "A growable packet (ATM) switch architecture: design principles and applications," *IEEE Trans. Commun.*, vol. 40, no. 2, pp. 423–430, Feb. 1992.
8. M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1587–1597, Dec. 1988.
9. W. L. Hoberecht, "A layered network protocol for packet voice and data integration," *IEEE J. Select. Areas Commun.*, vol. SAC-1, pp. 1006–1013, Dec. 1983.

10. W. Hoeffding, "On the distribution of the number of successes in independent trials," *Ann. Math. Statist.*, vol. 27, pp. 713–721, 1956.
11. A. Itoh, "A fault-tolerant switching network for B-ISDN," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1218–1226, Oct. 1991.
12. M. Jeng and H. J. Siegel, "Design and analysis of dynamic redundancy networks," *IEEE Trans. Comput.*, vol. 37, no. 9, pp. 1019–1029, Sep. 1988.
13. M. J. Karol and C.-L. I, "Performance analysis of a growable architecture for broadband packet (ATM) switching," *Proc. IEEE GLOBECOM '89*, pp. 1173–1180, Nov. 1989.
14. D. F. Kuhl, "Error recovery protocols: link by link vs edge to edge," *Proc. IEEE INFOCOM '83*, pp. 319–324, Apr. 1983.
15. V. P. Kumar and A. L. Reibman, "Failure dependent performance analysis of a fault-tolerant multistage interconnection network," *IEEE Trans. Comput.*, vol. 38, no. 12, pp. 1703–1713, Dec. 1989.
16. V. P. Kumar and S. J. Wang, "Reliability enhancement by time and space redundancy in multistage interconnection networks," *IEEE Trans. Reliability*, vol. 40, no. 4, pp. 461–473, Oct. 1991.
17. T. H. Lee and J. J. Chou, "Fault tolerance of banyan using multiple-pass," *Proc. INFOCOM '92*, Florence, Italy, May 1992.
18. T. T. Lee, "Non-blocking copy networks for multicast packet switching," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1455–1467, Dec. 1988.
19. T. T. Lee, "A modular architecture for very large packet switches," *IEEE Trans. Commun.*, vol. 38, no. 7, pp. 1097–1106, Jul. 1990.
20. S. C. Liew and K. W. Lu, "Performance analysis of asymmetric packet switch modules with channel grouping," *Proc. IEEE INFOCOM '90*, pp. 668–676.
21. R. H. Lin, C. H. Lam, and T. T. Lee, "Performance and complexity of multicast cross-path ATM switches," *Proc. IEEE INFOCOM '97*, Apr. 1997.
22. Y. F. Lin and C. B. Shung, "Fault-tolerant architectures for shared buffer memory switch," *IEEE Int. Symp. on Circuits and Systems*, vol. 4, pp. 61–64, 1994.
23. Y. Oie, M. Murata, K. Kubota, and H. Miyahara, "Effect of speedup in nonblocking packet switch," *Proc. IEEE ICC '89*, pp. 410–415.
24. A. Pattavina, "Multichannel bandwidth allocation in a broadband packet switch," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1489–1499, Dec. 1988.
25. A. Pattavina and G. Bruzzi, "Analysis of input and output queueing for nonblocking ATM switches," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 314–328, Jun. 1993.
26. C. L. Tarn and J. S. Meditch, "A high performance copy network for B-ISDN," *Proc. IEEE INFOCOM '91*, pp. 171–180, Apr. 1991.
27. S. C. Yang and J. A. Sylvester, "A fault tolerant reconfigurable ATM switch fabric," *Proc. IEEE INFOCOM '91*, pp. 1237–1244, 1991.
28. Y.-S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: a simple, modular architecture for high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. 5, no. 8, pp. 1274–1283, Oct. 1987.
29. A. Varma and S. Chalasani, "Fault-tolerance analysis of one-sided crosspoint switching networks," *IEEE Trans. Comput.*, vol. 41, no. 2, pp. 143–158, Feb. 1992.

CHAPTER 7

THE ABACUS SWITCH

The switches based on the knockout concept suffer from cell loss due to the lack of routing links in the switch fabric—for example, the concentrator in the knockout switch, or the multicast grouping network (MGN) in the MOBAS in Chapter 6. Although we can engineer the group expansion ratio L to achieve a satisfactory cell loss probability, say 10^{-10} , that is based on the assumption that the traffic from different input ports is uncorrelated and input traffic is uniformly distributed to all output ports. The latter assumption gives the worst case cell loss probability, while the former assumption may not be realistic for the applications such as Internet Web services. There may be a lot of traffic destined for the same popular site at the same time, resulting in a so-called hot-spot situation and an unacceptable cell loss probability. In order to reduce the cell loss rate, excess cells can be stored at the input buffers, which results in the switch having buffers at the input and output ports. The switch to be discussed in this chapter belongs to this category.

We describe a switch that has a similar architecture to the MOBAS but does not discard cells in the switch fabric. When the head-of-line (HOL) cells of the input ports are sent to the switch fabric, they are held at the input ports until they have been successfully transmitted to the desired output port. The switch fabric is a crossbar structure, where switch elements, with the capability of routing cells and resolving contention based on cells' priority levels, are arranged in a two-dimensional array, and is similar to an abacus. For this reason it is called the *abacus switch*. The challenging issue of designing an input–output-buffered switch is to design a fast and scalable arbitration scheme.

The arbitration algorithm proposed in the abacus switch takes advantage of the switch element's ability to resolve contention for the routing links according to their priority levels. As a result, with some extra feedback lines and logic circuits at the input ports, the arbitration scheme can be implemented without adding much complexity and cost. The switch uses a new arbitration scheme to resolve the contention among the HOL cells. The arbitration is done in a distributed manner and thus enables the switch to grow to a large size.

Section 7.1 describes the basic architecture of the abacus switch. Section 7.2 presents the new arbitration scheme, which is implemented in a distributed manner. Section 7.3 depicts the implementation of an input controller and how it resolves contention resolution. Section 7.4 discusses the performance of the abacus in throughput, delay, and loss. Section 7.5 shows a key component, the ATM routing and concentration (ARC) chip used to implement the abacus switch. Section 7.6 describes three approaches to scale the abacus switch to 1-Tbit/s capacity. Section 7.7 shows how the abacus switch can also route switch packets through the switch fabric.

7.1 BASIC ARCHITECTURE

The abacus switch is a scalable multicast architecture with input and output buffering. It uses input buffers to temporarily store cells that have lost contention to other inputs and thus eliminates the possibility of discarding cells due to the loss of contention in the switch fabric, as in the MOBAS.

Figure 7.1 shows the architecture of the abacus switch. It consists of input port controllers (IPCs), a multicast grouping network (MGN), multicast translation tables (MTTs), small switch modules (SSMs), and output port controllers (OPCs). The architecture is very similar to the MOBAS's except that MGN2 in the MOBAS is now replaced with the SSM and that the abacus switch has feedback lines from the routing modules (RMs) to the IPCs to facilitate output port contention resolution (see details in Section 7.2). The RM in the abacus switch is exactly the same as the SM in the MOBAS, but the term "RM" will be used from now on. If the group size M is carefully chosen in such a way that the second-stage switch network's capacity is (say) at most 20 Gbit/s, it will be more cost-effective to implement the MGN2 in the MOBAS with a shared-memory switch module. For instance, for $M = 32$, $L = 2$, and line rate 155.52 Mbit/s, the SSM's capacity is 10 Gbit/s. The IPC performs similar functions to those in the MOBAS, except that it also assists in resolving contention among cells that are destined to the same output group and buffering those cells losing contention.

The switch performs cell replication and cell routing simultaneously. Cell replication is achieved by broadcasting incoming cells to all RMs, which then selectively route cells to their output links. Cell routing is performed distributedly by an array of switch elements (SWEs). The concept of channel

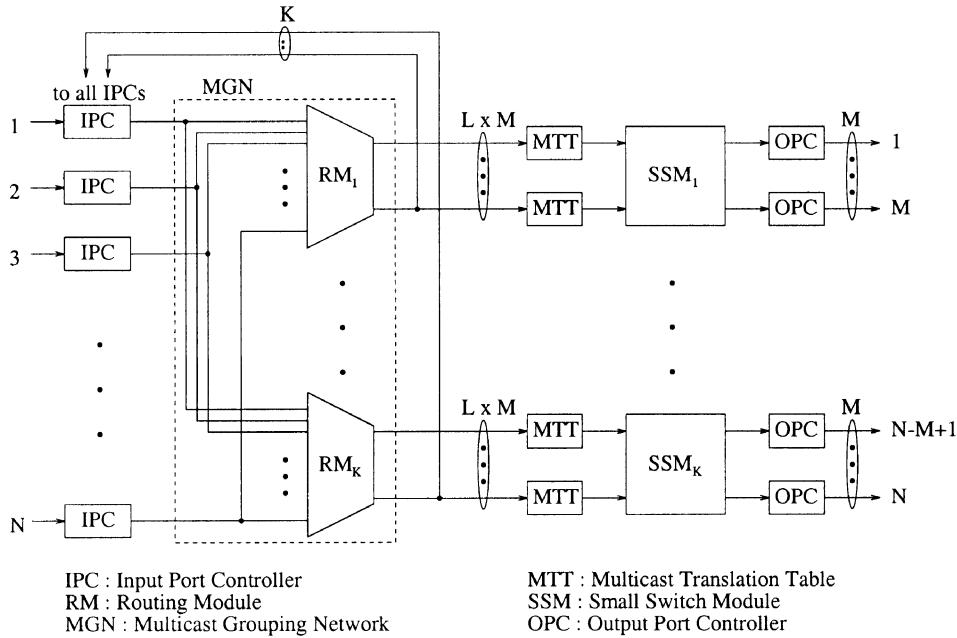


Fig. 7.1 The architecture of the abacus switch. (©1997 IEEE.)

grouping described in Section 6.2 is applied to construct the MGN in order to reduce hardware complexity, where every M output ports are bundled in a group. For a switch size of N input ports and N output ports, there are K output groups ($K = N/M$). The MGN consists of K routing modules; each of them provides LM routing links to each output group. L is defined as group expansion ratio: the ratio of the required number of routing links to the group size. Cells from the same virtual connection can be arbitrarily routed to any one of the LM routing links, and their sequence integrity will be maintained. Based on an arbitration mechanism to be described in Section 7.2, up to LM cells from N IPCs can be chosen in each RM. Cells that lose contention are temporarily stored in an input buffer and will retry in the next time slot. On the other hand, cells that are successfully routed through RMs will be further routed to proper output port(s) through the SSMs.

The group expansion ratio L is engineered in such a way that the required maximum throughput in a switch fabric can be achieved. Performance study shows that the larger M is, the smaller is the L required to achieve the same maximum throughput. For instance, for a group size M of 16 and input traffic with an average burst length of 15 cells, L has to be at least 1.25 to achieve a maximum throughput of 0.96. But, for a group size M of 32 and the same input traffic characteristic, L can be as low as 1.125 to achieve the same throughput. Since cell loss doesn't occur within the abacus switch (unlike the MOBAS), L is chosen to achieve sufficiently large maximum

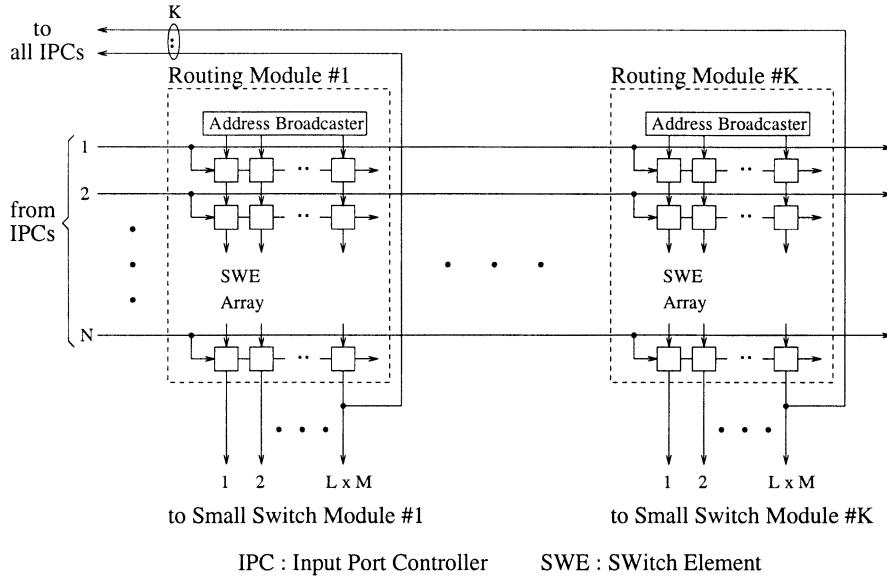


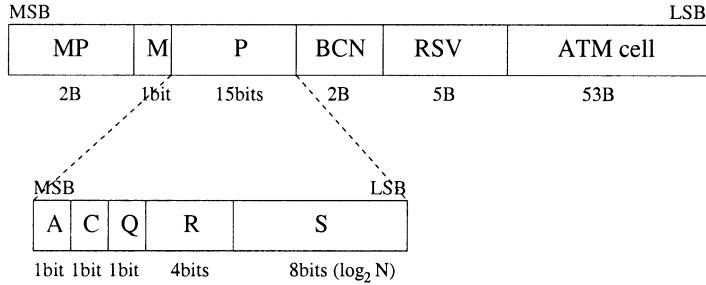
Fig. 7.2 The multicast grouping network (MGN). (©1997 IEEE.)

throughput and low delay in the input buffers, but not for cell loss rate as in the MOBAS. Its value can be slightly smaller than the one in the MOBAS (e.g., for $M = 32$, L is 2 for a cell loss rate of 10^{-10}).

Each RM in the MGN contains a two-dimensional array of switch elements and an address broadcaster (AB), as shown in Figure 7.2. It is similar to Figure 6.15 except that each RM provides a feedback line to all IPCs. The multicast pattern maskers (MPMs) are not shown here for simplicity.

Figure 7.3 shows routing information for a multicast ATM switch with $N = 256$ and $M = 16$, which consists of several fields: a multicast pattern (MP), a priority field (P), and a broadcast channel number (BCN). A MP is a bit map of all the output groups and is used in the MGN for routing cells to multiple output groups. Each bit indicates if the cell is to be sent to the associated output group. For instance, if the i th bit in the MP is set to 1, the cell is to be sent to the i th output group. The MP has K bits for an MGN that has K output groups (16 in this example). For a unicast call, its multicast pattern is basically a flattened output address (i.e., a decoded output address) in which only one bit is set to 1 and all the other $K - 1$ bits are set to 0. For a multicast call, there is more than one bit set to 1 in the MP, corresponding to the output groups for which the cell is destined.

A priority field (P), used to assist contention resolution, can be flexibly set to any value to achieve desired service preference. For instance, the priority field may consist of an activity bit (A), a connection priority (C), a buffer state priority (Q), a retry priority (R), and an input port priority (S). Let us



MP : Multicast Pattern	S: Input port priority
P: Priority for contention resolution	M : Multicast bit
A: Activity bit	BCN : Broadcast Channel Number
C : Connection priority	RSV : Reserved field for future growth
Q : Buffer state priority	MSB: Most Significant Bit
R: Retry priority	LSB: Least Significant Bit

Fig. 7.3 Routing information used by the abacus switch with $N = 256$, $M = 16$.

assume the smaller the priority value, the higher the priority level. The activity bit (A) indicates the validity of the cell. It is set to 0 if the cell is valid and set to 1 otherwise. The connection priority (C) indicates the priority of the virtual connection, which can be determined during the call setup or service provisioning. The buffer state priority (Q) provides for sharing among N input buffers by allowing the HOL cell in an almost overflowed buffer (e.g., exceeding a predetermined threshold) to be transmitted sooner, so that the overall cell loss probability is reduced. The retry priority (R) provides global first-come, first-served (FCFS) discipline, allowing a cell's priority level to move up by one whenever it loses contention once. The retry priority can initially be set to 1111 and decreased by one whenever losing contention once. In order to achieve fairness among input ports, the priority levels of the HOL cells at the input ports dynamically change at each time slot. The input port priority (S) can initially be set to its input port address with $\log_2 N$ bits and decreased by one at every time slot, thus achieving round-robin (RR) fairness.

The BCN in Figure 7.3 will be used to find a new multicast pattern in the MTT, allowing the copied cell to be further duplicated in the SSM. The BCN will also be used by the OPC to find a new VPI/VCI for each copy of the replicated cell.

7.2 MULTICAST CONTENTION RESOLUTION ALGORITHM

Here, we describe a novel algorithm that resolves output port contention among the input ports in a fair manner. It can also perform call splitting for multicasting and thus improves the system throughput. The output port

contention resolution is often implemented by a device called an *arbiter*. Most proposed arbiters can only handle unicast calls (i.e., point-to-point communication) and N -to-1 selection—for example, three-phase [10], ring reservation [1], and centralized contention resolution devices [6].

Implementing an arbiter capable of handling call splitting and N -to-multiple selection is much more challenging with respect to the timing constraint. At the beginning of the cell time slot, the arbiter receives N multicast patterns, one from each input port, and returns acknowledgment to those input ports whose HOL cells have won contention. These cells are then allowed to transmit to the switch fabric. Let us consider these N multicast patterns, each with K bits, being stacked up; there are K columns with N bits in each column. Each column associates with each output group. The arbiter's job is to select up to, for example, LM bits that are set to 1 from each column and perform that operation K times, all within one cell time slot. In other words, the arbitration's timing complexity is $O(NK)$. The arbiter may become the system's bottleneck when N or K is large.

The arbitration scheme described here performs N -to- LM selection in a distributed manner using the switch fabric and all IPCs, thus eliminating the speed constraint. Another difference between this arbitration scheme and others is that here the HOL cell is repeatedly sent to the switch fabric to compete with others until it has successfully transmitted to all necessary output groups that the cell is destined for. Unlike other arbitration schemes, the scheme described here does not wait for an acknowledgment before transmitting the cell. When a cell is routed in a switch fabric without waiting for an acknowledgment, two situations are possible. It may be successfully routed to all necessary output groups, or only routed to a subset of the output groups—possibly the empty set. The last case is considered a failure, and the HOL cell will retry in the next time slot. When a cell is transmitted to the switch fabric, since it does not know if it will succeed, it must be stored in a one-cell buffer for possible retransmission.

Now the question is how the IPC knows whether or not its HOL cell has been successfully transmitted to all necessary output groups. In the abacus switch, the RMs are responsible for returning the routing results to the IPC. One possible way is to let each RM inform IPCs of the identification (e.g., the broadcast channel number) of the cells that have been successfully routed. However, since a cell can be routed to multiple output groups (for instance, up to K output groups for a broadcast situation), one IPC may receive up to K acknowledgments from K RMs. The complexity of returning the identification of every successfully routed copy to all IPCs is too high to be practical for a large-scale switch. A scheme that significantly simplifies the complexity of the acknowledgment operation is described in the following.

The RM can not only route cells to proper output groups, but also, based on cells' priority levels, choose up to LM cells that are destined for the same output group. The HOL cell of each input port is assigned a *unique* priority level that is different from the others. After cells are routed through an RM,

they are sorted at the output links of the RM according to their priority levels from left to right in descending order (See Fig. 7.2). The cell that appears at the rightmost output link has the lowest priority among the cells that have been routed through this RM. This lowest-priority information is broadcast to all IPCs. Each IPC will then compare the local priority (LP) of the HOL cell with a feedback priority, say FP_j , to determine if the HOL cell has been routed through RM_j . Note that there are K feedback priorities, FP_1, \dots, FP_K . If the feedback priority level (FP_j) is lower than or equal to the local priority level (LP), the IPC determines that its HOL cell has reached one of the output links of RM_j . Otherwise, the HOL cell must have been discarded in RM_j due to loss of contention and will be retransmitted in the next time slot. Since there are K RMs in total, there will be K lines broadcast from K RMs to all IPCs, each carrying the lowest-priority information in its output group.

The priority assigned to the HOL cells will be dynamically changed according to some arbitration policies, such as random, RR, state-dependent, and delay-dependent [8]. The random scheme randomly chooses the HOL cells of input ports for transmission; the drawback is it has a large delay variation. The RR scheme chooses HOL cells from input ports in a RR fashion by dynamically changing the scanning point from the top to the bottom input port (e.g., the S field in Figure 7.3). The state-dependent scheme chooses the HOL cell in the longest input queue so that input queue lengths are maintained nearly equal, achieving input buffer sharing (e.g., the Q field in Figure 7.3). The delay-dependent scheme performs like a global FIFO, where the oldest HOL cell has the highest priority to be transmitted to the output (e.g., the R field in Fig. 7.3). Since the arbitration is performed in a distributed manner by K RMs and in parallel by IPCs, any of the above policies, or a combination of them, can be implemented by arbitrarily assigning a proper priority level to the HOL cell.

At the beginning of the time slot, each IPC sends its HOL cell to the MGN. Meanwhile, the HOL cell is temporarily stored in a one-cell-size buffer during its transmission. After cells have traversed through the RMs, priority information, FP_1 to FP_K (the priority of the rightmost link of each RM), is fed back to every IPC. Each IPC will then compare the feedback priority level FP_j , $j = 1, 2, \dots, K$, with its local priority level, LP. Three situations can happen. First, $MP_j = 1$ and $LP \leq FP_j$ (recall that the smaller the priority value, the higher the priority level), which means the HOL cell is destined for the j th output group and has been successfully routed through the j th RM. The MP_j bit is then set to 0. Second, $MP_j = 1$ and $LP > FP_j$, which means the HOL cell is destined for the j th output group but discarded in the j th RM. The MP_j bit remains 1. Third, $MP_j = 0$ (the j th bit of the HOL cell's multicast pattern can be equal to 0), which means the HOL cell is not destined for the j th output group. Then, the MP_j bit remains 0.

After all MP_j bits ($j = 1, 2, \dots, K$) have been updated according to one of the above three scenarios, a signal, *resend*, indicating whether the HOL cell should be retransmitted, will be asserted to 1 if one or more than one bits in

the multicast pattern remains 1. The *resend* signal is initially set to 0. If multicast pattern bits are all 0, meaning the HOL cell has been successfully transmitted to all necessary output groups, the *resend* signal will be disasserted. The IPC will then clear the HOL cell in the one-cell buffer and transmit the next cell in the input buffer in the next time slot (if any).

Figure 7.4 gives an example of how a multicast pattern is modified. Let us assume that at the beginning of the m th time slot, the HOL cell is destined for three output groups: 1, 3, K . Therefore, the multicast pattern at the m th time slot, MP^m , has three bits set to 1. Let us also assume that the local priority value (LP) of the HOL cell is 5 and the feedback priority values from groups 1, 2, 3, and K are 7, 2, 3, and 5, respectively, as shown in Figure 7.4. The result of comparing LP with the FPs is 0110...00, which is then logically ANDed with the MP^m and produces a new multicast pattern, 0010...00, for the next time slot (MP^{m+1}). Since only the MP_3^{m+1} is set to 1, the IPC

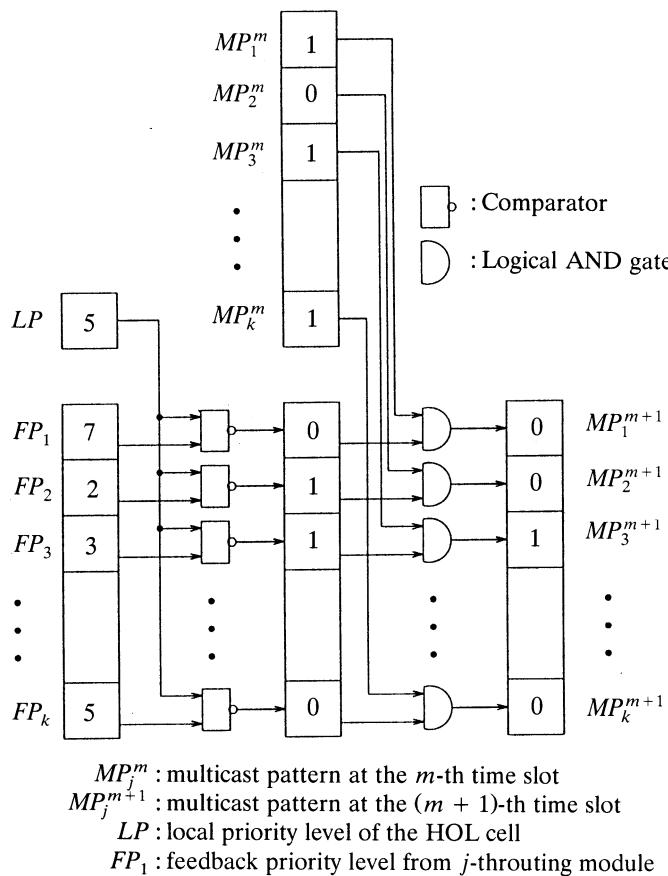


Fig. 7.4 An example of modifying a multicast pattern. (©1997 IEEE.)

determines that the HOL cell has been successfully routed to RMs 1 and K but discarded in RM 3 and will retransmit in the next time slot.

7.3 IMPLEMENTATION OF INPUT PORT CONTROLLER

Figure 7.5 shows a block diagram of the IPC. For easy explanation, let us assume the switch has 256 input ports and 256 output ports, and every 16 successive output ports are in one group. A major difference between this IPC and traditional ones is the addition of the multicast contention resolution unit (MCRU), shown in a dashed box. It determines, by comparing K feedback priorities with the local priority of the HOL cell, whether or not the HOL cell has been successfully routed to all necessary output groups.

Let us start from the left, where the input line from the SONET-ATM network is terminated. Cells 16 bits wide are written into an input buffer.

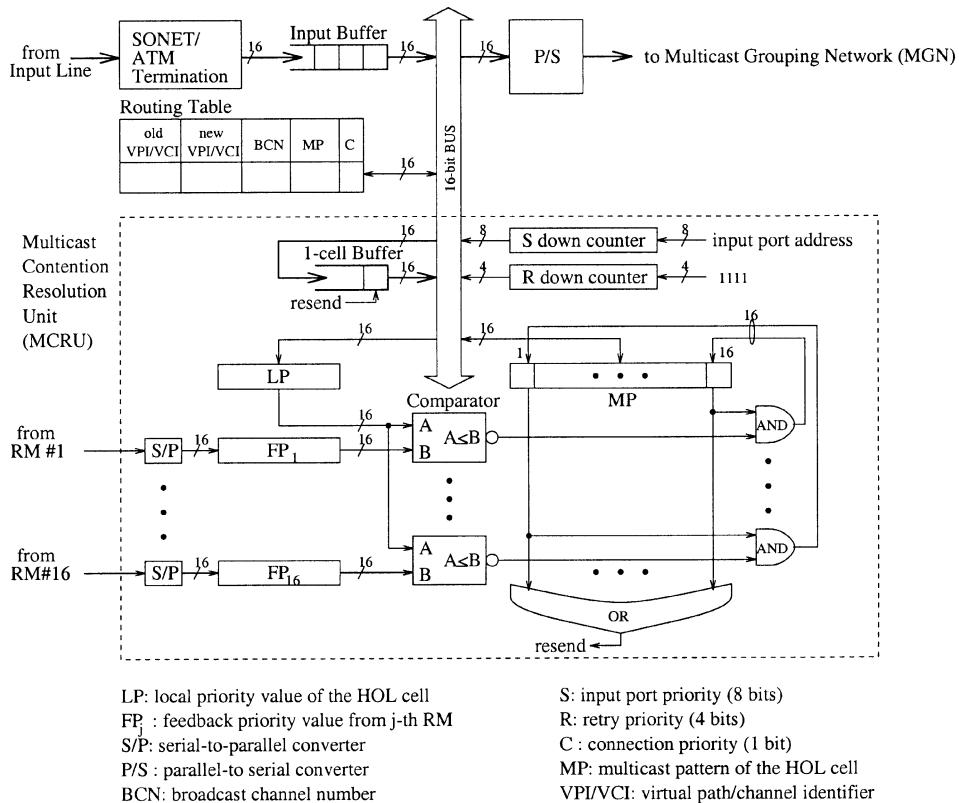


Fig. 7.5 Implementation of the input port controller with $N = 256$, $M = 16$. (©1997 IEEE.)

The HOL cell's VPI/VCI is used to extract necessary information from a routing table. This information includes a new VPI/VCI for unicast connections; a BCN for multicast connections, which uniquely identifies each multicast call in the entire switch; a MP for routing cells in the MGN; and the connection priority C . This information is then combined with a priority field to form the routing information, as shown in Figure 7.3.

As the cell is transmitted to the MGN through a parallel-to-serial converter (P/S), the cell is also temporarily stored in a one-cell buffer. If the cell fails to route successfully through the RMs, it will be retransmitted in the next cell cycle. During retransmission, it is written back to the one-cell buffer in case it fails to route through again. The S down counter is initially loaded with the input address and decremented by one at each cell clock. The R down counter is initially set to all 1s and decreased by one every time the HOL cell fails to transmit successfully. When the R-counter reaches zero, it will remain at zero until the HOL cell has been cleared and a new cell becomes the HOL cell.

K feedback priority signals, FP_1 to FP_K , are converted to 16-bit-wide signals by the serial-to-parallel converters (S/P) and latched at the 16-bit registers. They are simultaneously compared with the HOL cell's local priority (LP) by K comparators. Recall that the larger the priority value is, the lower the priority level is. If the value of FP_j is larger than or equal to LP, the j th comparator's output is asserted low, which will then reset the MP_j bit to zero regardless of what its value was (0 or 1). After the resetting operation, if any one of the MP_j bits is still 1, indicating that at least one HOL cell did not get through the RM in the current cycle, the *resend* signal will be asserted high and the HOL cell will be retransmitted in the next cell cycle with the modified multicast pattern.

As shown in Figure 7.5, there are K sets of S/P, FP register, and comparator. As a switch size increases, the number of output groups, K , also increases. However, if the time permits, only one set of this hardware is required for time-division multiplexing the operation of comparing the local priority value, LP, with K feedback priority values.

7.4 PERFORMANCE

This section discusses the performance analysis of the abacus switch. Both simulation and analytical results are given for comparison. Simulation results are obtained with a 95% confidence interval, with errors not greater than 10% for the cell loss probability or 5% for the maximum throughput and average cell delay.

Figure 7.6 considers an on–off source model in which an arrival process to an input port alternates between on (active) and off (idle) periods. A traffic source continues sending cells in every time slot during the on period, but stops sending cells in the off period. The duration of the on period is

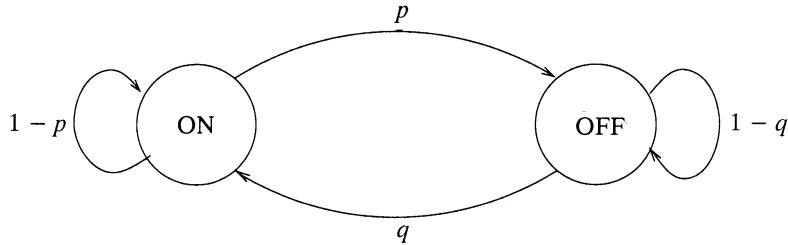


Fig. 7.6 On–off source model.

determined by a random variable $X \in \{1, 2, \dots\}$, which is assumed to be geometrically distributed with a mean of β cells. Similarly, the duration of the off period is determined by a random variable $Y \in \{0, 1, 2, \dots\}$, which is also assumed to be geometrically distributed with a mean of α cells. Define q as the probability of starting a new burst (on period) in each time slot, and p as the probability of terminating a burst. Cells arriving at an input port are assumed to be destined for the same output. Thus, the probability that the on period lasts for a duration of i time slots is

$$\Pr\{X = i\} = p(1 - p)^{i-1}, \quad i \geq 1,$$

and we have

$$\beta = E[X] = \sum_{i=1}^{\infty} i \Pr\{X = i\} = \frac{1}{p}.$$

The probability that an off period lasts for j time slots is

$$\Pr\{Y = j\} = q(1 - q)^{j-1}, \quad j \geq 0,$$

and we have

$$\alpha = E[Y] = \sum_{j=0}^{\infty} j \Pr\{Y = j\} = \frac{1 - q}{q}.$$

7.4.1 Maximum Throughput

The maximum throughput of an ATM switch employing input queuing is defined by the maximum utilization at the output port. An input-buffered switch's HOL blocking problem can be alleviated by speeding up the switch fabric's operation rate or increasing the number of routing links with an expansion ratio L . Several other factors also affect the maximum throughput. For instance, the larger the switch size (N) or burstiness (β), the smaller the

maximum throughput (ρ_{\max}) will be. However, the larger the group expansion ratio (L) or group size (M) is, the larger the maximum throughput will be.

Karol et al. [9] have shown that the maximum throughput of an input-buffered ATM switch is 58.6% for $M = 1$, $L = 1$, $N \rightarrow \infty$, with random traffic. Oie et al. [13] have obtained the maximum throughput of an input-output-buffered ATM switch for $M = 1$, an arbitrary group expansion ratio or speedup factor L , and an infinite N with random traffic. Pattavina [14] has obtained, through computer simulations, the maximum throughput of an input-output-buffered ATM switch, using the channel grouping concept for an arbitrary group size M , $L = 1$, and an infinite N with random traffic. Liew and Lu [12] have obtained the maximum throughput of an asymmetric input-output-buffered switch module for arbitrary M and L , and $N \rightarrow \infty$ with bursty traffic.

Figure 7.7 shows that the maximum throughput is monotonically increasing with the group size. For $M = 1$, the switch becomes an input-buffered switch, and its maximum throughput ρ_{\max} is 0.586 for uniform random traffic

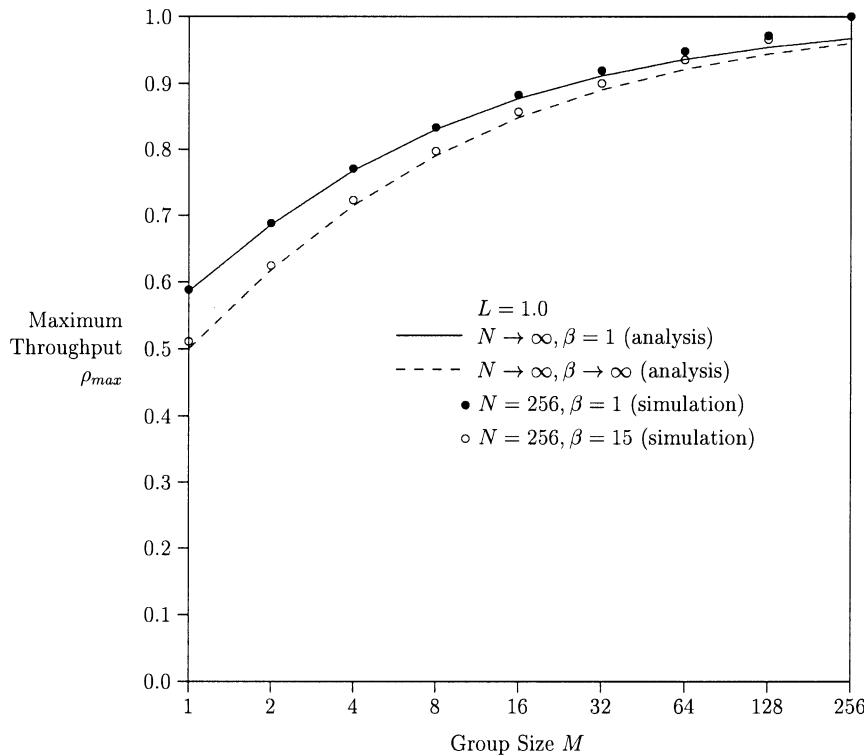


Fig. 7.7 Maximum throughput vs. group size, $L = 1.0$.

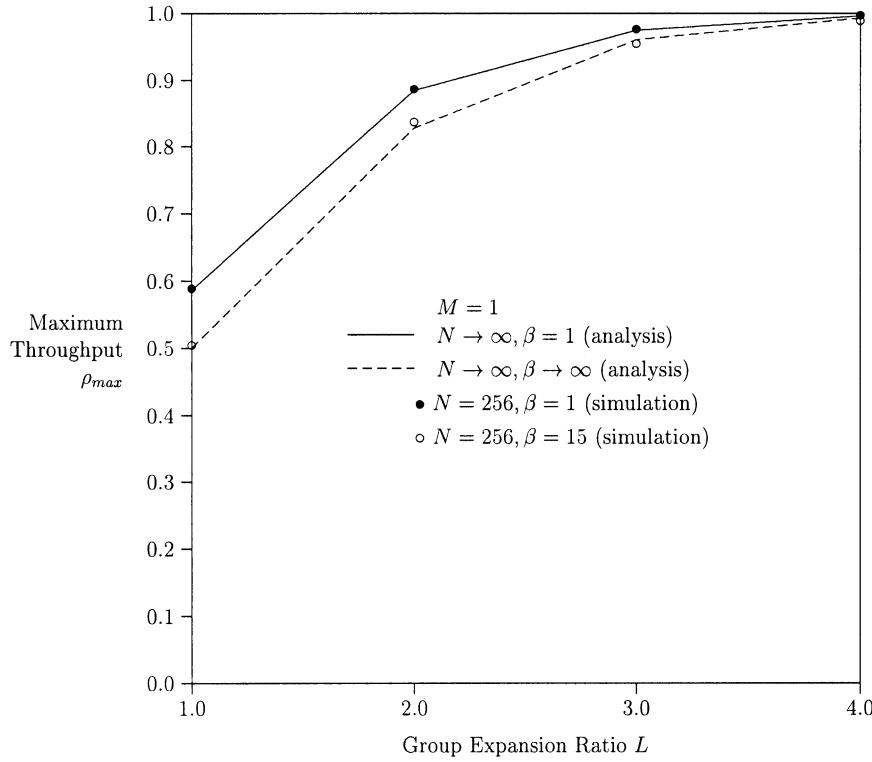


Fig. 7.8 Maximum throughput vs. group expansion ratio with $M = 1$.

($\beta = 1$), and 0.5 for completely bursty traffic ($\beta \rightarrow \infty$). For $M = N$, the switch becomes a completely shared-memory switch such as Hitachi's switch [11]. Although it can achieve 100% throughput, it is impractical to implement a large-scale switch using such an architecture. Therefore, choosing M between 1 and N is a compromise between the throughput and the implementation complexity.

Figures 7.8 and 7.9 compare theoretical values and simulation values of the maximum throughput with different group expansion ratios L for $M = 1$ and $M = 16$, respectively. The theoretical values can be obtained from Liew and Lu's analysis [12].

A HOL virtual queue is defined as the queue that consists of cells at the HOL of input buffers destined for a tagged output group. For uniform random traffic, the average number of cells, $E[C]$, in the HOL virtual queue becomes

$$E[C] = \frac{\rho_o(2LM - \rho_o) - LM(LM - 1)}{2(LM - \rho_o)} + \sum_{k=1}^{LM-1} \frac{1}{1 - z_k}, \quad (7.1)$$

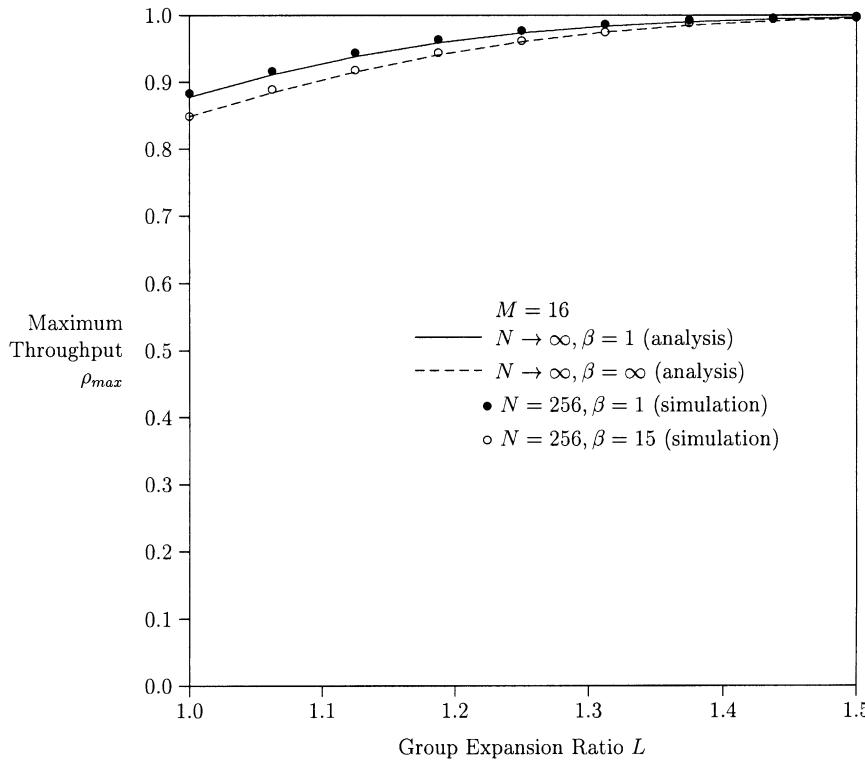


Fig. 7.9 Maximum throughput vs. group expansion ratio with $M = 16$.

where $z_0 = 1$ and z_k ($k = 1, \dots, LM - 1$) are the roots of $z^{LM}/A(z) = [p + (1-p)z]^{LM}$ and $A(z) = e^{-p\rho_o(1-z)}$. For completely bursty traffic, $E[C]$ becomes

$$E[C] = \frac{\sum_{k=0}^{LM-1} k(LM-k)c_k + \rho_o}{LM - \rho_o} \quad (7.2)$$

where

$$c_k = \begin{cases} \rho_o^k c_0 / k! & \text{if } k < LM, \\ \rho_o^k c_0 / [(LM)! (LM)^{k-LM}] & \text{if } k \geq LM, \end{cases}$$

$$c_0 = \frac{LM - \rho_o}{\sum_{k=0}^{LM-1} (LM-k)\rho_o^k / k!}.$$

The maximum throughput of the proposed ATM switch can be obtained by considering the total number of backlogged cells in all K HOL virtual queues to be N . This means under a saturation condition, there is always a HOL cell at each input queue. Since it is assumed that cells are uniformly distributed over all output groups, we obtain

$$E[C] = N/K = M. \quad (7.3)$$

The maximum throughput can be obtained by equating (7.1) and (7.3) for random traffic, and equating (7.2) and (7.3) for bursty traffic. If ρ_o^* satisfies both equations, the maximum throughput at each input, ρ_{\max} , is related to ρ_o^* by $\rho_{\max} = \rho_o^*/M$.

For a given M , the maximum throughput increases with L because there are more routing links available between input ports and output groups. However, since a larger L means higher hardware complexity, the value of L should be selected prudently so that both hardware complexity and the maximum throughput are acceptable. For instance, for a group size M of 16 (for input traffic with an average burst length of 15 cells), L has to be at least 1.25 to achieve a maximum throughput of 0.95. But for a group size M of 32 and the same input traffic characteristic, L need only be at least 1.125 to achieve the same throughput. Both analytical results and simulation results show that the effect of input traffic's burstiness on the maximum throughput is very small. For example, the maximum throughput for uniform random traffic with $M = 16$ and $L = 1.25$ is 97.35%, and for completely bursty traffic is 96.03%, only a 1.32% difference. The discrepancy between theoretical and simulation results comes from the assumptions on the switch size N and β . In the theoretical analysis, it is assumed that $N \rightarrow \infty$, $\beta \rightarrow \infty$, but in the simulation it is assumed that $N = 256$, $\beta = 15$. As these two numbers increase, the discrepancy decreases.

7.4.2 Average Delay

A cell may experience two kinds of delay while traversing the abacus switch: input-buffer delay and output-buffer delay. In the theoretical analysis, the buffer size is assumed to be infinite. But in the simulations, it is assumed that the maximum possible sizes for the input buffers and output buffers that can be sustained in computer simulations are $B_i = 1024$ and $B_o = 256$, respectively. Here, B_i is the input buffer size and B_o is the normalized output buffer size [i.e., the size of a physical buffer (4096) divided by M (16)]. Although finite buffers may cause cell loss, it is small enough to be neglected when evaluating average delay.

Let us assume each input buffer receives a cell per time slot with a probability λ , and transmits a cell with a probability μ . The probability μ is equal to 1.0 if there is no HOL blocking. But, as the probability of HOL blocking increases, μ will decrease. If $\lambda > \mu$, then the input buffer will

rapidly saturate, and the delay at it will be infinite. The analytical results for uniform random traffic can be obtained from Chang et al.'s analysis [2]:

$$E[T] = \frac{1 - \lambda}{\mu - \lambda}, \quad (7.4)$$

where $\lambda = \rho_i$, $\mu = \rho_i/E[C]$, and $E[C]$ is a function of M , L , ρ_i , and β as $N \rightarrow \infty$, which can be obtained from (7.1) or (7.2).

Note that the input buffer's average delay is very small when the input offered load is less than the maximum saturation throughput. This results from the small HOL blocking probability before the saturated throughput. It also shows that the effect of the burstiness of input traffic on the input buffer's average delay is very small when the traffic load is below the maximum throughput.

Figure 7.10 compares the average delay at the input and output buffers. Note that the input buffer's average delay is much smaller than the output

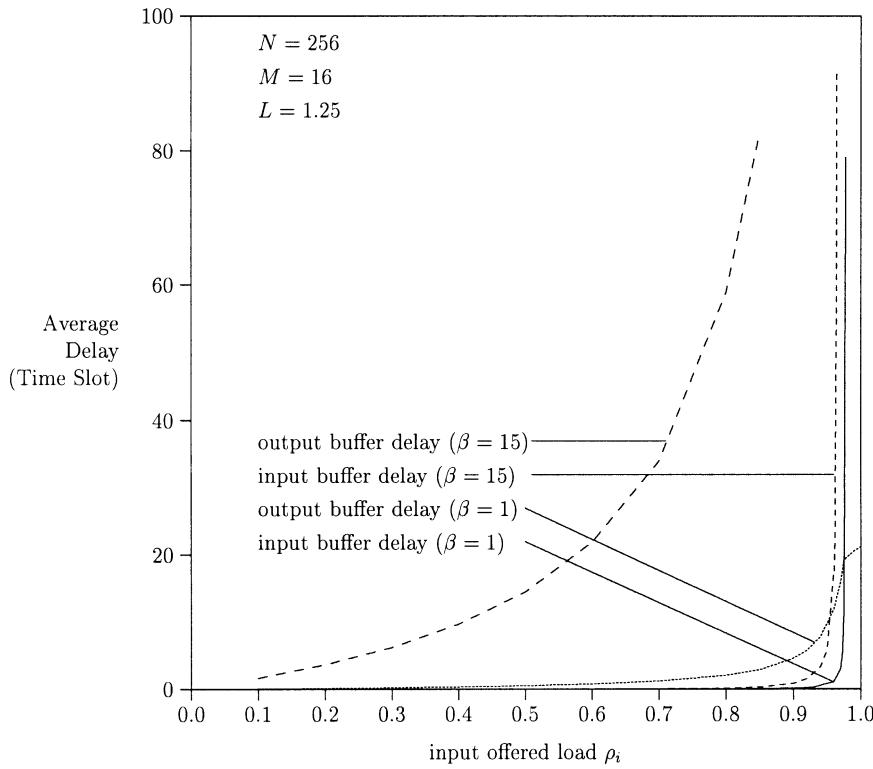


Fig. 7.10 Comparison of average input buffer delay and average output buffer delay.

buffer's at traffic loads less than the saturated throughput. For example, for an input offered load ρ_i of 0.8 and an average burst length β of 15, the output buffer's average delay T_o is 58.8 cell times, but the input buffer's average delay T_i is only 0.1 cell time.

Figure 7.11 shows simulation results on the input buffer's average delay vs. the expanded throughput ρ_j for both unicast and multicast traffic. It is assumed that the number of replicated cells is distributed geometrically with an average of c . The expanded throughput ρ_j is measured at the inputs of the SSM and normalized to each output port. Note that multicast traffic has a lower delay than unicast traffic, because a multicast cell can be sent to multiple destinations in a time slot, while a unicast cell can be sent to only one destination in a time slot. For example, assume that an input port i has 10 unicast cells and the other input port j has a multicast cell with a fanout of 10. Input port i will take at least 10 time slots to transmit the 10 unicast cells, while input port j can possibly transmit the multicast cell in one time slot.

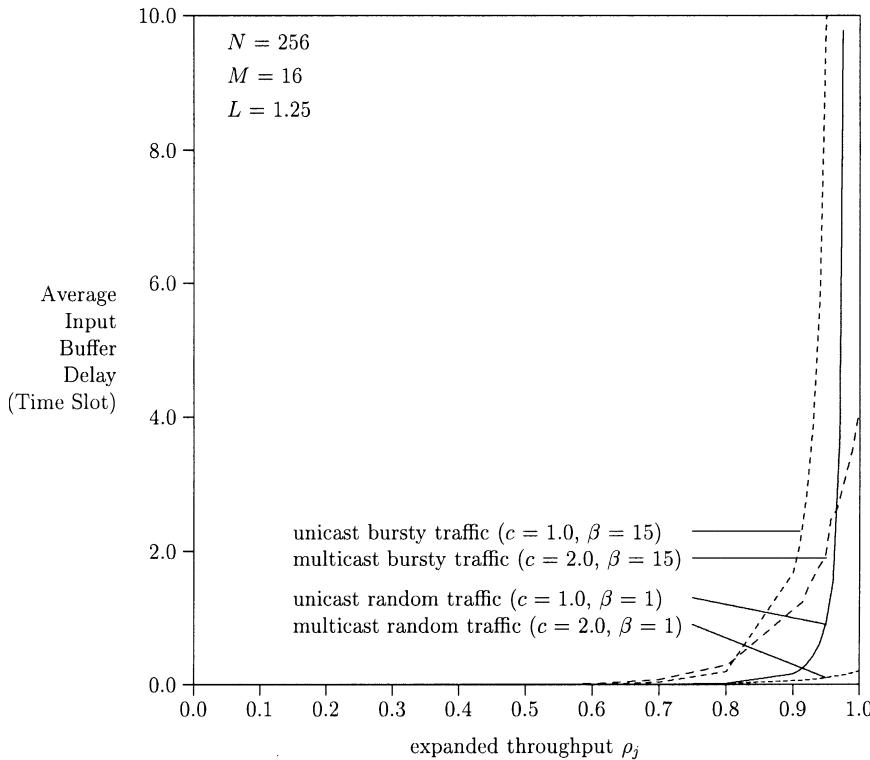


Fig. 7.11 Average input buffer delay vs. expanded throughput for unicast and multicast traffic (simulation).

7.4.3 Cell Loss Probability

As suggested in [15], there can be two buffer control schemes for an input-output-buffered switch: the queue loss (QL) scheme and the backpressure (BP) scheme. In the QL scheme, cell loss can occur at both input and output buffers. In the BP scheme, by means of backward throttling, the number of cells actually switched to each output group is limited not only to the group expansion ratio (LM), but also to the current storage capability in the corresponding output buffer. For example, if the free buffer space in the corresponding output buffer is less than LM , only the number of cells corresponding to the free space are transmitted, and all other HOL cells destined for that output group remain at their input buffers. The abacus switch can easily implement the backpressure scheme by forcing the AB in Figure 7.2 to send the dummy cells with the highest priority level, which will automatically block the input cells from using those routing links. Further-

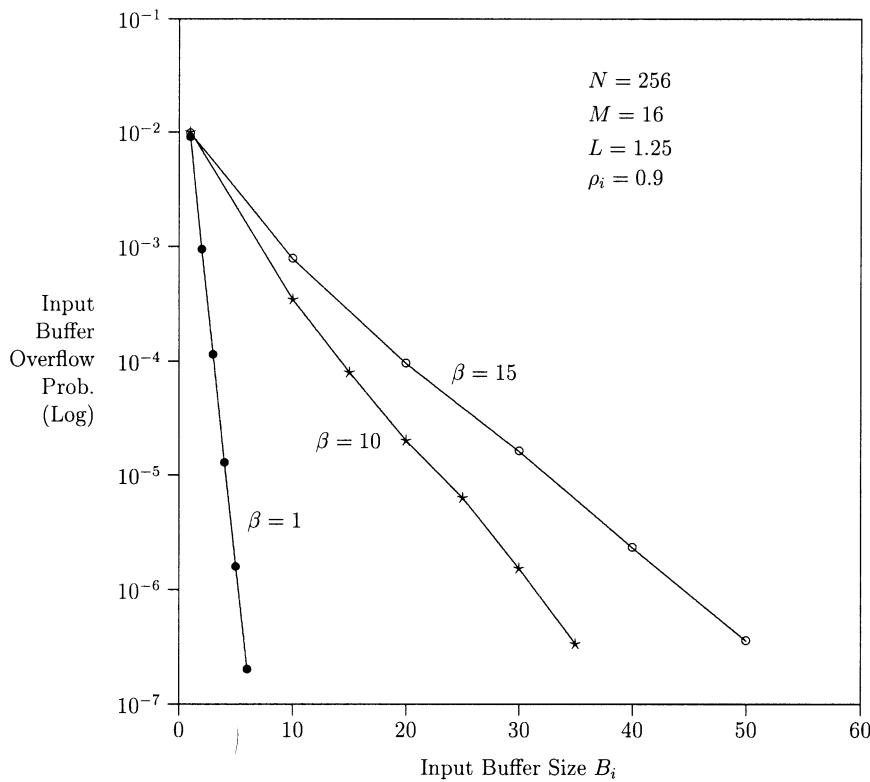


Fig. 7.12 Input buffer overflow probability vs. input buffer size (simulation).

more, the number of blocked links can be dynamically changed based on the output buffer's congestion situation.

Here, we only consider the QL scheme (cell loss at both input and output buffers). In the abacus switch, cell loss can occur at input and output buffers, but not in the MGN. Figure 7.12 shows input buffer overflow probabilities with different average burst lengths β . For uniform random traffic, an input buffer with a capacity of a few cells is sufficient to maintain the buffer overflow probability less than 10^{-6} . As the average burst length increases, so does the cell loss probability. For an average burst length β of 15, the required input buffer size can be a few tens of cells for the buffer overflow probability of 10^{-6} . By extrapolating the simulation result, the input buffer size is found to be about 100 cells for 10^{-10} cell loss rate.

Figure 7.13 shows output buffer overflow probabilities with different average burst lengths. Here, B_o is the normalized buffer size for each output. It is shown that the required output buffer size is much larger than the input buffer size for the same cell loss probability.

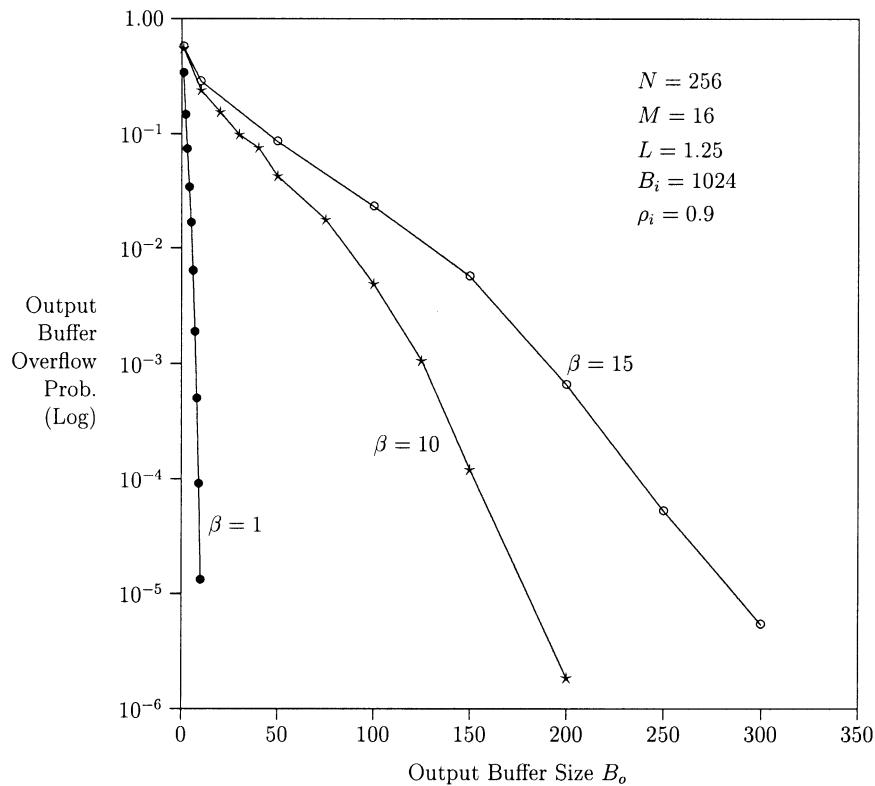


Fig. 7.13 Output buffer overflow probability vs. output buffer size (simulation).

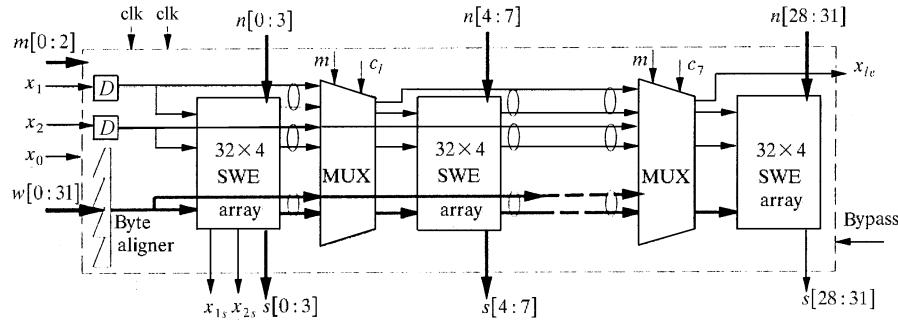


Fig. 7.14 Block diagram of the ARC chip.

7.5 ATM ROUTING AND CONCENTRATION CHIP

An application-specific integrated circuit (ASIC) has been implemented based on the abacus switch architecture. Figure 7.14 shows the ARC chip's block diagram. Each block's function and design are explained briefly in the following sections. Details can be found in [3]. The ARC chip contains 32×32 SWEs, which are partitioned into eight SWE arrays, each with 32×4 SWEs. A set of input data signals, $w[0:31]$, comes from the IPCs. Another set of input data signals, $n[0:31]$, either comes from the output, $s[0:31]$, of the chips on the above row, or is tied to high for the chips on the first row (in the multicast case). A set of the output signals, $s[0:31]$, either go to the north input of the chips one row below or go to the output buffer.

A signal x_0 is broadcast to all SWEs to initialize each SWE to across state, where the west input passes to the east and the north input passes to the south. A signal x_1 specifies the address bit(s) used for routing cells, while x_2 specifies the priority field. Other output signals x propagate along with cells to the adjacent chips on the east or south side.

Signals $m[0:1]$ are used to configure the chip into four different group sizes as shown in Table 7.1: (1) eight groups, each with 4 output links, (2) four groups, each with 8 output links, (3) two groups, each with 16 output links, and (4) one group with 32 output links. A signal $m[2]$ is used to configure the

TABLE 7.1 Truth Table for Different Operation Modes^a

$m[1]$	$m[0]$	Operation
0	0	8 groups with 4 links per group
0	1	4 groups with 8 links per group
1	0	2 groups with 16 links per group
1	1	1 group with 32 links per group

^a $m[2] = 1$ multicast, $m[2] = 0$ unicast.

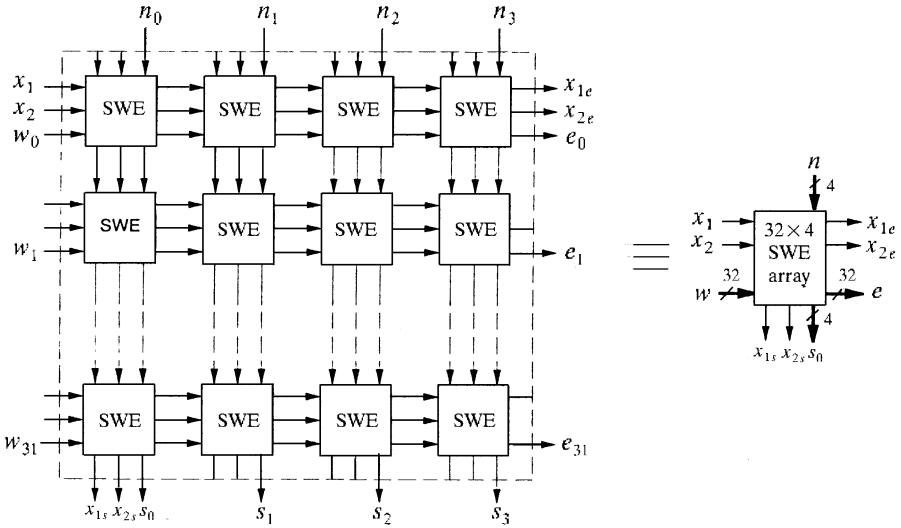


Fig. 7.15 32 × 4 SWE array.

chip to either unicast or multicast application. For the unicast case, $m[2]$ is set to 0, while for the multicast case, $m[2]$ is set to 1.

As shown in Figure 7.15, the SWEs are arranged in a crossbar structure, where signals only communicate between adjacent SWEs, easing the synchronization problem. ATM cells are propagated in the SWE array similarly to a wave propagating diagonally toward the bottom right corner. The signals x_1 and x_2 are applied from the top left of the SWE array, and each SWE distributes them to its east and south neighbors. This requires the same phase of the signal arriving at each SWE. x_1 and x_2 are passed to the neighbor SWEs (east and south) after one clock cycle delay, as are the data signals (w and n). A signal x_0 is broadcast to all SWEs (not shown in Figure 7.15) to precharge an internal node in the SWE in every cell cycle. The output signal x_{1e} is used to identify the address bit position of the cells in the first SWE array of the next adjacent chip.

The timing diagram of the SWE input signal and its two possible states are shown in Fig. 7.16. Two bit-aligned cells, one from the west and one from the north, are applied to the SWE along with the signals dx_1 and dx_2 , which determine the address and priority fields of the input cells. The SWE has two states: cross and toggle. Initially, the SWE is initialized to a cross state by the signal dx_0 , i.e., cells from the north side are routed to the south side, and cells from the west side are routed to the east side. When the address of the cell from the west (dw_a) is matched with the address of the cell from the north (dn_a), and when the west's priority level (dw_p) is higher than the north's (dn_p), the SWEs are toggled. The cell from the west side is then routed

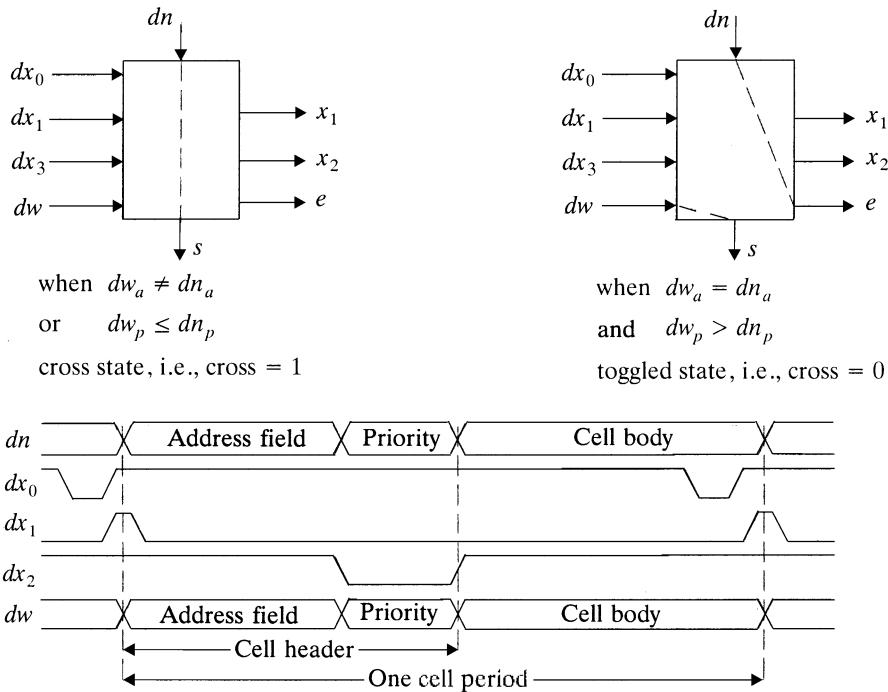


Fig. 7.16 Two states of the switch element.

to the south side, and the cell from the north is routed to the east. Otherwise, the SWE remains at the cross state.

The 32×32 ARC chip has been designed and fabricated using $0.8\text{-}\mu\text{m}$ CMOS technology with a die size of $6.6 \text{ mm} \times 6.6 \text{ mm}$. Note that this chip is pad-limited. The chip has been tested successfully up to 240 MHz, and its characteristics are summarized in Table 7.2. Its photograph is shown in Figure 7.17.

TABLE 7.2 Chip Summary

Process technology	0.8- μm CMOS, triple metal
Number of switching elements	32×32
Configurable group size	4, 8, 16, or 32 output links
Pin count	145
Package	Ceramic PGA
Number of transistors	81,000
Die size	$6.6 \times 6.6 \text{ mm}^2$
Clock signals	Pseudo ECL
Interface signals	TTL/CMOS inputs, CMOS outputs
Maximum clock speed	240 MHz
Worst case power dissipation	2.8 W at 240 MHz

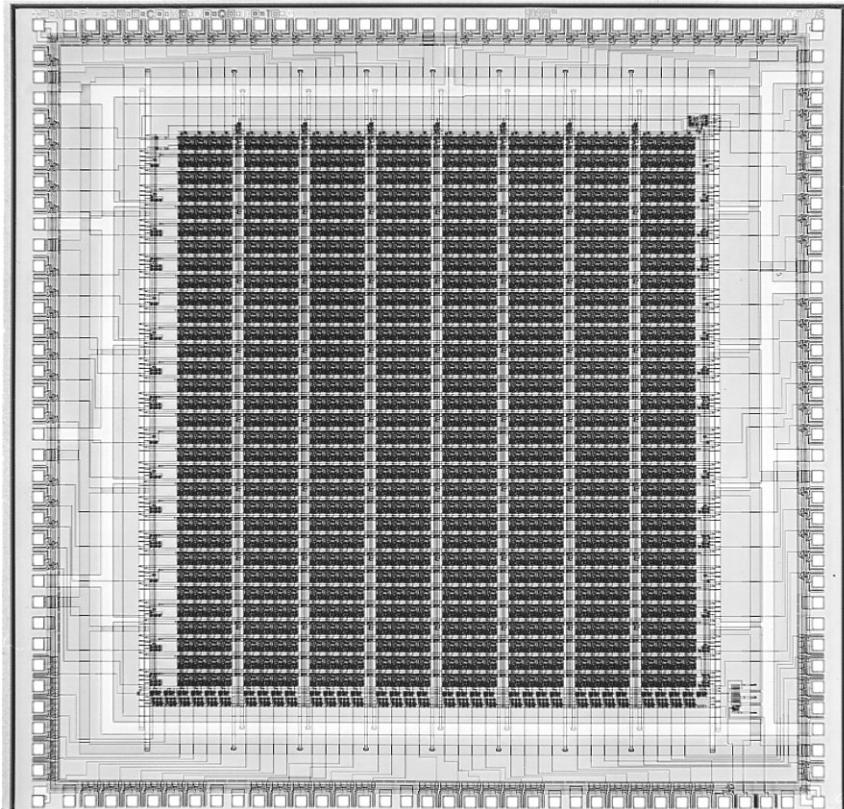


Fig. 7.17 Photograph of the ARC chip. (©1997 IEEE.)

7.6 ENHANCED ABACUS SWITCH

This section discusses three approaches of implementing the MGN in Figure 7.1 to scale up the abacus switch to a large size. As described in Section 7.2, the time for routing cells through an RM and feeding back the lowest-priority information from the RM to all IPCs must be less than one cell slot time. The feedback information is used to determine whether or not the cell has been successfully routed to the destined output group(s). If not, the cell will continue retrying until it has reached all the desired output groups. Since each SWE in an RM introduces a 1-bit delay as the signal passes it in either direction, the number of SWEs between the uppermost link and the right-most link of an RM should be less than the number of bits in a cell. In other words, $N + LM - 1 < 424$ (see Section 7.2). For example, if we choose $M = 16$, $L = 1.25$, the equation becomes $N + 1.25 \times 16 - 1 < 424$. The maximum value of N is 405, which is not large enough for a large-capacity switch.

7.6.1 Memoryless Multistage Concentration Network

One way to scale up the abacus switch is to reduce the time spent on traversing cells from the uppermost link to the rightmost link in an RM (see Fig. 7.2). Let us call this time the *routing delay*. In a single-stage abacus switch, the routing delay is $N + LM - 1$, which limits the switch size, because it grows with N .

To reduce the routing delay, the number of SWEs that a cell traverses in an RM must be minimized. If we divide an MGN into many small MGNs, the routing delay can be reduced. Figure 7.18 shows a two-stage memoryless multistage concentration network (MMCN) architecture that can implement a large-capacity abacus switch. It consists of N IPCs, $J (= n/M)$ MGNs, and $K (= N/M)$ concentration modules (CMs). Each MGN has K RMs, and each RM has n input links and LM output links. Each CM has $J \times LM$ input links and LM output links.

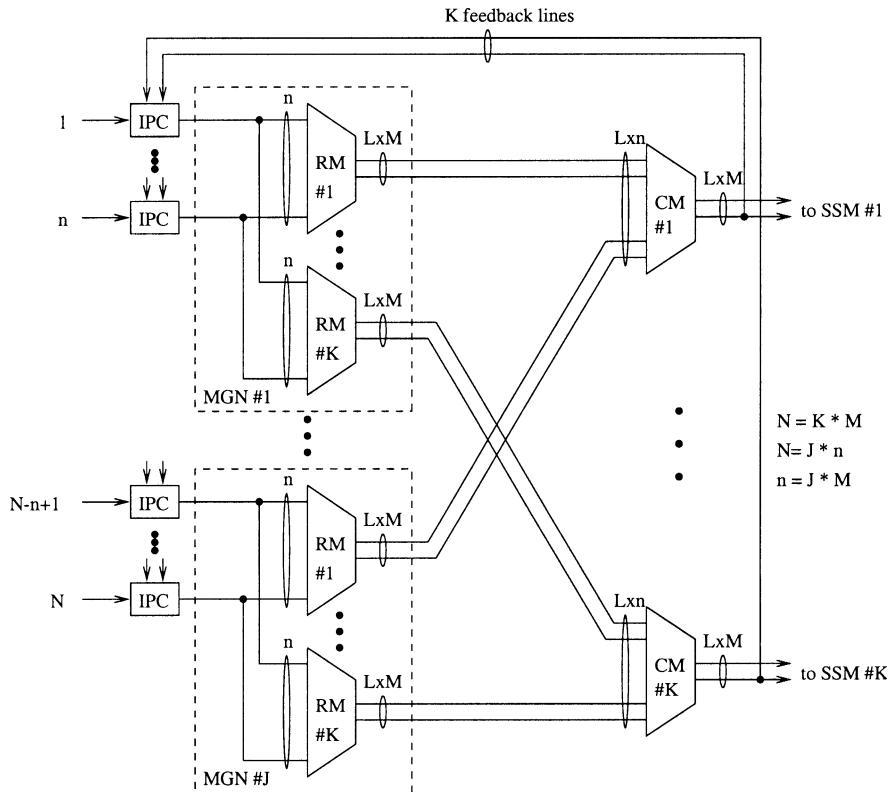


Fig. 7.18 A two-stage memoryless multistage concentration network.

After cells are routed through the RMs, they need to be further concentrated at the CMs. Since cells that are routed to the CM always have correct output group addresses, we do not need to perform a routing function in the CM. In the CM, only the concentration function is performed by using the priority field in the routing information. The structure and implementation of the RM and the CM are identical, but the functions performed are slightly different.

Recall that each group of M output ports requires LM routing links to achieve high delay-throughput performance. The output expansion ratio of the RM must be equal to or greater than that of the CM. If not, the multicast contention resolution algorithm does not work properly. For example, let us assume that $N = 1024$, $M = 16$, and $n = 128$. Consider the case that there are 16 links between an RM and a CM, while there are 20 links between a CM and an SSM. If all 128 cells of MGN 1 are destined for output group 1 and no cells from other MGNs are destined for output group 1, the feedback priority of CM 1 will be the priority of the address broadcaster, which has the lowest priority level. Then, all 128 cells destined for output group 1 are cleared from the IPCs of MGN 1, even though only 20 cells can be accepted in the SSM. The other 108 cells will be lost. Therefore, the output expansion ratio of the RM must be equal to or greater than that of the CM.

Let us define n as the module size. The number of input links of an RM is n , and the number of input links of the CM is $J \times LM$. By letting $n = JM$, the number of input links of the CM is of the same order as the number of input links of the RM, because we can engineer M so that L is close to one.

In the MMCN, the feedback priorities (FPs) are extracted from the CMs and broadcast to all IPCs. To maintain the cell sequence integrity from the same connection, the cell behind the HOL cell at each IPC cannot be sent to the switch fabric until the HOL cell has been successfully transmitted to the desired output port(s). In other words, the routing delay must be less than one cell slot. This requirement limits the MMCN to a certain size.

Cells that have arrived at a CM much carry the address of the associated output group (either valid cells or dummy cells from the RM's AB). As a result, there is no need of using the AB in the CM to generate dummy cells to carry the address of the output group. Rather, the inputs that are reserved for the AB are replaced by the routing links of MGN1. Thus, the routing delay of the two-stage MMCN is $n + (J - 1)(LM) + LM - 1$, as shown in Figure 7.19, which should be less than 424. Therefore, we have the following equation by replacing J with n/M : $n + (n/M - 1)(LM) + LM - 1 < 424$. This can be simplified to $n < 425/(1 + L)$. Thus, $N = Jn = n^2/M < 425^2/M(1 + L)^2$. Table 7.3 shows the minimum value of L for a given M to get a maximum throughput of 99% with random uniform traffic.

Clearly, the smaller the group size M , the larger the switch size N . The largest abacus switch can be obtained by letting $M = 1$. But in this case the group expansion ratio L must be equal to or greater than 4 to have

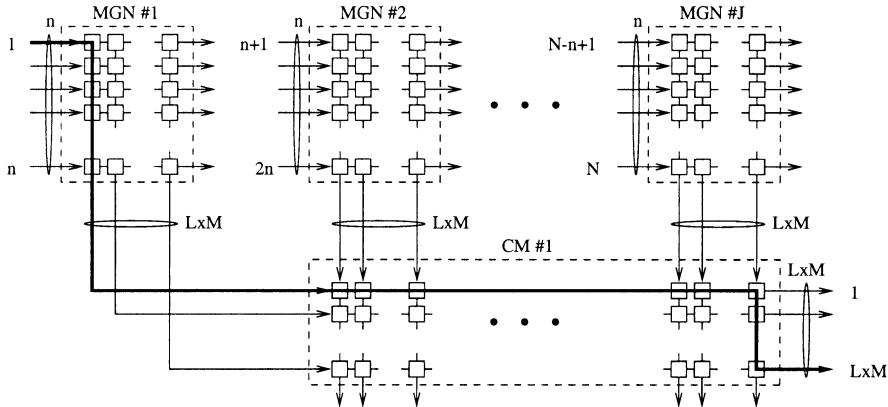


Fig. 7.19 Routing delay in a two-stage MMCN.

TABLE 7.3 The Minimum Value of L For a Given M

M	1	2	4	8	16	32
L	4	3	2.25	1.75	1.25	1.125

satisfactory delay-throughput performance. Increasing the group size M reduces the maximum switch size N , but also reduces the number of feedback links (N^2/M) and the number of SWEs ($LN^2 + L^2Nn$). Therefore, by engineering the group size properly, we can build a practical large-capacity abacus switch. For example, if we choose $M = 16$ and $L = 1.25$, then the maximum module size n is 188, and the maximum switch size N is 2209. With the advanced CMOS technology (e.g., 0.25 μm), it is feasible to operate at OC-12 rate (i.e., 622 Mbit/s). Thus, the MM CN is capable of providing more than 1 Tbit/s capacity.

7.6.2 Buffered Multistage Concentration Network

Figure 7.20 shows a two-stage buffered multistage concentration network (BMCN). As discussed in the previous section, the MM CN needs to have the feedback priority lines connected to all IPCs, which increases the interconnection complexity. This can be resolved by keeping RMs and CMs autonomous, where the FPs are extracted from the RMs rather than from the CMs. However, buffers are required in the CMs, since cells that successfully pass through the RMs and are cleared from input buffers may not pass through the CMs.

Figure 7.21 shows three ways of building the CM. Figure 7.21(a) uses a shared-memory structure similar to the MainStreet Xpress 36190 core services

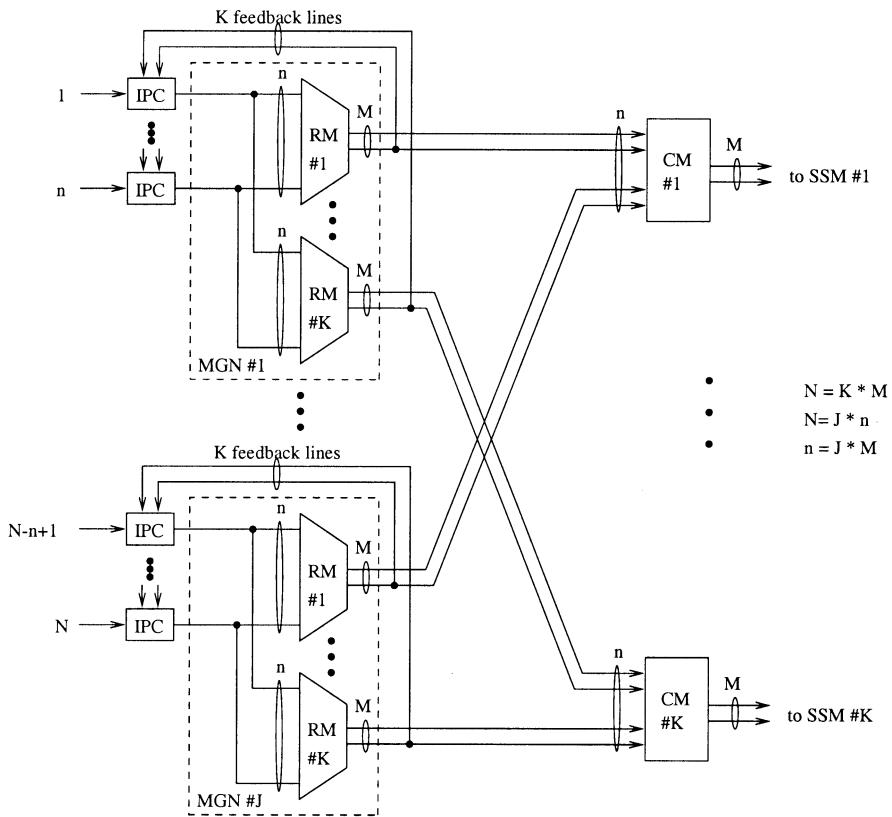


Fig. 7.20 A two-stage buffered multistage concentration network.

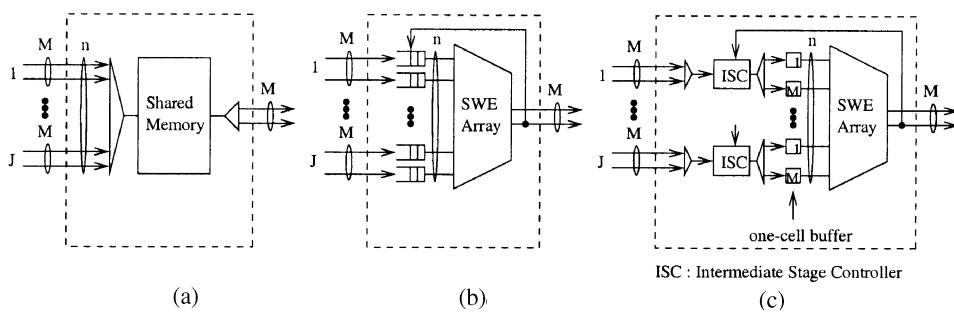


Fig. 7.21 Three ways of building the concentration module.

switch [16]} and the concentrator-based growable switch architecture [7]. Its size is limited by the memory speed constraint.

Figure 7.21(b) shows another way to implement the CM, by using a two-dimensional SWE array and input buffers. One potential problem of having buffers at the input links of the CM is cells out of sequence. This is because after cells that belong to the same virtual connection are routed through an RM, they may be queued at different input buffers. Since the queue lengths in the buffers can be different, cells that arrive at the buffer with shorter queue lengths will be served earlier by the CM, resulting in cells out of sequence.

This out-of-sequence problem can be eliminated by time-division multiplexing cells from an RM (M cells), storing them in an intermediate stage controller (ISC), and sending them sequentially to M one-cell buffers, as shown in Figure 7.21(c). The ISC has an internal FIFO buffer and logic circuits that handles feedback priorities as in the abacus switch. This achieves a global FIFO effect and thus maintains the cells' sequence. Each ISC can receive up to M cells and transmit up to M cells during each cell time slot.

The key to maintaining cell sequence is to assign priority properly. At each ISC, cells are dispatched to the one-cell buffers whenever the one-cell buffers become empty and there are cells in the ISC. When a cell is dispatched to the one-cell buffer, the ISC assigns a priority value to the cell. The priority field is divided into two parts, port priority and sequence priority. The former is the more significant. The port priority field has $\lceil \log_2 J \rceil$ bits for the J ISCs in a CM, where $\lceil x \rceil$ denotes the smallest integer that is equal to or greater than x . The sequence priority field must have at least $\lceil \log_2 JM \rceil$ bits to ensure the cell sequence integrity to accommodate LM priority levels, as will be explained in an example later. The port priority is updated in every arbitration cycle (i.e., in each cell slot time) in a RR fashion. The sequence priority is increased by one whenever a cell is dispatched from the ISC to a one-cell buffer. When the port priority has the highest priority level, the sequence priority is reset to zero at the beginning of the next arbitration cycle (assuming the smaller the priority value, the higher the priority level). This is because all cells in the one-cell buffers will be cleared at the current cycle. Using this dispatch algorithm, cells in the ISC will be delivered to the output port in sequence. The reason that the sequence priority needs to have LM levels is to accommodate the maximum number of cells that can be transmitted from an ISC between two reset operations for the sequence priority.

Figure 7.22 shows an example of cell's priority assignment scheme for $J = 3$ and $M = 4$. The port priority p and sequence priority q are represented by two numbers in $[p, q]$, where $p = 0, 1, 2$, and $q = 0, 1, 2, \dots, 11$. During each time slot, each ISC can receive up to four cells and transmit up to four cells. Let us consider the following case. ISC 1 receives four cells in every time slot, ISC 3 receives one cell in every time slot, and ISC 2 receives no cells.

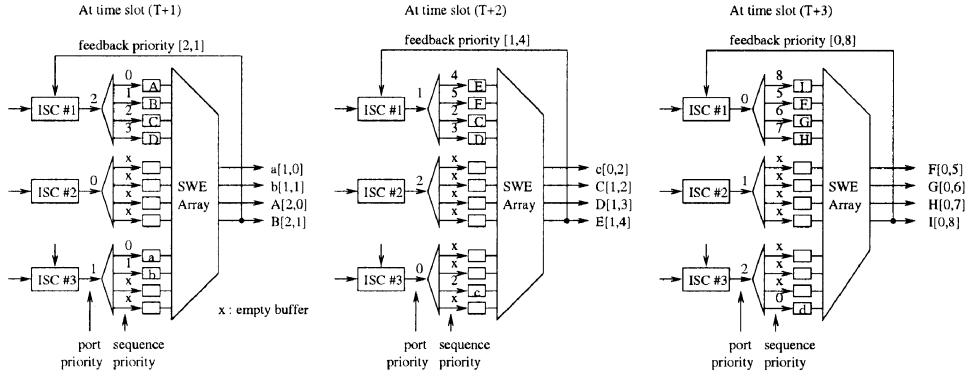


Fig. 7.22 An example of priority assignment in the concentration module.

The port priority is changed in every time slot in a RR fashion. In time slot T , ISC 1 has the highest port priority level, and ISC 3 has the lowest port priority level. In time slot $T + 1$, ISC 2 has the highest port priority level, and ISC 1 has the lowest port priority level, and so on. ISC 3 has a higher port priority level than ISC 1 in time slots $T + 1$ and $T + 2$.

In time slot T , all four cells in the one-cell buffers of ISC 1 pass through the CM, because they have the highest priority levels. In time slot $T + 1$, ISC 3 transmits two cells (a and b) and ISC 1 transmits two cells (A and B). In time slot $T + 2$, ISC 3 transmits one cell (c), while ISC 1 transmits three cells (C , D , and E). In time slot $T + 3$, ISC 1 has the highest port priority (0) and is able to transmit its all cells (F , G , H , and I). Once they are cleared from the one-cell buffers, ISC 1 resets its sequence priority to zero.

7.6.3 Resequencing Cells

As discussed before, the routing delay in the abacus switch must be less than 424 bit times. If it is greater, there are two possibilities. First, if a cell is held up in the input buffer longer than a cell slot time, the throughput of the switch fabric will be degraded. Second, if a cell next to the HOL cell is sent to the MGN before knowing if the HOL cell is successfully transmitted to the desired output switch module(s), it may be ahead of a HOL cell that did not pass through the MGN. This cell-out-of-sequence problem can be resolved with a resequencing buffer (RSQB) at the output port of the MGN.

For a switch size N of 1024, output group size M of 16, and group expansion ratio L of 1.25, the maximum routing delay of the single-stage abacus switch is 1043 (i.e., $N + LM - 1$ as described previously). Therefore, an arbitration cycle is at least three cell time slots. If up to three cells are allowed to transmit during each arbitration cycle, the IPC must have three one-cell buffers arranged in parallel.

Figure 7.23 illustrates an example of the maximum extent of being out of sequence. Assume cells A to J are stored in the same input buffer in

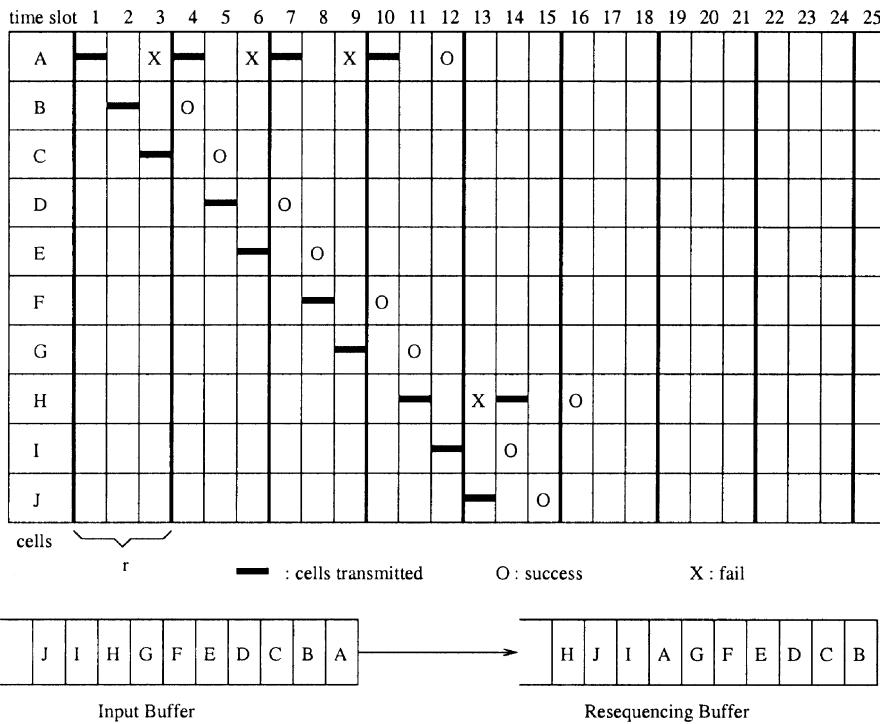


Fig. 7.23 An example of cell out of sequence with an arbitration cycle of 3 cell slots.

sequence. In time slot 1, cell *A* is sent to the switch fabric. It takes three time slots to know if cell *A* has passed through the switch fabric successfully. In time slot 2, cell *B* is sent to the switch fabric, and in time slot 3, cell *C* is sent to the switch fabric. Before the end of time slot 3, the IPC knows that cell *A* has failed to pass the switch fabric, so cell *A* will be transmitted again in time slot 4. If cell *A* passes through the switch fabric successfully on the fourth try, up to six cells (cells *B* to *G*) can be ahead of cell *A*. The cell arrival sequence in the RSQB is shown in Figure 7.23.

For this scheme to be practical, the maximum extent of being out of sequence and the size of the RSQB should be bounded. Let us consider the worst case. If all HOL cells of N input ports are destined for the same output group and the tagged cell has the lowest port priority in time slot T , then LM highest-priority cells will be routed in time slot T . In time slot $T + 1$, the priority level of the tagged cell will be incremented by one, so that there can be at most $N - LM - 1$ cells whose priority levels are higher than that of the tagged cell. In time slot $T + 2$, there can be at most $(N - 2LM - 1)$ cells whose priority levels are higher than that of the tagged cell, and so on.

The maximum extent of being out of sequence can be obtained from the following. An arbitration cycle is equal to or greater than $r =$

TABLE 7.4 The Maximum Extent of Being Out of Sequence (t)

N	M	L	r	s	t
1024	16	1.25	3	52	156
8192	16	1.25	20	410	8200

$[(N + LM - 1)/424]$ cell time slots. A tagged cell succeeds in at most $s = \lceil N/LM \rceil$ tries. Therefore, in the worst case, the HOL cell passes through the switch fabric successfully in rs time slots. Denote the maximum extent of being out of sequence by t ; then $t = rs$.

One way to implement the resequencing is to timestamp each cell with a value equal to the actual time plus t when it is sent to the switch fabric. Let us call this timestamp the *due time* of the cell. Cells at the RSQBs are moved to the SSM when their due times are equal to the actual time. This maintains the cell transmission sequence.

The drawbacks of this scheme are high implementation complexity of the RSQB and large cell transfer delay. Since a cell must wait in the RSQB until the actual time reaches the due time of the cell, every cell experiences at least a t cell-slot delay even if there is no contention. Table 7.4 shows the maximum extent of being out of sequence for switch sizes of 1024 and 8192. For switch size 1024, the maximum is 156 cell slots, which corresponds to 441 μs for the OC-3 line rate (i.e., $156 \times 2.83 \mu\text{s}$).

7.6.4 Complexity Comparison

This subsection compares the complexity of the above three approaches for building a large-capacity abacus switch and summarizes the results in Table 7.5. Here, the switch element in the RMs and CMs is a 2×2 crosspoint device. The number of interstage links is the number of links between the RMs and the CMs. The number of buffers is the number of ISCs. For the BMCN, the CMs have ISCs and one-cell buffers [Fig. 7.21(c)]. The second and third parts of Table 7.5 give some numerical values for a 160-Gbit/s abacus switch. Here, it is assumed that the switch size N is 1024, the input line speed is 155.52 Mbit/s, the group size M is 16, the group expansion ratio L is 1.25, and the module size n is 128.

With respect to the MMCN, there are no internal buffers and no out-of-sequence cells. Its routing delay is less than 424 bit times. But the numbers of SWEs and interstage links are the highest among the three approaches.

For the BMCN, the routing delay is no longer a concern for a large-capacity abacus switch. Its numbers of SWEs and interstage links are smaller than those of the MMCN. However, it may not be cost-effective when it is required to implement buffer management and cell scheduling in the intermediate-stage buffers to meet QoS requirements.

The last approach to resequencing out-of-sequence cells has no interstage links or internal buffers. It requires a resequencing buffer at each output

TABLE 7.5 Complexity Comparison of Three Approaches for a 160-Gbit/s Abacus Switch

	MMCN	BMCN	Resequencing
No. of switch elements	$LN^2 + L^2Nn$	$N^2 + Nn$	LN^2
No. of interstage links	JLN	JN	0
No. of internal buffers	0	JK	0
No. of MP bits	K	K	K
Out-of-sequence delay	0	0	$\left\lceil \frac{N + LM - 1}{424} \right\rceil \times \left\lceil \frac{N}{LM} \right\rceil$
Routing delay in bits	$n + Ln - 1$	$n + LM - 1$	$N + LM - 1$
Switch size N	1024	1024	1024
Group size M	16	16	16
Output exp. ratio L	1.25	1.25	1.25
Module size n	128	128	128
No. of switch elements	1,515,520	1,179,648	1,310,720
No. of interstage links	10,240	8,192	0
No. of internal buffers	0	512	0
No. of MP bits	64	64	64
Out-of-sequence delay	0	0	156
Routing delay in bits	287	147	1,043

port. For a switch capacity of 160 Gbit/s, the delay caused by resequencing cells is at least 156 cell slot times.

7.7 ABACUS SWITCH FOR PACKET SWITCHING

The abacus switch can also handle variable-length packets. To preserve the cell¹ sequence in a packet, two cell scheduling schemes are used at the input buffers. The *packet interleaving* scheme transfers all cells in a packet consecutively, while the *cell interleaving* scheme transfers cells from different inputs and reassembles them at the output.

7.7.1 Packet Interleaving

A packet switch using the packet interleaving technique is shown in Figure 7.24. The switch consists of a memoryless nonblocking multicast grouping network (MGN), input port controllers (IPCs), and output port controllers (OPCs). Arriving cells are stored in an input buffer until the last cell of the packet arrives. When the last cell arrives, the packet is then eligible for transmission to output port(s).

In the packet interleaving scheme, all cells belonging to the same packet are transferred consecutively. That is, if the first cell in the packet wins the

¹The cell discussed in this section is just a fixed-length segment of a packet and does not have to be 53 bytes like an ATM cell.

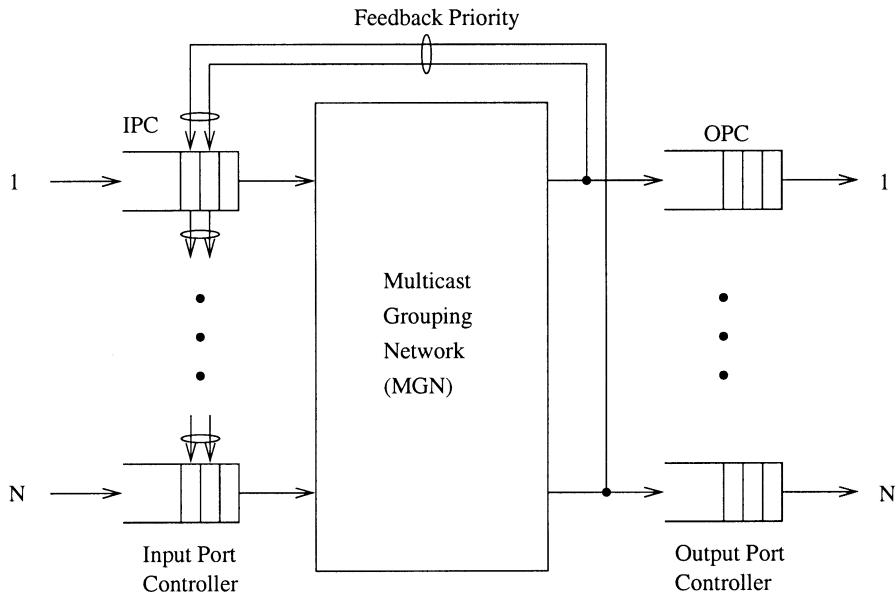


Fig. 7.24 A packet switch with packet interleaving.

output port contention for a destination among the contending input ports, all the following cells of the packet will be transferred consecutively to the destination.

A packet interleaving switch can be easily implemented by the abacus switch. Only the first cells of HOL packets can contend for output ports. The contention among the first cells of HOL packets can be resolved by properly assigning priority fields to them. The priority field of a cell has $1 + \log_2 N$ bits. Among them, the $\log_2 N$ -bit field is used to achieve fair contention by dynamically changing its value as in the abacus switch. Prior to the contention resolution, the most significant bit (MSB) of the priority field is set to 1 (low priority). As soon as the first cell of an HOL packet wins the contention (known from the feedback priorities), the MSB of the priority field of all the following cells in the same packet is asserted to 0 (high priority). As soon as the last cell of the packet is successfully sent to the output, the MSB is set to 1 for the next packet. As a result, it is ensured that cells belonging to the same packet are transferred consecutively.

Figure 7.25 shows the average packet delay vs. offered load for a packet switch with packet interleaving. In the simulations, it is assumed that the traffic source is an on-off model. The packet size is assumed to have a truncated geometric distribution with an average packet size of 10 cells and maximum packet size of 32 cells (to accommodate the maximum Ethernet frame size). The packet delay through the switch is defined as follows. When the last cell of a packet arrives at an input buffer, the packet is timestamped

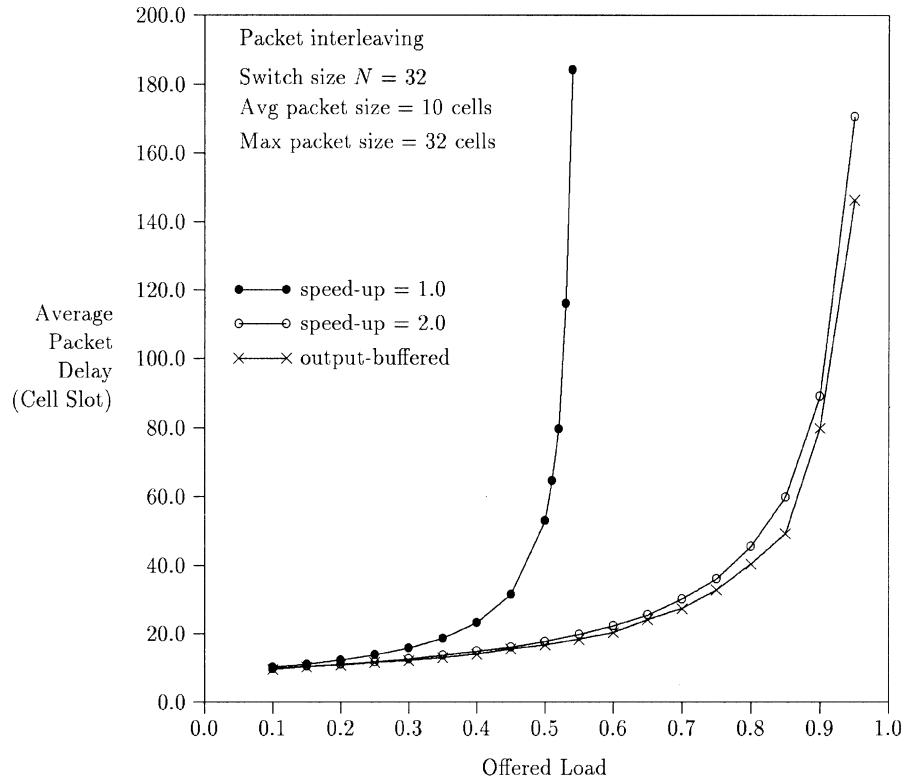


Fig. 7.25 Delay performance of a packet switch with packet interleaving.

with an arrival time. When the last cell of a packet leaves the output buffer, the packet is timestamped with a departure time. The difference between the arrival time and the departure time is defined as the packet delay. When there is no internal speedup ($S = 1$) in the switch fabric, the delay performance of the packet interleaving switch is very poor, mainly due to the HOL blocking. The delay–throughput performance is improved by increasing the speedup factor S (e.g., $S = 2$). Note that the input buffer’s average delay is much smaller than the output buffer’s average delay. With an internal speedup of two, the output buffer’s average delay dominates the total average delay and is very close to that of the output-buffered switch.

7.7.2 Cell Interleaving

A packet switch using a cell interleaving technique is shown in Figure 7.26. Arriving cells are stored in the input buffer until the last cell of a packet arrives. Once the last cell arrives, cells are transferred in the same way as in an ATM switch. That is, cells from different input ports can be interleaved

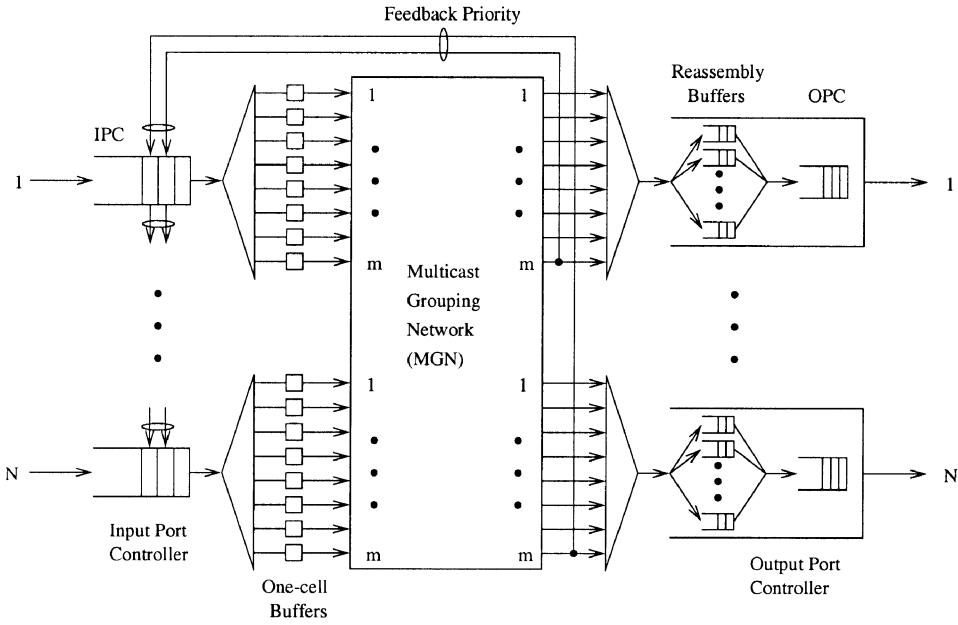


Fig. 7.26 A packet switch with cell interleaving.

with each other as they arrive at the output port. Cells have to carry input port numbers so that output ports can distinguish them from different packets. Therefore, each output port has N reassembly buffers, each corresponding to an input port. When the last cell of a packet arrives at the reassembly buffer, all cells belonging to the packet are moved to the output buffer for transmission to the output link. In real implementation, only pointers are moved, not the cells. This architecture is similar the one in [17] in the sense that they both use reassembly buffers at the outputs, but it is more scalable than the one in [17].

The operation speed of the abacus switch fabric is limited to several hundred megabits per second with state-of-the-art CMOS technology. To accommodate the line rate of a few gigabits per second (e.g., Gigabit Ethernet and OC-48), we can either use a bit-slice technique or the one shown in Figure 7.26, where the high-speed cell stream is distributed to m one-cell buffers at each input port. The way of dispatching cells from the IPC to the m one-cell buffers is identical to dispatching cells from the ISC to the M one-cell buffers in Figure 7.21(c). The advantage of the technique in Figure 7.26 over the bit-slice technique is its smaller overhead bandwidth: the latter shrinks the cell duration while keeping the same overhead for each cell.

Figure 7.27 shows the average packet delay vs. offered load for a packet switch with cell interleaving. When there is no internal speedup ($S = 1$), the delay performance is poor. With an internal speedup S of two, the delay

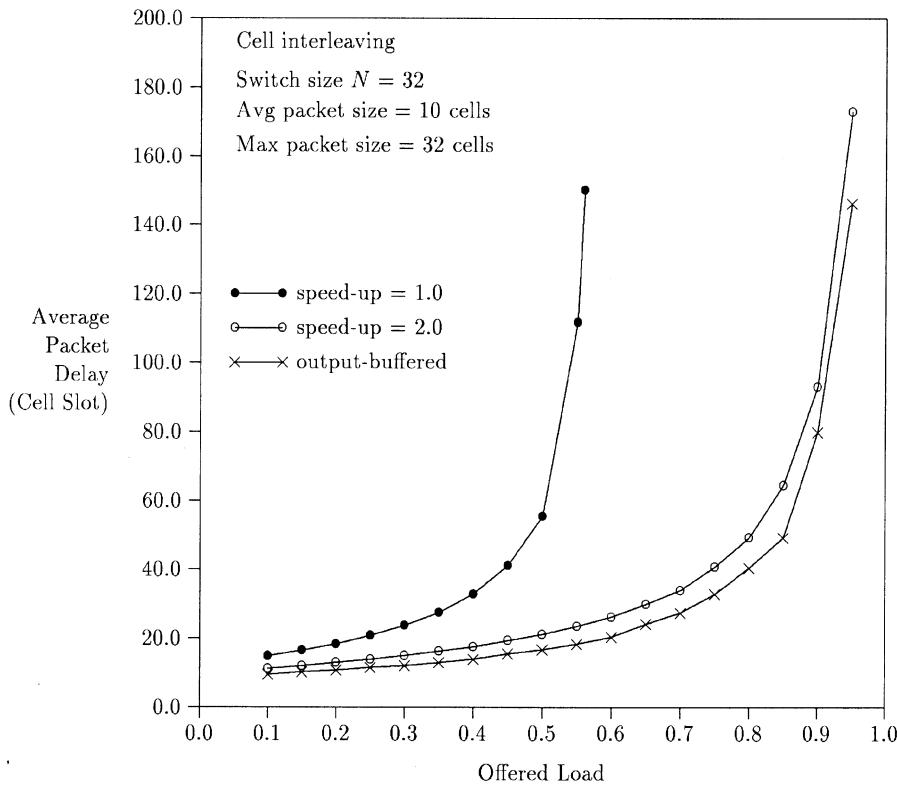


Fig. 7.27 Delay performance of a packet switch with cell interleaving.

performance is close to that of an output-buffered packet switch. By comparing the delay performance between Figure 7.25 and Figure 7.27, we can see that the average delay performance is comparable. However, we believe that the delay variation of cell interleaving will be smaller than that of packet interleaving because of its finer granularity in switching.

REFERENCES

1. B. Bingham and H. Bussey, "Reservation-based contention resolution mechanism for Batcher-banyan packet switches," *Electron. Lett.*, vol. 24, no. 13, pp. 772–773, Jun. 1988.
2. C. Y. Chang, A. J. Paulraj, and T. Kailath, "A broadband packet switch architecture with input and output queueing," *Proc. IEEE GLOBECOM '94*, pp. 448–452, Nov. 1994.
3. H. J. Chao and N. Uzun, "An ATM routing and concentration chip for a scalable multicast ATM switch," *IEEE J. Solid-State Circuits*, vol. 32, no. 6, pp. 816–828, Jun. 1997.

4. H. J. Chao, B. S. Choe, J. S. Park, and N. Uzun, "Design and implementation of abacus switch: a scalable multicast ATM switch," *IEEE J. Select. Areas Commun.*, vol. 15, no. 5, pp. 830–843, Jun.1997.
5. H. J. Chao and J. S. Park, "Architecture designs of a large-capacity abacus ATM switch," *Proc. IEEE GLOBECOM*, Nov. 1998.
6. A. Cisneros and C. A. Brackett, "A large ATM switch based on memory switches and optical star couplers," *IEEE J. Select. Areas Commun.*, vol. SAC-9, no. 8, pp. 1348–1360, Oct. 1991.
7. K. Y. Eng and M. J. Karol, "State of the art in gigabit ATM switching," *IEEE BSS '95*, pp. 3–20, Apr. 1995.
8. R. Händel, M. N. Huber, and S. Schröder, *ATM Networks: Concepts, Protocols, Applications*, Addison-Wesley Publishing Company, Chap. 12, 1998.
9. M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, pp. 1347–1356, Dec. 1987.
10. J. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Select. Areas Commun.*, vol. SAC-5, no. 8, pp. 1264–1273, Oct. 1987.
11. T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano, "32 × 32 shared buffer type ATM switch VLSI's for B-ISDN's," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1239–1247, Oct. 1991.
12. S. C. Liew and K. W. Lu, "Comparison of buffering strategies for asymmetric packet switch modules," *IEEE J. Select. Areas Commun.*, vol. 9, no. 3, pp. 428–438, Apr. 1991.
13. Y. Oie, M. Murata, K. Kubota, and H. Miyahara, "Effect of speedup in nonblocking packet switch," *Proc. IEEE ICC '89*, pp. 410–415.
14. A. Pattavina, "Multichannel bandwidth allocation in a broadband packet switch," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1489–1499, Dec. 1988.
15. A. Pattavina and G. Bruzzi, "Analysis of input and output queueing for nonblocking ATM switches," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 314–328, Jun. 1993.
16. E. P. Rathgeb, W. Fischer, C. Hinterberger, E. Wallmeier, and R. Wille-Fier, "The MainStreet Xpress core services node—a versatile ATM switch architecture for the full service network," *IEEE J. Select. Areas Commun.*, vol. 15, no. 5, pp. 830–843, Jun.1997.
17. I. Widjaja and A. I. Elwalid, "Performance issues in VC-merge capable switches for IP over ATM networks," *Proc. IEEE INFOCOM '98*, pp. 372–380, Mar. 1998.

CHAPTER 8

CROSSPOINT-BUFFERED SWITCHES

When more than one cell is contending for the limited link capacity inside a switch or at its outputs, buffers are provided to temporarily store the cells. Since the size of a buffer is finite, cells will be discarded when it is full. However, in previous chapters we have described several switch architectures with input buffering, output buffering (including shared memory), input and output buffering, and internal buffering in a multistage structure.

This chapter describes crosspoint-buffered switches, where each crosspoint has a buffer. This switch architecture takes advantage of today's CMOS technology, where several millions of gates and several tens of millions of bits can be implemented on the same chip. Furthermore, there can be several hundred input and output signals operating at 2–3 Gbit/s on one chip. This switch architecture does not require any increase of the internal line speed. It eliminates the HOL blocking that occurs in the input-buffered switch, at the cost of having a large amount of crosspoint-buffer memory at each crosspoint.

The remainder of this chapter is organized as follows. Section 8.1 describes a basic crosspoint-buffered switch architecture. The arbitration time among crosspoint buffers can become a bottleneck as we increase the switch size. Section 8.2 introduces a scalable distributed-arbitration (SDA) switch to avoid the bottleneck of the arbitration time. Section 8.3 describes an extended version of SDA to support multiple QoS classes.

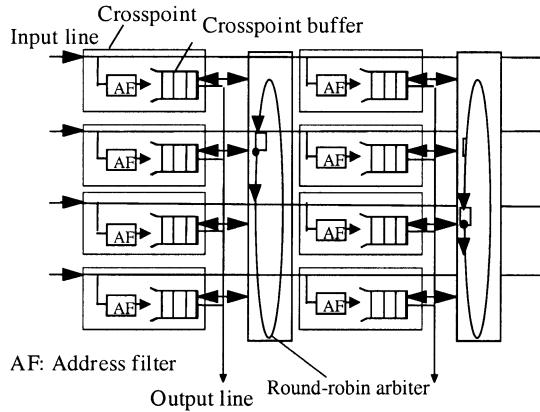


Fig. 8.1 Crosspoint-buffered switch structure based on round-robin arbitration.

8.1 OVERVIEW OF CROSSPOINT-BUFFERED SWITCHES

A basic crosspoint-buffered switch architecture is shown in Figure 8.1. Each crosspoint has a buffer to store cells that come from the associated input port and are destined for the associated output port.

Contention control is needed to resolve contention among crosspoint buffers that belong to the same output port. One candidate for the contention control is to use round-robin (RR) arbitration. This is because the RR arbitration provides fairness and its implementation is very simple.

The RR arbiter searches, from some starting point, for a crosspoint buffer that has made a request to transfer a cell to the output line. The starting point is just below the crosspoint buffer from which a cell was sent to the output line at the previous cell time. If the RR arbiter finds the request, the cell at the head in the crosspoint buffer is selected to release its cell. At the next cell time, the starting point is reset to just below the selected crosspoint buffer. Thus, in the worst case, the control signal for ring arbitration must pass through all the crosspoint buffers belonging to the same output line within one cell time.

For that reason, in the buffered crossbar that employs RR arbitration for the contention control, the maximum output-line speed is limited by the number of input ports, or switch size, and the transmission delay of the control signals in each crosspoint.

The maximum output-line speed C_{\max} (bits/s) is given by the following equation:

$$C_{\max} = \frac{L}{NT_s}, \quad (8.1)$$

where the number of input ports (in other word, the switch size) is N , the transmission delay of the control signals in a crosspoint is T_s (s), and the

length of a cell is L (bits). T_s depends on the performance of devices and the length between crosspoints. When we construct a large-scale switch, its crossbar function can not be implemented on one chip, due to constraints from memory and gate amounts and the number of I/O pins. Therefore, we need to connect several chips to construct a large-scale switch.

As N increases, C_{\max} decreases. For example, at $T_s = 3.0$ ns and $N = 16$, C_{\max} is 8.8 Gbit/s, when we set L to 53×8 bits. Thus, since the crosspoint-buffered switch employs RR arbitration, the arbitration time limits the output-line speed according to the number of input ports to ensure that the RR arbitration can be completed within one cell time. As a result, unless T_s is made small by using ultrahigh-speed devices, the RR based switch cannot achieve large throughput.

8.2 SCALABLE DISTRIBUTED-ARBITRATION SWITCH

This section describes a scalable distributed-arbitration (SDA) switch, to solve the problem of the RR based switch as described Section 8.1. The SDA switch was developed by Nippon Telegraph and Telephone Corporation (NTT) [2].

8.2.1 SDA Structure

Figure 8.2 shows the structure of the SDA switch. The SDA switch has a crosspoint buffer, a transit buffer, an arbitration-control part (CNTL), and a selector at every crosspoint.

A crosspoint buffer sends a request (REQ) to CNTL if there is at least one cell stored in the crosspoint buffer. A transit buffer stores several cells that are sent from either the upper crosspoint buffer or the upper transit buffer. The transit buffer size is one or a few cells, so that both overflow and underflow can be avoided. The required transit buffer size is determined by the round-trip delay of control signals between two adjacent crosspoints. The transit buffer sends REQ to CNTL, as does the crosspoint buffer, if there is at least one cell stored in the transit buffer. If the transit buffer is full, it sends not-acknowledgment (NACK) to the upper CNTL.

If there are any REQs and CNTL does not receive NACK from the next lower transit buffer, CNTL selects a cell within one cell time. CNTL determines which cell should be sent according to the following cell selection rule. The selected cell is sent through a selector to the next lower transit buffer or the output line.

The rule selects a cell as follows. If either the crosspoint buffer or the transit buffer requests cell release, the cell in the requesting buffer is selected. If both the crosspoint buffer and the transit buffer request cell release, the cell with the larger delay time is selected. The delay time is defined as the time since the cell entered the crosspoint buffer.

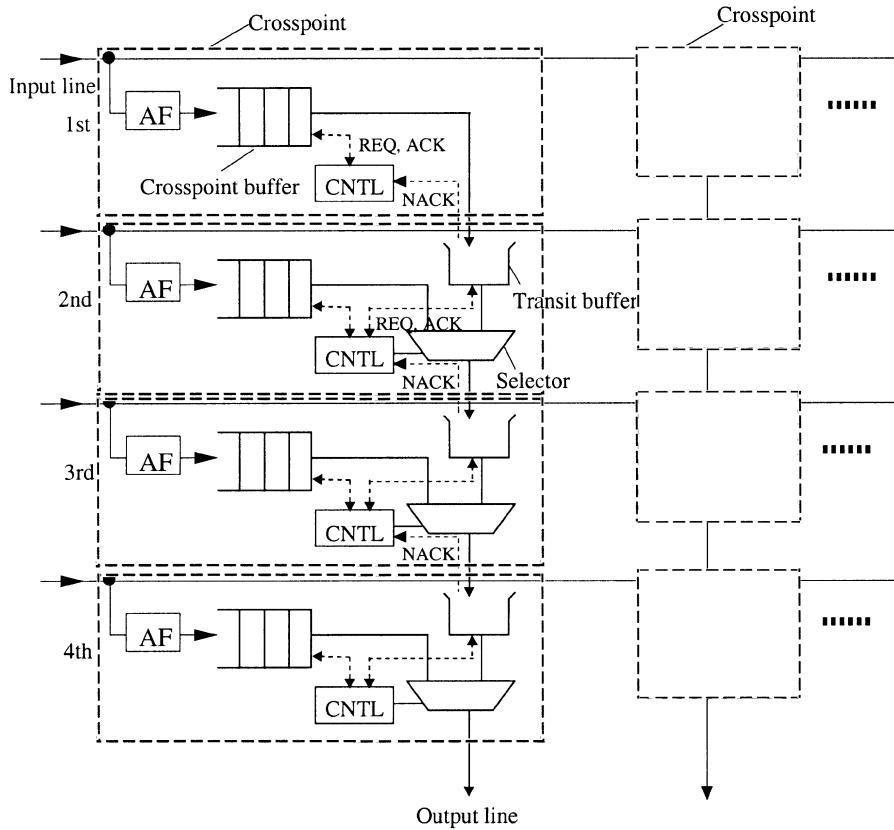


Fig. 8.2 Scalable distributed-arbitration switch structure. (©1997 IEEE.)

One way of comparing the delay time of competitive cells is to use asynchronous counter, which needs S bits, and also the same overhead bit in each cell. The synchronous counter is incremented by one in each cell time. All the synchronous counter's values are synchronized. When a cell enters a crosspoint buffer, the value of the synchronous counter is written in the overhead of the cell. When both a crosspoint buffer and a transit buffer issue requests for cell release, the values of both counters are compared. If the difference in values is more than 2^{S-1} , the cell with smaller value is selected. To the contrary, if the difference is equal to or less than 2^{S-1} , the cell with larger value is selected. Under the condition that the maximum delay time is less than 2^{S-1} , this delay-time comparison works. As will be explained in the next section, $S = 8$ is sufficiently large in the SDA switch.

When the delay time of the cell in the crosspoint buffer equals that in the transit buffer, CNTL determines which cell should be sent using the second cell selection rule. Let us consider the k th crosspoint and transit buffers

counting from the top. The second rule is that the k th crosspoint buffer is selected with probability $1/k$, while the k th transit buffer is selected with probability of $(k - 1)/k$. For example, the third crosspoint buffer and the transit buffer are selected with probabilities $\frac{1}{3}$ and $\frac{2}{3}$, respectively.

Thus the SDA switch achieves distributed arbitration at each crosspoint. The longest control signal transmission distance for arbitration within one cell time is obviously the distance between two adjacent crosspoints. In the conventional switch, the control signal for ring arbitration must pass through all crosspoint buffers, belonging to the same output line. For that reason, the arbitration time of the SDA switch does not depend on the number of input ports.

8.2.2 Performance of SDA Switch

SDA switch performance was evaluated in terms of delay time and crosspoint buffer size by computer simulation. It is assumed that, in an $N \times N$ cross-point-buffered switch, the input traffic is random, the input load is 0.95, and cells are distributed uniformly to all crosspoint buffers belonging to the same input line.

The SDA switch ensures delay time fairness. Figure 8.3 shows the probability of the delay time being larger than d at $N = 8$. The probability is shown for each crosspoint buffer entered by cells. The delay time is defined as the time from the cell's entering the crosspoint buffer until it reaches the

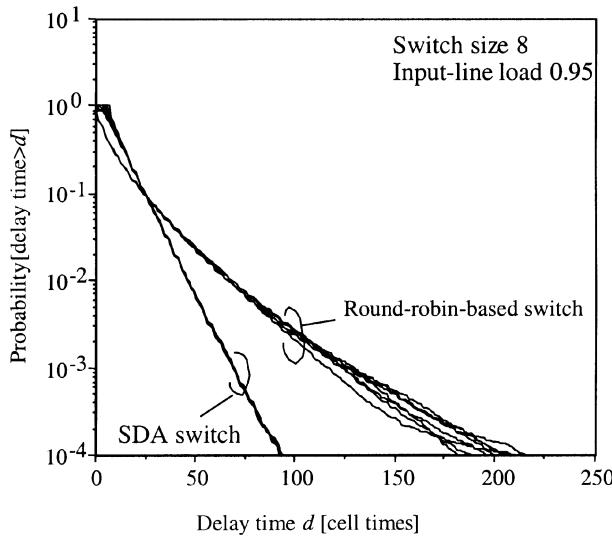


Fig. 8.3 Delay performance of SDA switch. (©1997 IEEE.)

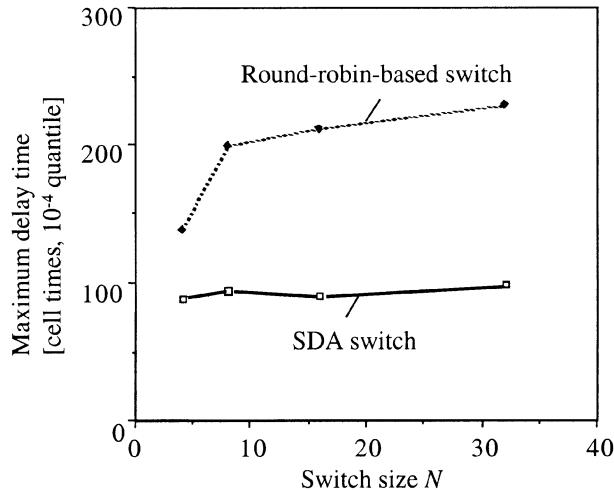


Fig. 8.4 Maximum delay time. (©1997 IEEE.)

output line. In the SDA switch, when d is more than about 10 cell times, all delay times have basically the same probability and delay time fairness is achieved. (Since it takes at least $N = 8$ cell times for the cell in the top crosspoint buffer to enter the output line, fairness is not maintained at smaller values.)

In addition, when d is larger than a certain time, the probability of the SDA switch delay time being larger than d is smaller than that of the RR switch, as shown in Figure 8.3. This is because, in the SDA switch, the cell with the largest delay time is selected.

This effect becomes clearer as N increases. Figure 8.4 shows that the maximum delay time (10^{-4} quantile) of the SDA does not change very much when N increases, while that of the RR switch increases rapidly. Furthermore, maximum SDA delay is smaller than 2^7 ($= 128$) cell times even at large N . This means that synchronous counter size is just $S = 8$, as mentioned before.

The required crosspoint buffer size of the SDA switch is smaller than that of the switch, as shown in Figure 8.5. The required buffer sizes were estimated so as to guarantee the cell loss ratio of 10^{-9} . In the SDA switch, since the required buffer sizes differ for the crosspoint buffers, Figure 8.5 shows the smallest (top crosspoint buffer) and the largest (bottom crosspoint buffer) sizes. The sizes of the intermediate crosspoint buffers lie between these two values. Because the SDA switch has shorter delay time as explained before, the queue length of the crosspoint buffer is also reduced. This is why the crosspoint buffer size of the SDA switch is less than that of the RR switch.

The switch throughput of the SDA switch increases as the switch size N increases, as shown in Figure 8.6. Since the arbitration time does not limit

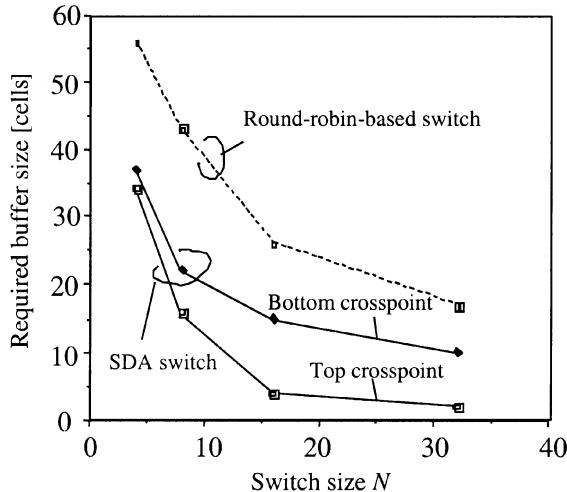


Fig. 8.5 Required buffer size. (©1997 IEEE.)

the output-line speed, the SDA switch can be expanded to achieve high switch throughput even if N is large. The switch throughput is calculated as $C_{\max} N$, where C_{\max} is the maximum output line speed.

On the other hand, the switch throughput of the RR-based switch does not increase when N becomes large. Instead it depends on the transmission delay of the control signal in a crosspoint. The RR arbitration time limits the output line speed. The RR-based switch is not expandable, because of the limitation of the RR arbitration time.

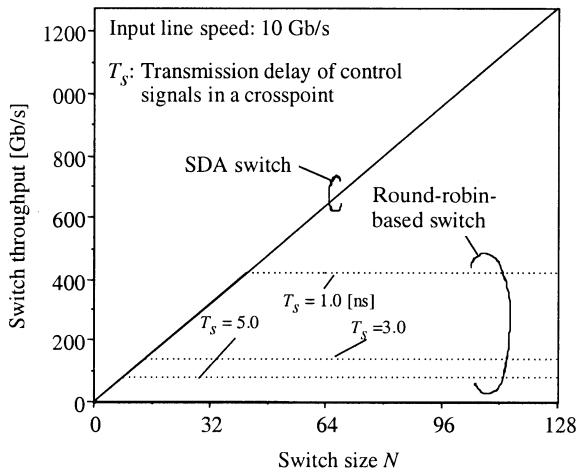


Fig. 8.6 Switch throughput vs. switch size. (©1997 IEEE.)

8.3 MULTIPLE-QOS SDA SWITCH

Section 8.2 describes an SDA switch, that can support a single QoS class. This section describes a multiple-QoS SDA (MSDA), to support multiple QoS classes by extending the concept of the SDA switch. The MSDA switch was presented in [4, 5]. We call the single-QoS SDA switch described in Section 8.2 SSDA in order to differentiate it from MSDA.

To support multiple QoS classes, a priority queuing control at each crosspoint buffer is needed. One priority queuing approach is strict priority control. Consider two priority buffers. Under the strict priority system, cells waiting in the low-priority buffer (delay-tolerant) are served only if there are no cells awaiting transmission in the high-priority (delay-sensitive) buffer. Therefore, in the strict priority discipline, the low-priority traffic effectively uses the residual bandwidth.

However, a problem occurs when we use a SSDA mechanism in a strict priority system that supports multiple QoS classes. The delay time of cells in the low-priority buffer will be very large, and the maximum delay time cannot be designed. Therefore, we cannot use the delay-time-based cell selection mechanism, as is used in the SSDA switch, for the low-priority class, due to the limitation on the number of bits for the delay measure in the cell header.

The MSDA switch was developed to support high- and low-priority classes. In order to solve the problem of a cell selection mechanism for the low-priority class, NTT introduced a distributed RR-based cell selection mechanism at each crosspoint for the low-priority class, which avoids using a synchronous counter such as is used for the high-priority class [4, 5]. The low-priority transit buffer at each crosspoint has virtual queues in accordance with the upper input ports. Cells for the low-priority class are selected by distributed ring arbitration among the low-priority crosspoint buffer and the virtual queues at the low-priority transit buffer. For the high-priority class, the same delay-time-based cell selection mechanism is used as in the SSDA switch. As a result, the proposed MSDA switch ensures fairness in terms of delay time for the high-priority class, while it ensures fairness in terms of throughput for the low-priority class.

8.3.1 MSDA Structure

This subsection describes the structure of the MSDA switch. Although we describe two priority classes in this paper for simplicity, we can easily extend the number of priority classes to more than two.

The low-priority class tolerates delay, while the high-priority class requires a small delay time. In addition, the low-priority class is supposed to be a best-effort service class such as the unspecified bit rate (UBR) class. It requires fairness in terms of throughput rather than in terms of delay time, in order to effectively use the residual bandwidth that is not used by the high-priority traffic. Therefore, it needs a cell selection mechanism that

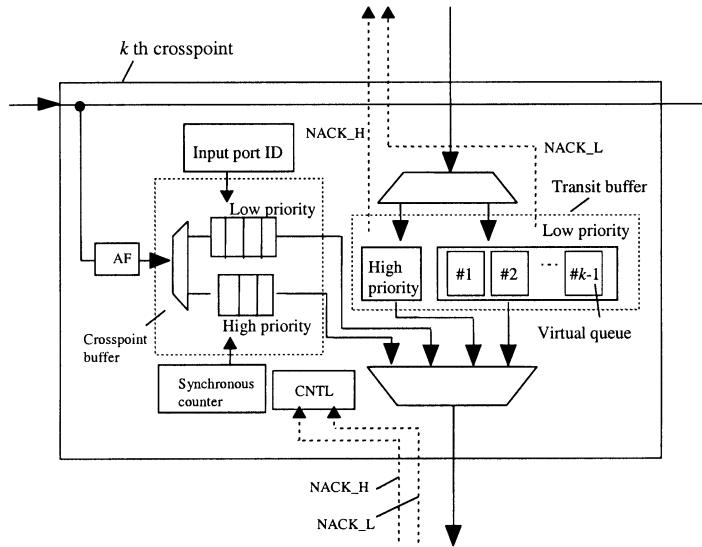


Fig. 8.7 Multi-QoS SDA (MSDA) switch structure. (©1999 IEEE.)

preserves fairness in terms of delay time for the high-priority buffer and in terms of throughput for the low-priority class.

In order to avoid the delay-time-based cell selection mechanism for the low-priority class, a distributed RR-based cell selection mechanism at each crosspoint for the low-priority class is used.

Figure 8.7 shows the structure of the MSDA switch at the k th crosspoint. The MSDA switch has a crosspoint buffer and a transit buffer, each consisting of a high-priority buffer and a low-priority buffer, an arbitration-control part (CNTL), and a selector at every crosspoint.

A cell that passes an address filter (AF) enters into either the high- or the low-priority crosspoint buffer according to its priority class. At that time, at the high-priority crosspoint buffer, the value of a synchronous counter is written into the cell overhead, as in the SSDA switch. On the other hand, at the low-priority buffer, an input port identifier (ID) is written. For example, at the k th crosspoint, the value of the input port ID is k . This is used to distinguish which input port a cell comes from. The high- and low-priority crosspoint buffers send REQ to CNTL if there is at least one cell stored in each buffer.

A cell that is transmitted from the upper crosspoint enters either the high-priority transit buffer or the low-priority crosspoint buffer according to the priority class. The low-priority transit buffer has $k - 1$ virtual queues, which are numbered $1, 2, \dots, k - 1$. A low-priority cell that has input port ID i ($1 \leq i \leq k - 1$) enters virtual queue i . The high-priority transit buffer and the low-priority transit virtual queues send REQ to CNTL if there is at

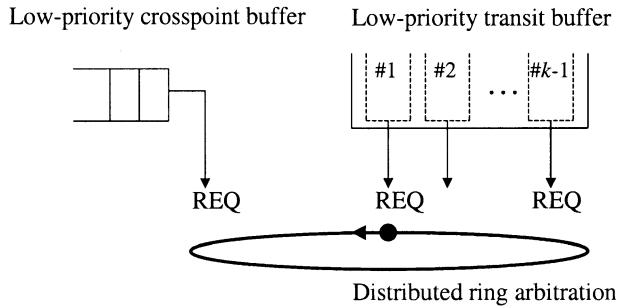


Fig. 8.8 Low-priority selection rule. (©1999 IEEE.)

least one cell stored in each buffer or virtual queue. If the high- or low-priority transit buffers are about to become full, they send not-acknowledgments NACK_H and NACK_L, respectively, to the upper CNTL.

The cell selection algorithm in the MSDA switch is as follows. If CNTL receives NACK_dH from the lower high-priority transit buffer, neither a high-priority cell nor a low-priority cell is transmitted. This is because, when the lower high-priority transit buffer is about to become full, there is no chance for the low-priority cell in the lower transit buffer to be transmitted. Low-priority cells cannot be transmitted when there is at least one high-priority REQ from the crosspoint buffer and the transit buffer. When both the high-priority crosspoint buffer and the high-priority transit buffer send REQS to CNTL, the high-priority cell selection rule used is the cell selection rule used in the SSDA switch.

Low-priority cells can be transmitted only when there are no REQS from either the high-priority crosspoint buffer or the high-priority transit buffer. If this condition is satisfied and CNTL does not receive either NACK_H or NACK_L from the lower transit buffer, then the low-priority selection rule is used. The low-priority crosspoint buffer and virtual queues in the low-priority transit buffer send REQS to CNTL as shown in Figure 8.8. Ring arbitration is executed at each crosspoint in a distributed manner. CNTL selects a cell and transmits it to the lower transit buffer.

Thus the MSDA switch achieves distributed arbitration at each crosspoint. It uses the delay-time-based cell selection rule for the high-priority buffer and the distributed RR-based cell selection rule for the low-priority class.

8.3.2 Performance of MSDA Switch

The performance of the MSDA switch is described. It is assumed that, in an $N \times N$ crosspoint-buffered switch, input traffic for both the high- and low-priority classes is random, and cells are distributed uniformly to all crosspoint buffers belonging to the same input line.

TABLE 8.1 Throughput in MSDA Switch (Case 1)

Import Port	Input Load		Throughput	
	High	Low	High	Low
1	0.060	0.050	0.060	0.050
2	0.060	0.050	0.060	0.050
3	0.060	0.150	0.060	0.050
4	0.180	0.050	0.180	0.050
5	0.060	0.050	0.060	0.050
6	0.060	0.050	0.060	0.050
7	0.060	0.050	0.060	0.050
8	0.060	0.050	0.060	0.050
Total	0.600	0.500	0.600	0.400

TABLE 8.2 Throughput in MSDA Switch (Case 2)

Import Port	Input Load		Throughput	
	High	Low	High	Low
1	0.060	0.030	0.060	0.030
2	0.060	0.030	0.060	0.030
3	0.060	0.400	0.060	0.190
4	0.180	0.030	0.180	0.030
5	0.060	0.030	0.060	0.030
6	0.060	0.030	0.060	0.030
7	0.060	0.030	0.060	0.030
8	0.060	0.030	0.060	0.030
Total	0.600	0.600	0.600	0.400

Since the high-priority is not influenced by, but does influence the low-priority class, the results of the high-priority class are the same as those of the SSDA switch. Therefore, only the performance for the low-priority buffer is presented here.

Tables 8.1 and 8.2 show that the MSDA switch keeps the fairness in terms of the throughput for the low-priority class. We present results for two traffic conditions, case 1 and case 2. The switch size was set to $N = 8$.

In case 1, the high-priority load of the fourth input port is 0.18 and that of other input ports is 0.06. The low-priority load of the third input port is 0.15 and that of other input ports is 0.05, as shown in Table 8.1. The total input load is 1.1 (0.6 + 0.5), which is overloaded. The output load, which we call the throughput, for the high-priority class is the same as the high-priority input load for each input port. The low-priority throughput of all input ports is equally divided into 0.05 to utilize the residual bandwidth. Thus, the residual bandwidth is fairly shared with all the low-priority input traffic, although its requests for bandwidth are different.

In case 2, the low-priority load of the third input port is 0.4 and that of other input ports is 0.03, as shown in Table 8.2. The high-priority input load is the same as in case 1. The total input load is 1.2 (0.6 + 0.6), which is also overloaded. The low-priority throughput for input ports except for the third input port is 0.03, which is the same as the input load, and the low-priority throughput for the third input port is 0.19, which is larger than 0.03. The low-priority throughput is first equally divided into 0.03, which satisfies the input ports except for the third. Since some bandwidth remains, the residual bandwidth is given to the third input port. Therefore, the low-priority throughput of the third input port is 0.19. This means that the MSDA switch achieves max-min fair share for the low-priority class.

REFERENCES

1. H. J. Chao, B.-S. Choe, J.-S Park, and N. Uzun, "Design and implementation of abacus switch: a scalable multicast ATM switch," *IEEE J. Select. Areas Commun.*, vol. 15, no. 5, pp. 830–843, 1997.
2. E. Oki and N. Yamanaka, "Scalable crosspoint buffering ATM switch architecture using distributed arbitration scheme," *Proc. IEEE ATM '97 Workshop*, pp. 28–35, 1997.
3. E. Oki and N. Yamanaka, "A high-speed ATM switch based on scalable distributed arbitration," *IEICE Trans. Commun.*, vol. E80-B, no. 9, pp. 1372–1376, 1997.
4. E. Oki, N. Yamanaka, and M. Nabeshima, "Scalable-distributed-arbitration ATM switch supporting multiple QoS classes," *Proc. IEEE ATM '99 Workshop*, 1999.
5. E. Oki, N. Yamanaka, and M. Nabeshima, "Performance of scalable-distributed-arbitration ATM switch supporting multiple QoS classes," *IEICE Trans. Commun.*, vol. E83-B, no. 2, pp. 204–213, 2000.
6. H. Tomonaga, N. Matsuoka, Y. Kato, and Y. Watanabe, "High-speed switching module for a large capacity ATM switching system," *Proc. IEEE GLOBECOM '92*, pp. 123–127, 1992.

CHAPTER 9

THE TANDEM-CROSSPOINT SWITCH

The HOL blocking problem in input-buffered switches can be eliminated by using the parallel-switch technique, where one switch fabric consists of multiple switch planes. The switch fabric operates at the line rate, and thus the arbitration timing is relaxed compared with the internal speedup switch architecture.

However, the parallel-switch architecture suffers from a cell-out-of-sequence problem at output ports. A resequencing circuit needs to be implemented at the output ports to ensure that cells are delivered in order. For example, timestamps can be carried in the cell headers and stored at output buffers.

A tandem-crosspoint (TDXP) switch [11, 12] developed by NTT has logically multiple crossbar switch planes. These switch planes are connected in tandem at every crosspoint. The TDXP switch achieves a high throughput without increasing the internal speed of switch fabric. It also preserves the cell-sequence order.

The remainder of this chapter is as follows. Section 9.1 briefly reviews basic input and output buffered switch architectures. Section 9.2 presents the TDXP switch architecture. Section 9.3 shows its performance. Throughout this chapter, we assume that the switch size is $N \times N$ (N input ports and N output ports). Input and output have the same line speed.

9.1 OVERVIEW OF INPUT-OUTPUT-BUFFERED SWITCHES

A switch with a crossbar structure can be easily scaled because of its modularity.

One can build a larger switch simply adding more crosspoint switch devices. In addition, the cell transmission delay in the switch is smaller than in Banyan-type switches. This is because it has the smallest number of connecting points between any input–output pair.

Variants of crossbar-type switches include the input-buffered switch and the output-buffered switch. The advantage of the former is that the operation

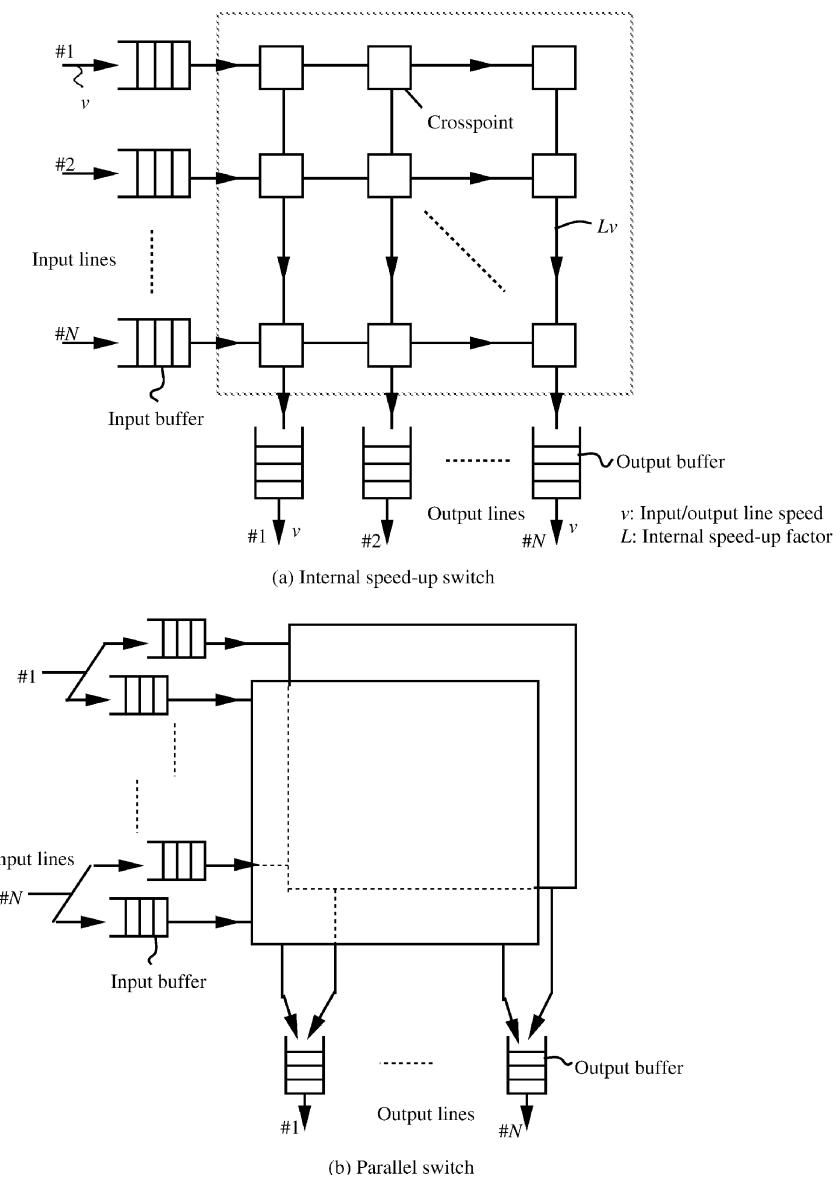


Fig. 9.1 Basic input–output-buffered switches.

speed of the switch fabric is the same as the input line rate. However, it suffers from the HOL blocking problem. The performance of the input-buffered switch was analyzed by Karol et al. [5]. They showed that, when the switch size N is infinite, the maximum throughput of the switch is 0.586, assuming that the internal speed of the switch is equal to that of input/output lines. The limitation of the maximum throughput is due to HOL blocking in the input buffers.

Several problems have to be addressed in order to improve the limited throughput of the input buffering switch [10]. One possible solution to HOL blocking is to increase the internal line speed of the switch as shown in Figure 9.1(a). Oie et al. analyzed the performance of the internal speedup switch with input and output buffers when the speedup factor is L ($1 \leq L \leq N$) [9]. Yamanaka et al. developed a high-speed switching system that has 160-Gbit/s throughput; the internal line speed was twice that of the input/output lines [14, 4, 6]. In the switch reported, the input/output speed is 10 Gbit/s, so the internal line speed is 20 Gbit/s. To realize these speeds, the switch adopted ultrahigh-speed Si bipolar devices and special high-density multichip module (MCM) techniques [6, 14]. However, for much larger throughputs this internal speedup crossbar switch architecture is not so cost-effective, given the limitation of current hardware technologies.

Another possible approach to improve the performance of crossbar-type switches is to employ a parallel switch architecture as shown in Figure 9.1(b) [13, 8]. The parallel switch consists of K identical switch planes. Each switch plane has its own input buffer and shares output buffers with other planes. The parallel switch with $K = 2$ achieves the maximum throughput of 1.0. This is because the maximum throughput of each switch plane is more than 0.586 for arbitrary switch size N . Using this concept, Balboni et al. developed an industrial 160-Gbit/s cross-connect system [1, 7]. At the input buffers, however, timestamp values must be placed in each cell header. At the output ports, cells are buffered by implementing a maximum-delay equalization mechanism in order to rebuild the cell sequences, which, due to the internal routing algorithm, can not be guaranteed [3]. Thus, this type of parallel switch requires timestamps and also requires cell sequence regeneration at the output buffers. In addition, the hardware resources needed to implement the switch are double those of a single-plane switch. Considering the implementation for much larger switches, rebuilding of the cell sequences at high-speed also makes cost-effective implementation unlikely.

9.2 TDXP STRUCTURE

9.2.1 Basic Architecture

Figure 9.2 shows the structure of the TDXP switch. It has, logically, multiple crossbar switch planes. The number of crossbar switch planes is K in general. The case with $K = 3$ is shown in Figure 9.2. The larger K is, the better the

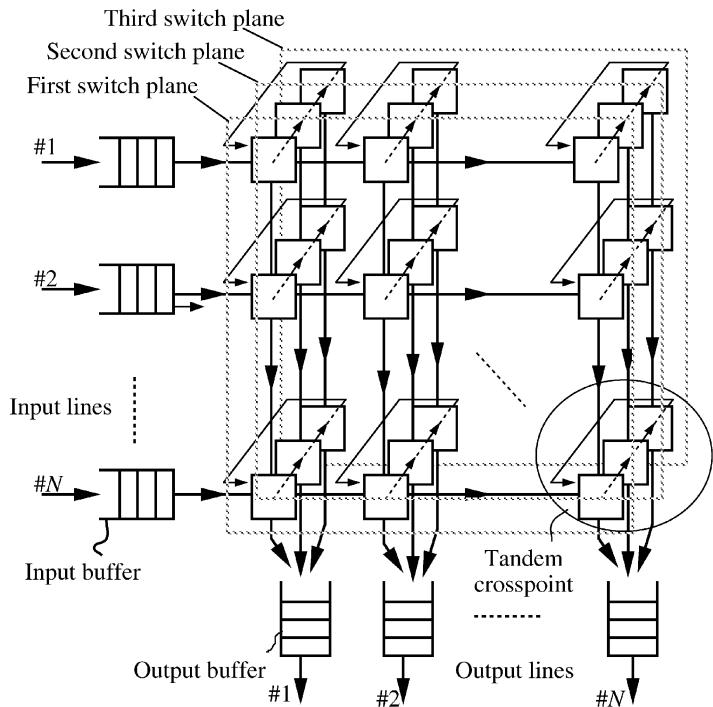


Fig. 9.2 Tandem-crosspoint switch structure. (©1997 IEEE.)

switch performance, is but at the expense of implementation cost. These switch planes are connected in tandem at every crosspoint. That is why this switch is called a TDXP switch. The internal speed in each plane is the same as the input/output line speed. In other words, each switch plane can transmit only one cell to each output port within one cell time slot. If more than one cell goes to the same output port on the same switch plane, unsuccessful cells that are not transmitted to the output port are stored in the TDXP. However, the TDXP switch that has multiple switch planes can transmit up to K cells to each output port within one time slot.

9.2.2 Unicasting Operation

The cell transmission algorithm in the TDXP switch for unicasting is first explained.

Step 1 A cell at the head of the input buffer sends a request signal (REQ) to the destination TDXP according to the routing bits written in the cell header. Then go to Step 2.

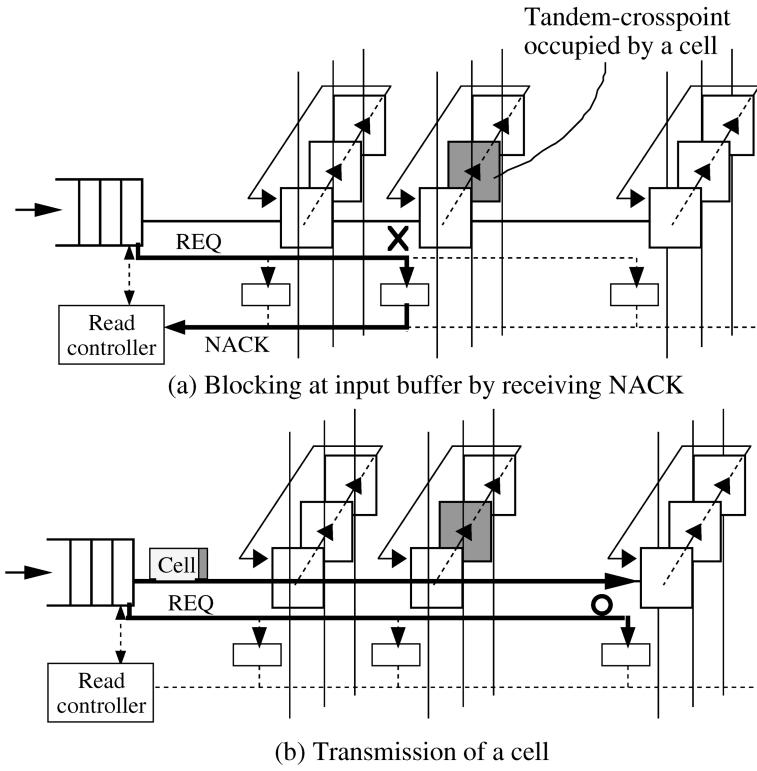


Fig. 9.3 Cell transmission mechanism of TDXP switch. (©1997 IEEE.)

Step 2 The TDXP that receives an REQ sends a not-acknowledge signal (NACK) back to the input buffer if the TDXP is already handling or buffering a cell that cannot be transmitted to the output line because of contention on the output line, as shown in Figure 9.3(a). Then go to Step 3.

Step 3 The cell at the head of the input buffer that sent the REQ is sent to the destination crosspoint on the first switch plane if NACK is not received within a certain time, as shown in Figure 9.3(b). Then go to Step 4, setting $k = 1$. Otherwise, the cell is not sent to the crosspoint; at the next cell time, go to Step 1.

Step 4 The k th crosspoint sends a request signal to an arbitration controller asking for transmission to the destination output buffer. (We refer to the crosspoint on the k th plane as the k th crosspoint.) The arbitration control on the k th plane is executed independently of that of the other planes. Ring arbitration is one possible approach. If the request is accepted by the arbitration controller, the cell is transmitted to the output buffer. Then, go to Step 5. Otherwise, the cell is moved to

the $k + 1$ th crosspoint if k is not equal to K , and k is set to $k + 1$. If k is equal to K , the cell is moved to the first crosspoint and k is set to 1. Then, go back to the beginning of Step 4 after one cell time slot.

Step 5 The cell transmitted from the TDXP is stored in the output buffer. The output buffer can receive more than one cell within one cell time.

It is noted that each TDXP needs only one cell buffer for arbitrary K . This is because, when one cell is stored in a TDXP, the following cell does not go to the same TDXP, due to the back-pressure mechanism.

To clarify the cell transmission mechanism, let us consider that $K + 1$ cells request to be transmitted to the same output port on the first switch plane. First, on the first switch plane, only one cell is transmitted to the output port, and the other K cells go to the second switch plane. At the next cell time slot, only one cell is transmitted to the output port on the second switch plane, and $K - 1$ cells go to the third plane switch. In the same way, on the K th switch plane, one cell of the remaining two is transmitted to the output port. The unsuccessful cell that cannot be transmitted to the output port goes back to the first switch plane and tries again to be transmitted to the output port, competing with other cells that request to be transmitted on the first switch plane.

Figure 9.4 shows the behavior of the cell transmission mechanism with $K = 3$, when four cells request to be transmitted to the same output port on the first switch plane at $t = 0$. In Figure 9.4, the states of only one output port are depicted. At $t = 0$, the cell at the second input port is transmitted on the first switch plane. Then, the cell at the third input port is transmitted on the second switch plane at $t = 1$, the cell at the first input port is transmitted on the third switch plane at $t = 2$, and the cell at the fifth input port is transmitted on the first plane at $t = 3$.

These procedures are executed in a pipelined manner at every cell time. Therefore, more than one cell can be transmitted to the same output buffer within one cell time slot, even though the internal line speed of each switch plane equals the input/output line speed. When $K = 3$, three cells that come from different input lines can, as the maximum case, be transmitted to the output buffer at the same time slot at $t = 2$, as shown in Figure 9.4. Thus, the TDXP switch achieves a similar result to the internal speedup switch in eliminating HOL blocking. However, the effect on HOL blocking in the TDXP switch is not exactly same as that of the internal speedup switch. This is because the TDXP switch has one cell buffer at each TDXP and employs a backpressure mechanism in the input buffers, while the internal speedup switch does not have any crosspoint buffers. A detailed discussion is given in Section 9.3, considering the effect of such a backpressure mechanism.

In addition, while a TDXP is handling a cell, the input buffer does not send the head-of-line cell to the same TDXP. The same TDXP never transmits more than one cell within the same cell time slot. Therefore, cell

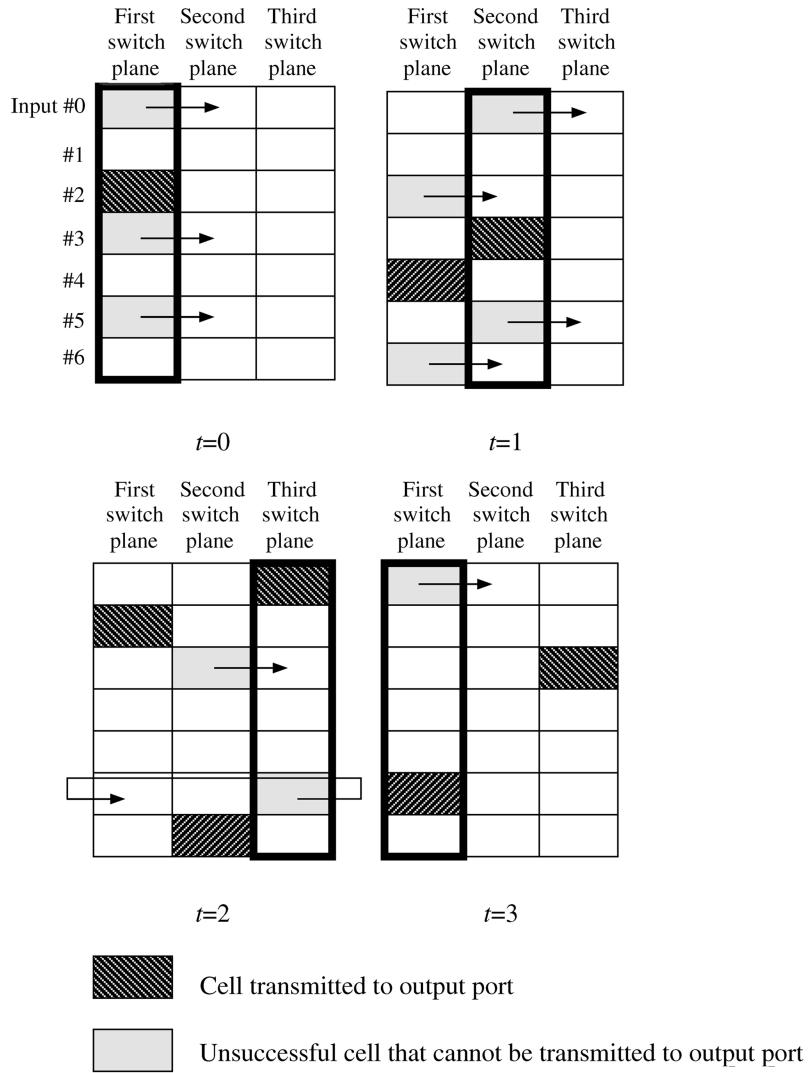


Fig. 9.4 Behavior of cell transmission mechanism ($K = 3$).

sequences are completely guaranteed, and there is no need to rebuild them at the output buffers.

Another advantage of the TDXP switch is that, although it has logically multiple crossbar switch planes, the hardware resources required are much less than for the parallel switch. This is because the parallel switch cannot share the hardware resources, while the TDXP switch can share input buffers, internal input lines, and so on. This is a significant implementation benefit.

9.2.3 Multicasting Operation

Next, we explain the multicasting mechanism in the TDXP switch. Unicast is a subset of multicast. For multicasting, Steps 1 and 3 are modified a little. The remaining procedures, Steps 2, 4, and 5, are the same as above. The modified procedures, Steps 1' and 3', for multicast are as follows.

Step 1' A cell at the head of the input buffer sends request signals (REQ) to all the destination tandem crosspoints according to the routing bits written in the cell header. The routing bits for multicasting are, for example, written using a bit-map scheme. Then go to Step 2.

Step 3' The cell at the head of the input buffer that sent REQ before is sent to all the destination crosspoints on the first switch plane, if *no* NACK is received from *any* of the crosspoints polled to the input buffer within a certain time. Then go to Step 4. If even one NACK is received by the input buffer within a certain time, the cell is not sent to any of the destination crosspoints; after the next cell time, go to Step 1'.

9.3 PERFORMANCE OF TDXP SWITCH

The performance of the TDXP switch was evaluated by event-driven computer simulation. The simulation programs were written using the C language. The performance parameters of interest were the maximum throughput, the delay time, and the buffer size required to guarantee a specified cell loss ratio. The maximum throughput is defined as the ratio of the total number of cells transmitted to output ports to the total number of offered input cells. In the estimation of the maximum throughput, all offered input loads are set at 1.0.

Assume that cell arrival at N input ports follows a Bernoulli process. When the input traffic load is ρ , an incoming cell arrives with probability ρ in a cell time, and there is no arrival with probability $1 - \rho$. The incoming cells are distributed uniformly to all output ports. The input traffic is assumed to be homogeneous, and it is distributed uniformly to all input ports. Bernoulli traffic is considered. In addition, a simple arbitration mechanism for cell output in the switch (round-robin arbitration) is established between the crosspoints belonging to the same output port.

First we present the performance of the TDXP switch for unicasting traffic. Table 9.1 shows how many tandem switch planes K are needed to obtain the maximum throughput in the TDXP switch architecture. To evaluate the maximum throughput, it is assumed that the sizes of the input and output buffers are infinite. We can see that the maximum throughput is almost saturated with $K = 2$.

We thus conclude that $K = 2$ is large enough to obtain the maximum throughput. Therefore, in the following performance evaluation of the TDXP

TABLE 9.1 Maximum Throughput Determined by the Number K of Tandem Switch Planes^a

K	Maximum Throughput				
	$N = 8$	$N = 16$	$N = 32$	$N = 64$	$N = 128$
1	0.618	0.599	0.598	0.588	0.588
2	0.922	0.948	0.970	0.983	0.990
3	0.939	0.962	0.979	0.989	0.994
4	0.942	0.965	0.981	0.990	0.995

^a N = switch size.

switch, K is set to 2. In addition, we also show the performance of the conventional internal speedup switch with input and output buffers for reference. The internal speedup factor L is set to 2 for rough equivalency [9].

The maximum throughput of the TDXP switch increases with switch size N , and is higher than that of the double-speedup switch. Figure 9.5 compares the maximum throughput with those of the internal double-speedup switch ($L = 2$) and the input buffering switch. The maximum throughput of the TDXP switch decreases with small N , say $N \leq 8$, and increases with larger N . The reason is as follows. The probability P_{suc} that two successive cells at

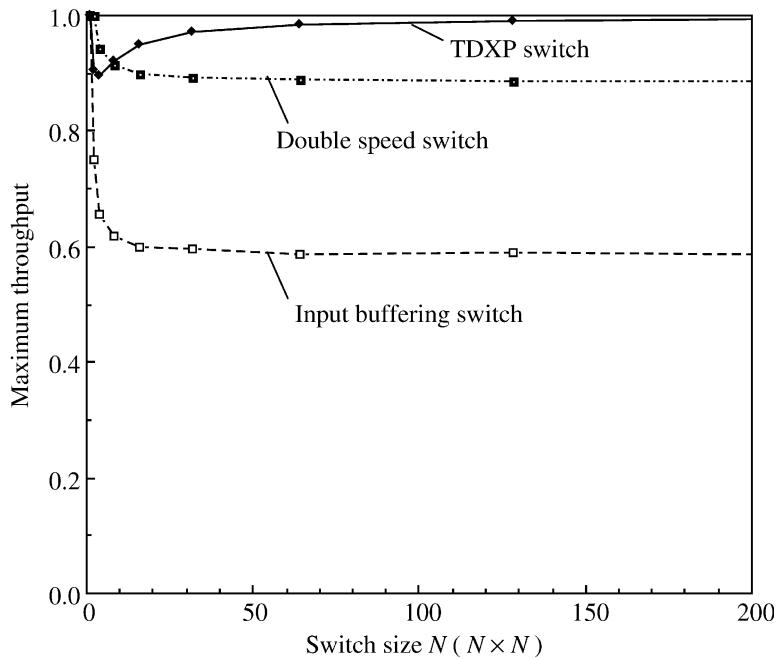


Fig. 9.5 Comparison of maximum throughput. (©1997 IEEE.)

the same input buffer are destined to the same output port is $1/N$. When N is small, P_{suc} is large. As a result, the later cell in the input buffer is likely to be blocked (NACK is received) because the first cell is still being handled in the tandem crosspoint. On the other hand, if N is large, P_{suc} is small, i.e., the blocking probability is small. The maximum throughput approaches 1.0 with $N \rightarrow \infty$, because of $P_{\text{suc}} \rightarrow 0$. In the double-speedup switch, the maximum throughput decreases to a certain value with increasing N , as is true for the input buffering switch. Thus, the maximum throughput of the TDXP switch is higher than that of the double-speedup switch when N is large.

The cell transmission delay of the TDXP switch is compared with that of the double-speedup switch with $N = 32$. The maximum delay, defined as the 99.9% value, and the average delay are shown in Figure 9.6. The maximum and average delay of the TDXP switch are almost the same as those of the double-speedup switch with small ρ (≤ 0.85). However, when the offered load ρ is larger than 0.85, the maximum and average delay values of the double-speedup switch increase strongly, because the limitation of the maximum throughput is smaller than that of the TDXP switch. In the TDXP switch, although the internal speed is the same as that of the input and output lines and a cell is buffered in a tandem crosspoint before it is

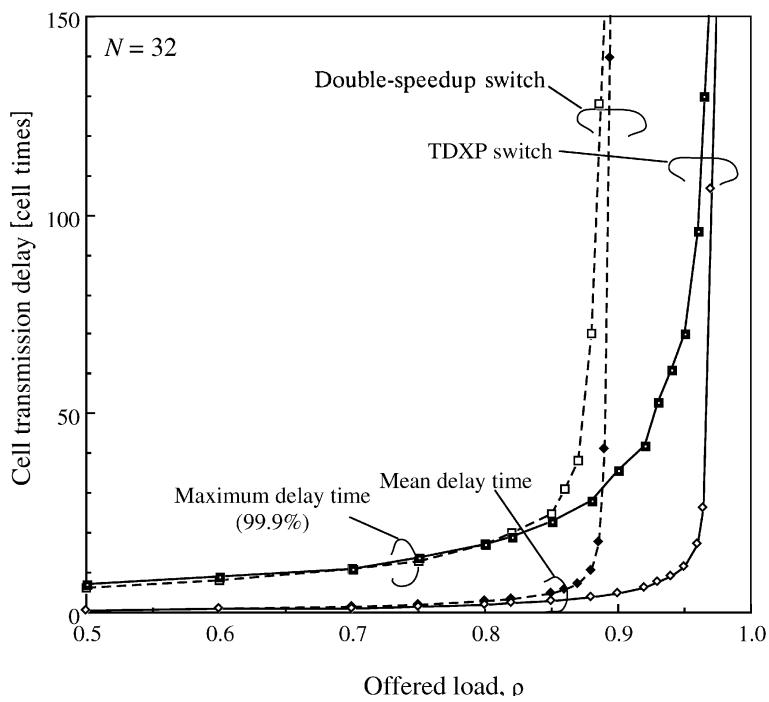


Fig. 9.6 Delay performance.

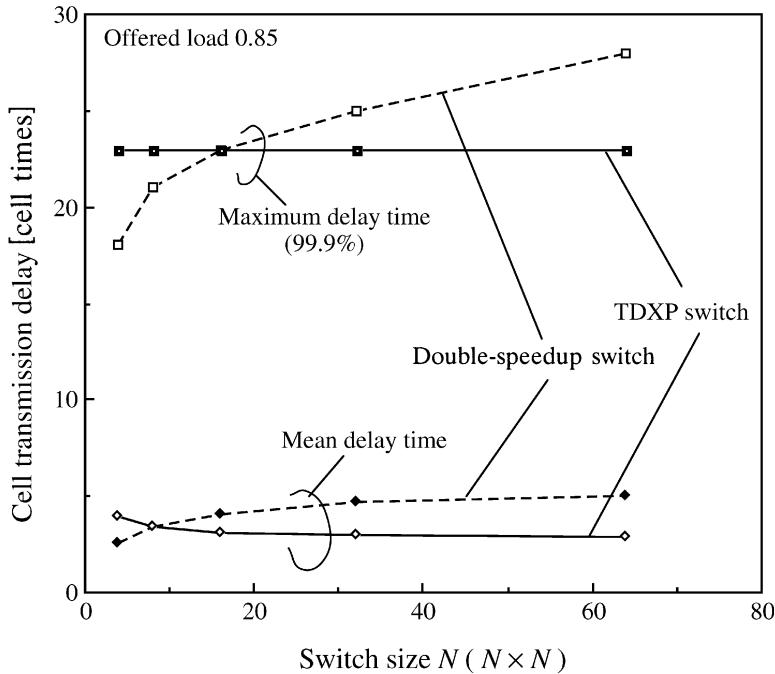


Fig. 9.7 Delay performance vs. switch size. (©1997 IEEE.)

transmitted to the output ports, the delay performance is better than that of the double-speedup switch. This is simply because the TDXP switch has the higher maximum throughput.

The switch size dependence of the cell transmission delay is shown in Figure 9.7. We can see that the maximum and average delay of the TDXP switch do not increase with N , while the delay of the double-speedup switch does. This results from the lower HOL blocking probability of the TDXP switch with large N , as explained in Figure 9.5. Thus, the TDXP switch has scalability in terms of N .

Figure 9.8 shows the required buffer sizes per port for an input buffer, an output buffer, and the sum of the input and output buffers, assuming a cell loss ratio below 10^{-9} . As is true for the delay performance results in Figure 9.6, the total size of the TDXP switch is almost the same as that of the double-speedup switch, but with offered loads higher than $\rho = 0.85$, that of the double-speedup switch increases very strongly. Therefore, buffer resources in the TDXP switch are used effectively.

Next, the results of the TDXP switch for multicasting traffic are presented. The multicast cell ratio ρ_{mc} is defined as the ratio of offered total input cells to offered multicast cells. The distribution of the number of copies is assumed to follow a binomial distribution.

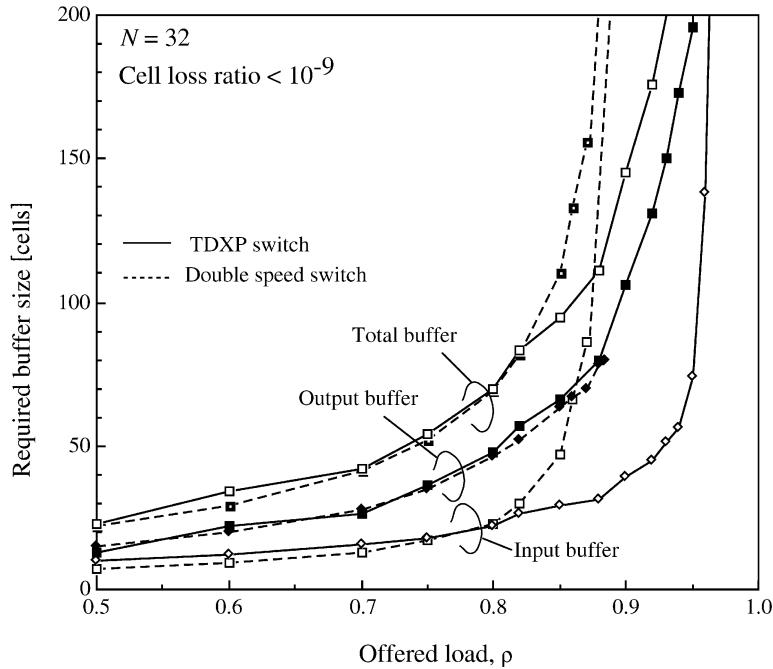


Fig. 9.8 Buffer size requirement. (©1997 IEEE.)

Figures 9.9 and 9.10 show the blocking ratio B_{block} for multicasting traffic. B_{block} is defined as the ratio of the offered input cells to the cells not transmitted from the input buffer on account of HOL blocking. In this case, the offered input traffic load is set to 1.0. The multicast mechanism employed is that presented in Section 9.2.3. The B_{block} of the TDXP switch is smaller than that of the double-speedup switch with $N = 32$. The reason is as follows. In the double-speedup switch, if at least one of the destination output ports is occupied by another cell belonging to a different input port, the multicast cell is blocked at the HOL in the input buffer. On the other hand, in the TDXP switch, even when the destined output ports are so occupied, the cell is buffered in the tandem crosspoint, and an attempt is made to send it through the next switch plane in the next time slot. Therefore, the following multicast cell in the input buffer can be transmitted to all destination tandem crosspoints as long as none of them is buffering a cell. This benefit of the TDXP switch is large when N is large and the average number of copies (ANC) is small—in other words, N/ANC is large—as shown in Figure 9.10. However, if N/ANC is small, B_{block} of the TDXP switch is higher than that of the double-speedup switch. For example, we can see such a case with $N \leq 15$ and $\text{ANC} = 4$.

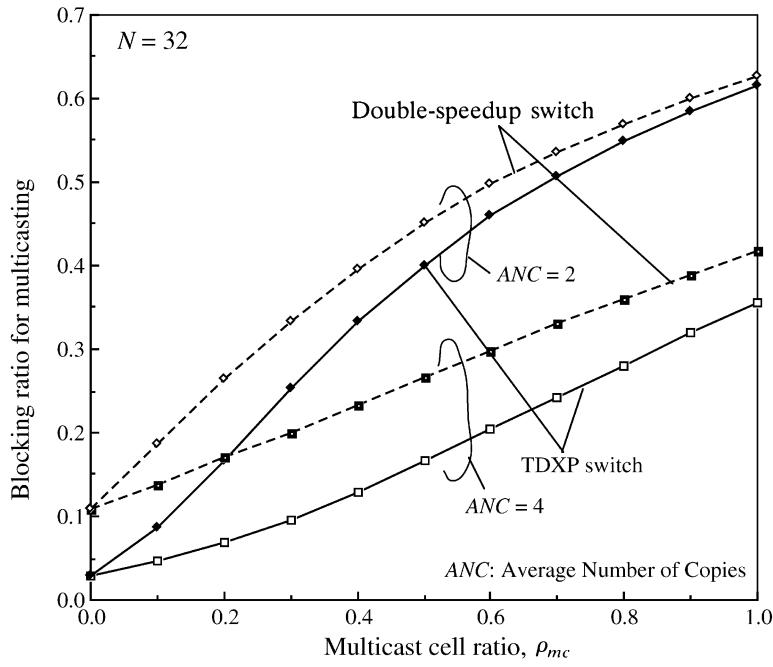


Fig. 9.9 Blocking ratio for multicasting.

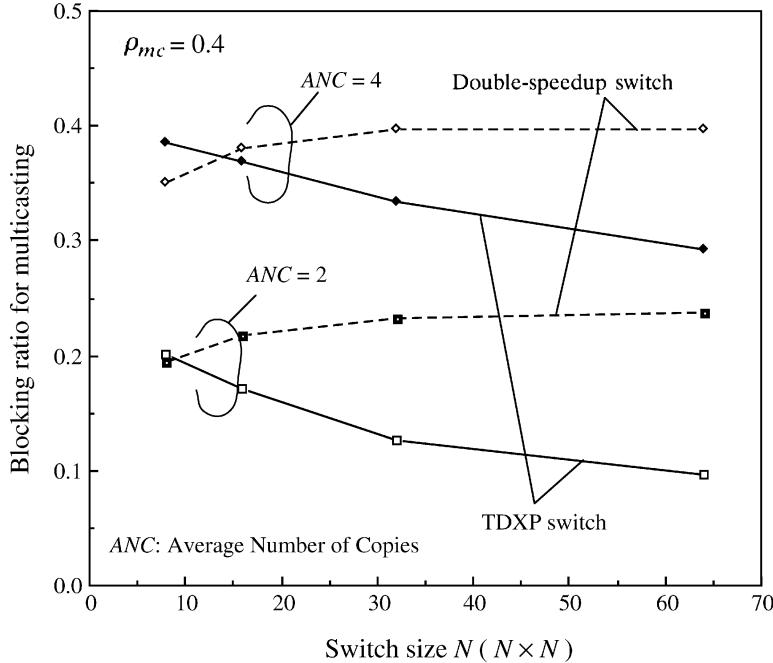


Fig. 9.10 Blocking ratio for multicasting vs. switch size and average number of copies.

REFERENCES

1. G. Balboni, M. Collivignarelli, L. Licciardi, A. Paglialunga, G. Rigolio, and F. Zizza, "From transport backbone to service platform: facing the broadband switch evolution," *Proc. ISS '95*, pp. 26, 1995.
2. T. Chaney, J. A. Fingerhut, M. Flucke, and J. S. Turner, "Design of a gigabit ATM switch," *Proc. IEEE INFOCOM '97*, pp. 2-11, 1997.
3. M. Collivignarelli, A. Daniele, P. De Nicola, L. Licciardi, M. Turolla, and A. Zappalorto, "A complete set of VLSI circuits for ATM switching," *Proc. IEEE GLOBECOM '94*, pp.134-138, 1994.
4. K. Genda, Y. Doi, K. Endo, T. Kawamura, and S. Sasaki, "A 160-Gb/s ATM switching system using an internal speed-up crossbar switch," *Proc. IEEE GLOBECOM '94*, pp. 123-133, 1994.
5. M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. COM-35, pp. 1347-1356, Dec. 1987.
6. T. Kawamura, H. Ichino, M. Suzuki, K. Genda, and Y. Doi, "Over 20 Gbit/s throughput ATM crosspoint switch large scale integrated circuit using Si bipolar technology," *Electron. Lett.*, vol. 30, pp. 854-855, 1994.
7. L. Licciardi and F. Serio, "Technical solutions in a flexible and modular architecture for ATM switching," *Proc. Int. Symp. Commun. (ISCOM '95)*, pp. 359-366, 1995.
8. P. Newman, "A fast packet switch for the integrated services backbone network," *IEEE J. Select. Areas Commun.*, vol. 6, pp.1468-1479. 1988.
9. Y. Oie, M. Murata, K. Kubota, and H. Miyahara, "Effect of speedup in nonblocking packet switch," *Proc. IEEE ICC '89*, p. 410, 1989.
10. Y. Oie, T. Suda, M. Murata, D. Kolson, and H. Miyahara, "Survey of switching techniques in high-speed networks and their performance," *Proc. IEEE INFOCOM '90*, pp. 1242-1251, 1990.
11. E. Oki, N. Yamanaka, and S. Yasukawa, "Tandem-crosspoint ATM switch architecture and its cost-effective expansion," *IEEE BSS '97*, pp. 45-51, 1997.
12. E. Oki and N. Yamanaka, "A high-speed tandem-crosspoint ATM switch architecture with input and output buffers," *IEICE Trans. Commun.*, vol. E81-B, no. 2, pp. 215-223, 1998.
13. J. S. Turner, "Design of a broadcast packet switching networks," *IEEE Trans. Commun.*, vol. 36, pp. 734-743, 1988.
14. N. Yamanaka, K. Endo, K. Genda, H. Fukuda, T. Kishimoto, and S. Sasaki, "320 Gb/s high-speed ATM switching system hardware technologies based on copper-polyimide MCM," *IEEE Trans. CPMT*, 1996, vol.18, pp. 83-91, 1995.

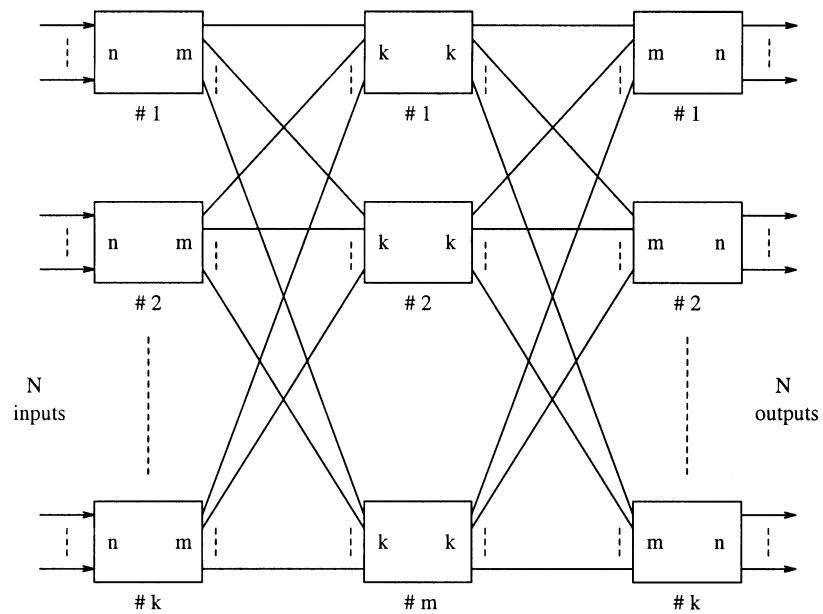
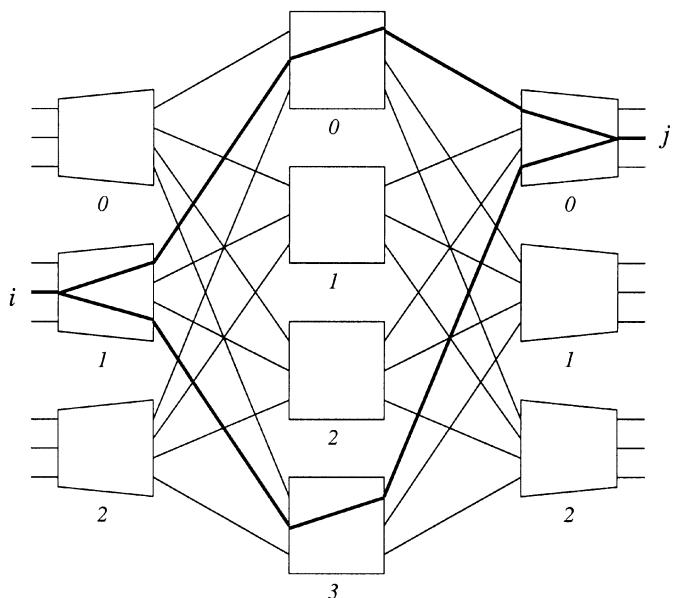
CHAPTER 10

CLOS-NETWORK SWITCHES

In Chapter 6 and Chapter 7, we have studied how to recursively construct a large switch from smaller switch modules based on the channel grouping principle. In such a construction, every input is broadcast to each first-stage module, and the input size of first-stage modules is still the same as that of the entire switch.

In this chapter we consider a different approach to build modular switches. The architecture is based on the Clos network (see Fig. 10.1). Switch modules are arranged in three stages, and every module is interconnected with every module in the adjacent stage via a unique link. In this book, the three stages are referred as input stage, middle stage, and output stage, respectively. The modules in those stages are accordingly called input modules, central modules, and output modules. Each module is assumed to be nonblocking and could be, for example, one of the crossbar switches described previously. Inputs are partitioned into groups, and only one group of inputs are connected to each input module, thereby reducing the size of each module.

One may wonder why we are not just considering a two-stage interconnection network in which every pair of modules of adjacent stages are interconnected with a dedicated link. In that case, no two cells can be simultaneously transmitted between any pair of modules, because there is just one path between them. In a Clos network, however, two cells from an input module can take distinct paths via different central modules to get to the same module at the output stage (see Fig. 10.2). The central modules in the middle stage can be looked on as routing resources shared by all input and output modules. One can expect that this will give a better tradeoff between the switch performance and the complexity.

**Fig. 10.1** A growable switch configuration.**Fig. 10.2** Routing in a Clos network.

Because of this desirable property, the Clos network was widely adopted in traditional circuit-switched telephone networks, where a path is reserved in the network for each call. If a Clos network has enough central modules, a path can always be found for any call between an idle input and an idle output. Such a property is called nonblocking. There are however two senses of nonblocking: strict and rearrangeable. In a *strictly* nonblocking Clos network, every newly arriving call will find a path across the network without affecting the existing calls. In a *rearrangeably* nonblocking network, we may have to arrange paths for some existing calls in order to accommodate a new arrival. Readers are encouraged to learn more about circuit-switched Clos networks from the related references listed at the end of this chapter.

This chapter will then focus on how the Clos network is used in packet switching. Section 10.1 describes the basic routing properties in a Clos network and formulates the routing as a scheduling problem. In the following sections, we will discuss several different scheduling approaches. Section 10.2 uses a vector for each input and output module to record the availability of the central modules. Those vectors are matched up in pairs, each of which has one vector for an input module and the other for an output module. For those pairs of modules that have a cell to dispatch between them, the two vectors will be compared to locate an available central module if any. This method is conceptually easy, but it requires a high matching speed. In Section 10.3, the ATLANTA switch handles the scheduling in a distributed manner over all central modules, each of which performs random arbitration to resolve the conflicts. Section 10.4 describes a concurrent dispatching scheme that does not require internal bandwidth expansion while achieving a high switch throughput. The path switch in Section 10.5 adopts the quasi-static virtual-path routing concept and works out the scheduling based on a predetermined central stage connection pattern, which may be adjusted when the traffic pattern changes. All of these will give us different viewpoints to look at the tradeoff between scheduling efficiency and complexity in a packet-switched Clos network.

10.1 ROUTING PROPERTIES AND SCHEDULING METHODS

A three-stage Clos network is shown in Figure 10.1. The first stage consists of k input modules, each of dimension $n \times m$. The dimension of each central module in the middle stage is $k \times k$. As illustrated in Figure 10.2, the routing constraints of Clos network are briefly stated as follows:

1. Any central module can only be assigned to *one* input of each input module, and *one* output of each output module;
2. Input i and output j can be connected through any central module;
3. The number of alternative paths between input i and output j is equal to the number of central modules.

For the $N \times N$ switch shown in Figure 10.1, both the inputs and the outputs are divided into k modules with n lines each. The dimensions of the input and output modules are $n:m$ and $m:n$, respectively, and there are m middle-stage modules, each of size $k \times k$. The routing problem is one of directing the input cells to the irrespective output modules without path conflicts. For every time slot, the cell traffic can be written as

$$T = \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,k} \\ \vdots & \vdots & & \vdots \\ t_{k,1} & t_{k,2} & \cdots & t_{k,k} \end{bmatrix},$$

where $t_{i,j}$ represents the number of cells arriving at the i th input module destined for the j th output module. The row sum is the total number of cells arriving at each input module, while the column sum is the number of cells destined for each output module, and they are given by

$$R_i = \sum_{j=1}^k t_{i,j} \leq n \leq m, \quad i = 1, 2, \dots, k, \quad (10.1)$$

$$S_j = \sum_{i=1}^k t_{i,j} \leq n, \quad j = 1, 2, \dots, k. \quad (10.2)$$

Since at most m cells for each output group are considered, the column sum constraint can then be rewritten as

$$S_j \leq m, \quad j = 1, 2, \dots, k. \quad (10.3)$$

The routing requirement is to assign each cell a specific path through the first two stages such that no two cells from an input module will leave at the same outlet of the input module, and no two cells destined for the same output module will arrive at the same inlet of the output module. Such an assignment is guaranteed by the nonblocking property of the three-stage Clos network for $m \geq n$.

Let us label the set of outlets from each input module by A_i ($i = 1, \dots, k$) and the set of inlets into each output module by B_j ($j = 1, \dots, k$). Each A_i or B_j contains exactly m elements denoting m physical lines. If these elements are viewed as time slots instead of physical lines, then the first two stages can be transformed into a $k \times k$ time-space-time (TST) switch (see Figure 10.3), where each input frame (A_i) and output frame (B_j) has m slots. The assignment problem is now seen as identical to that of a classic TST switch, and the necessary and sufficient conditions to guarantee a perfect or optimal schedule are satisfied by (10.1) and (10.3) [3]. For the assignment of an arbitrary cell at a given input module to its output module, the status of its

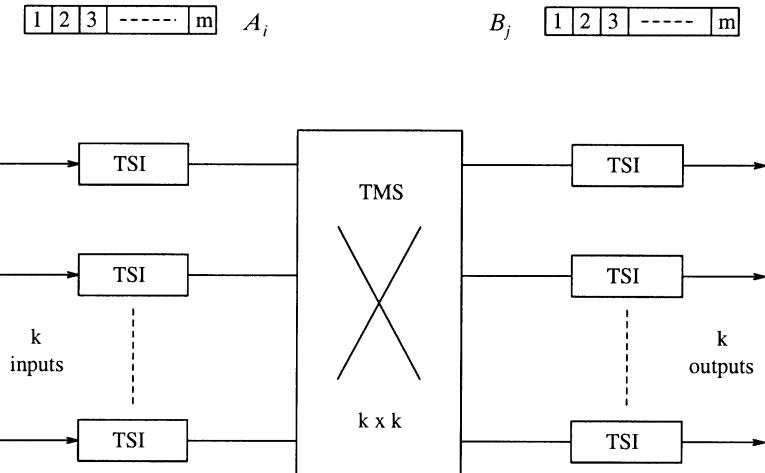


Fig. 10.3 A TST switch representation after the space-to-time transformation. (©1992 IEEE.)

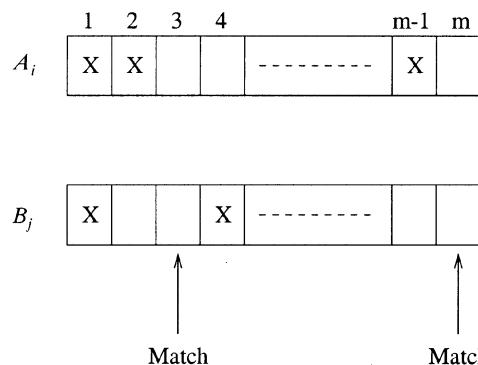


Fig. 10.4 Matching of A_i and B_j (X = busy, blank = open). (©1992 IEEE.)

A_i and the corresponding B_j may be examined, as shown in Figure 10.4, where any vertical pair of idle slots (representing physical lines) could be chosen as a valid interconnect path. An overall schedule is perfect if all cells are successfully assigned without any path conflict. However, optimal scheduling requires global information on the entire traffic T , and the implementation tends to be very complex.

10.2 A SUBOPTIMAL STRAIGHT MATCHING METHOD FOR DYNAMIC ROUTING

A crucial requirement on the routing or scheduling algorithm is that it must run extremely fast, because cells have to be assigned paths on a slot-by-slot basis. A distributed routing algorithm is considered, which is suboptimal in performance, but runs fast and is simple to implement in hardware.

Let us consider a particular input module, say the first one. It may have a total of n cells arriving at a time, and the assignment hardware has to finish assigning these n cells within some prescribed duration Δ less than a time slot. We further divide Δ into k mini-slots of length Δ/k , and now focus on the assignment operation inside a minislot. During every mini-slot, the B_j of a specific output module is made available for assignment to an input module A_i , i.e., matching of A_i with B_j . Note that there are at most n matches needed in each mini-slot, as there are at most n input cells at an input module. While A_i is matched against B_j , simultaneously $A_{i+1 \bmod k}$ is matched against $B_{j+1 \bmod k}$, and so on, thereby permitting parallel assignments. In the next mini-slot, each B_j is moved on to the next A_i (i.e., $A_{i+1 \bmod k}$) in a cyclic manner (see Fig. 10.5). Since there are k output modules, a total of k mini-slots are necessary in each slot.

Since the switch architecture uses output queuing, it has the best possible delay-throughput performance [5]. Consequently, we only need to compute the lost-cell probability. There are two sources of cell loss in the switch. First, cells are dropped if too many simultaneously arrive destined for an output group (the loss due to knockout). Second, additional cells are dropped if the distributed, suboptimal routing algorithm cannot schedule a path through the switch. The analysis of this scheduling loss is complex and can be found in [5]. Here we just highlight the main results for the switch configuration without middle-stage trunking.

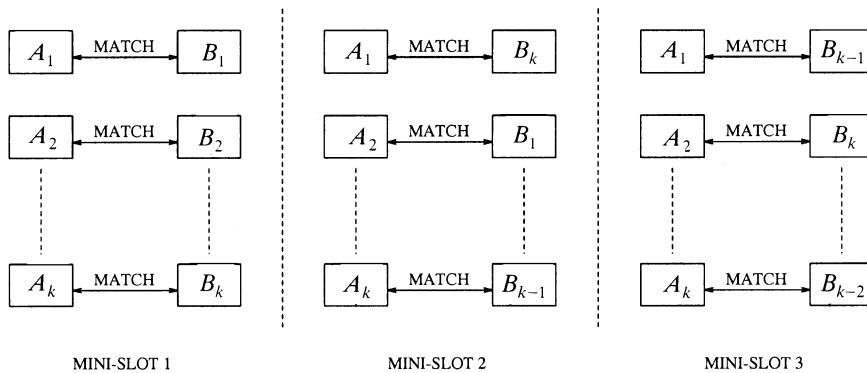


Fig. 10.5 Parallel assignments in a cyclic manner from mini-slot to mini-slot.

Assuming the cells have independent, uniform destination address distributions, the loss probability increases from one mini-slot to the next as more and more cells are assigned routing paths in a given time slot. Upper bounds on the lost-cell probability for the worst case input–output pairs (the last to be scheduled in a time slot) are derived in [5].

For a group size $n = 16$, Figure 6.11 shows the worst case cell loss probability as a function of m , the number of simultaneous cells accepted, under various loads. The loss due to knockout in Figure 6.11 is for the worst case (last) input, and is given by (a generalization of (4) in [16])

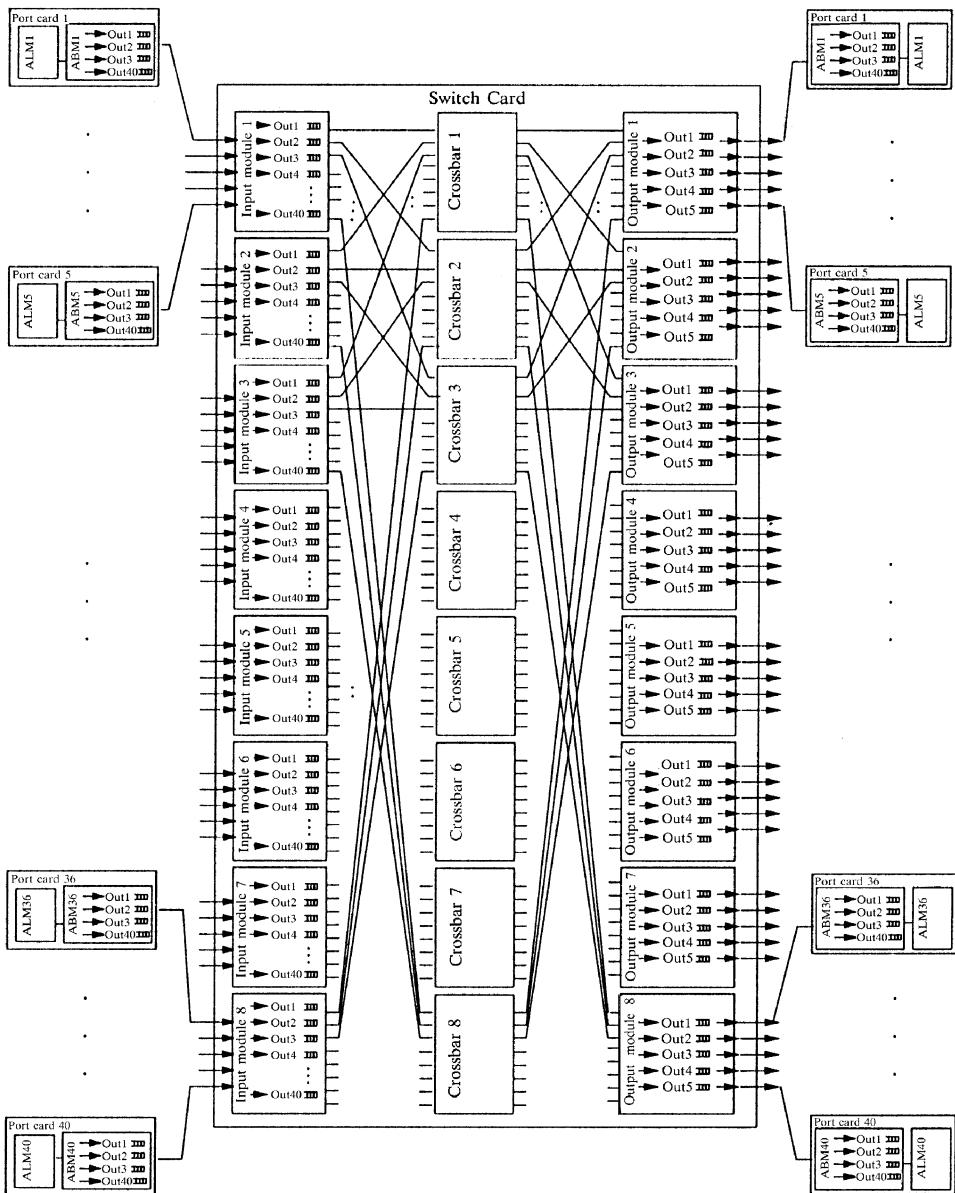
$$\Pr[\text{cell loss for worst case input}] = 1 - \sum_{k=0}^m \frac{(np)^k e^{-np}}{k!}.$$

Note that $m = 35$ is large enough to keep the cell loss probability below 10^{-6} for an 80% load, compared to a value of $m = 34$ required by the knockout loss alone. Each increase in m results in approximately an order of magnitude decrease in the lost-cell probability, and both sources of loss can be made negligible small. Since the knockout loss and knockout + scheduling loss curves are so close to each other, an optimal routing algorithm would not provide much improvement; the lost-cell probability cannot be less than that indicated by the generalized knockout principle. In addition, middle-stage trunking can provide only marginal improvement over the results presented in Figure 6.11.

10.3 THE ATLANTA SWITCH

The ATLANTA switch architecture was developed by Bell Labs [10]. The switch fabric is a three-stage multimodule memory–space–memory (MSM) arrangement. The MSM configuration uses buffers in the input and output stages, while the second stage is bufferless. A simple distributed self-routing algorithm is used to dispatch cells from the input to the output stage. Although cells are routed individually and multiple paths are provided from each input to each output, cell sequence is preserved due to its bufferless second stage. Selective backpressure is used from the output to the input buffers in the fabric, so the required buffers in the output stage are also relatively small.

The ATLANTA architecture provides optimal support of multicast traffic. Cells belonging to a multicast virtual circuit are always replicated according to a minimum multicast tree, that is, they are replicated as far downstream as possible in the switch; this minimizes the amount of resources required to sustain the expansion in traffic volume internally to the switch due to multicasting. A single copy of a multicast cell is locally stored in each buffer, and replicated (if necessary) only when the cell is sent to the following stage in the switch or to its desired destinations.



ALM : ATM Layer Manager

ABM : ATM Buffer Manager

Fig. 10.6 Schematic configuration of a 40×40 multistage ATLANTA switch. (©1997 IEEE.)

In the following, we describe the operation of the MSM switch fabric with reference to the specific 40×40 configuration shown in Figure 10.6. The operation of configurations of other sizes can be easily inferred from this discussion.

10.3.1 Basic Architecture

The MSM configuration is based on three main principles:

- By using buffers in the first stage to store cells that cannot be routed to the output buffers at a given time, the number of paths necessary for nonblocking behavior can be greatly reduced.
- By using a bufferless center stage, cells belonging to the same virtual circuit can be routed individually without affecting the cell sequence.
- By using selective backpressure from the output buffers to the input buffers, buffers can be located where it is most economical.

Under these design principles in the ATLANTA switch, a memory switch is used to implement the switching modules in the first and the third stages, while crossbars are implemented in the second stage. Each module in the first and third stages must be connected to all crossbars. All interconnection lines between adjacent stages have the same rate as the input and output ports. To realize nonblocking in the MSM configuration, it is well known that its internal capacity must be higher than the aggregate capacity of the input ports. We call this expansion. The required expansion is achieved by connecting fewer than eight ports to each input and output module. In the 40×40 configuration of Figure 10.6, five ports are connected to each edge module for an expansion factor of 5:8. The expansion ratio is 1.6 ($= 8/5$). Each module in the first stage maintains 40 groups of queues; each group corresponds to one of the output ports in the switch. Each module in the third stage manages a number of groups equal to the number of ports connected to that module (in this case five).

10.3.2 Distributed and Random Arbitration

In order to minimize the required expansion, an efficient routing algorithm is necessary to route cells from the input to the output modules. Intuitively, the idea is that the fabric is nonblocking as long as the equivalent service capacity (i.e., the maximum switching capacity provided by the expansion and the routing algorithm) in the input queues is higher than the aggregate input capacity of those queues. A practical constraint for such a system to be cost-effective is that the routing algorithm must be fully distributed and independently run by each input module. Below is the novel concurrent dispatching algorithm that is used in the ATLANTA architecture.

The concurrent dispatching works as follows. In each time slot, each input module in the first stage selects up to eight cells to be served in that time slot. The selection process over the 40 groups of queues uses a two-level weighted-round-robin mechanism.¹ Once the cells are selected, each input module sends up to eight bids to the crossbars, one for each crossbar. A bid contains the desired destination and service priority of one of the selected cells. Since there is no coordination among the input modules, a crossbar can receive more than one bid at a time for the same output module. In case of conflict between two or more bids, the crossbar selects one as the winning bid. In selecting the winning bid, and generally in determining whether a bid is successful or not, the crossbar takes into account whether or not the specific queue in the output module requested by each bid has available buffer space (the third-stage modules continuously send backpressure information to the crossbars informing them of the availability of buffers for each queue), and never declares successful a bid that requests a queue with no available buffer space. Then the crossbar sends a feedback signal to the input modules, informing each module whether or not the bid was successful, and in the latter case whether the bid was unsuccessful because of lost contention in the crossbar or because of selective backpressure from the output modules.

If the bid was successful, in the following time slot the input module transmits the corresponding cell through the crossbar; in the same time slot, the input module also selects another cell and initiates a new bidding process for it on that crossbar. If the bid was unsuccessful because of lost contention in the crossbar, the input module again sends that same bid to the crossbar in the following time slot. If the bid was unsuccessful because of backpressure from the output modules, the input module selects a different cell from the buffer and initiates a bidding process for it in the following time slot.

10.3.3 Multicasting

Multicast cells in the ATLANTA architecture are treated very similarly to unicast cells. They are always replicated as far downstream as possible in the switch. In the first stage, a multicast cell is replicated in only m_1 copies, where m_1 is the number of different modules in the third stage that the cell is destined for. Referring to the situation depicted in Figure 10.7, for example, cells belonging to connection V with the destinations highlighted in the figure would only be replicated three times in the input module. In particular, a cell that is destined to multiple ports in an output module is only queued in one of the corresponding queues. See Figure 10.7. Cells from connection V are only queued in the queues corresponding to ports 1, 11, 13, 15, 39, and 40. Then, a multicast cell is further replicated to the desired output ports in each module in the third stage.

¹Cells in each group are further classified into several categories of different service priority.

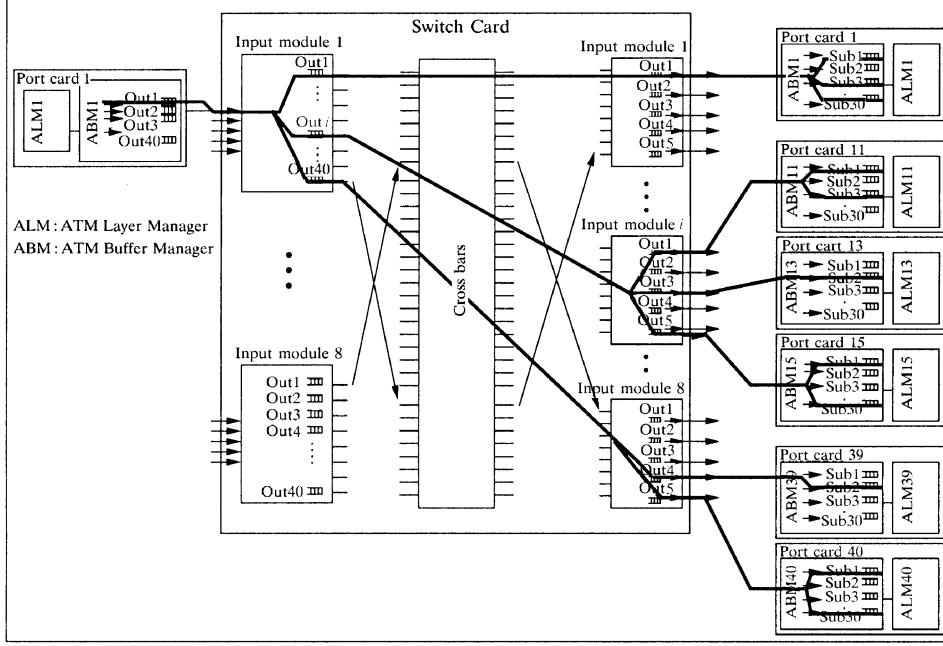


Fig. 10.7 Example of a multicast connection in a 40×40 ATLANTA switch with multistage fabric. The minimum multicast tree for cells of that multicast connection is highlighted. (©1997 IEEE.)

In each edge module throughput the switch, only one copy is kept in memory for each multicast cell. A cell is replicated only when it is to be transmitted to the next stage. An appropriate destination bit map is stored in the cell header to control the replication through the fabric. The whole replication mechanism is therefore embedded in the self-routing structure of the switch.

10.4 THE CONTINUOUS ROUND-ROBIN DISPATCHING SWITCH

The ATLANTA switch described in Section 10.3 is not able to achieve a high throughput unless the internal bandwidth is expanded, because contention at the second stage cannot be avoided. The relationship between the internal expansion ratio and switch size in the ATLANTA switch was analytically derived in [14]. To achieve 100% throughput by using the random dispatching scheme, the internal expansion ratio is set to about 1.6 when the switch size is large [10]. This makes it difficult to implement a high-speed switch in a cost-effective manner.

It is a challenge to find a cost-effective dispatching scheme that is able to achieve a high throughput in Clos-network switches, without allocating any buffers in the second stage to avoid the out-of-sequence problem and without expanding the internal bandwidth.

A solution to the challenge was introduced in [14], where a round-robin-based dispatching scheme, called the concurrent round-robin dispatching (CRRD) scheme, was proposed for a Clos-network switch. The basic idea of CRRD is to use the desynchronization effect [13] in the Clos-network switch. The desynchronization effect has been studied using simple scheduling algorithms as *i*SLIP [13, 11] and dual round-robin matching (DRRM) [1, 2] in an input-queued crossbar switch. CRRD provides high switch throughput without expanding the internal bandwidth, while the implementation is simple because only simple round-robin arbiters are employed. We showed that CRRD achieves 100% throughput under uniform traffic.

10.4.1 Basic Architecture

Figure 10.8 shows the CRRD switch. The terminology used in this section is as follows:

IM	Input module at the first stage.
CM	Central module at the second stage.
OM	Output module at the third stage.
n	Number of input ports (output ports) in each IM (OM)
k	Number of IMs or OM
m	Number of CMs.
i	IM number, where $0 \leq i \leq k - 1$.
j	OM number, where $0 \leq j \leq k - 1$.
h	Input port (IP) or output port (OP) number in each IM/OM, respectively, where $0 \leq h \leq n - 1$.
r	Central module (CM) number, where $0 \leq r \leq m - 1$.
$\text{IM}(i)$	i th IM.
$\text{CM}(r)$	r th CM.
$\text{OM}(j)$	j th OM.
$\text{IP}(i, h)$	h th input port at $\text{IM}(i)$.
$\text{OP}(j, h)$	h th output port at $\text{OM}(j)$.
$\text{VOQ}(i, v)$	Virtual output queue (VOQ) at $\text{IM}(i)$ that stores cells destined for $\text{OP}(j, h)$, where $v = hk + j$ and $0 \leq v \leq nk - 1$.
$G(i, j)$	VOQ group at $\text{IM}(i)$ that consists of n $\text{VOQ}(i, j, h)$ s.
$L_i(i, r)$	Output link at $\text{IM}(i)$ that is connected to $\text{CM}(r)$
$L_c(r, j)$	Output link at $\text{CM}(r)$ that is connected to $\text{OM}(j)$.

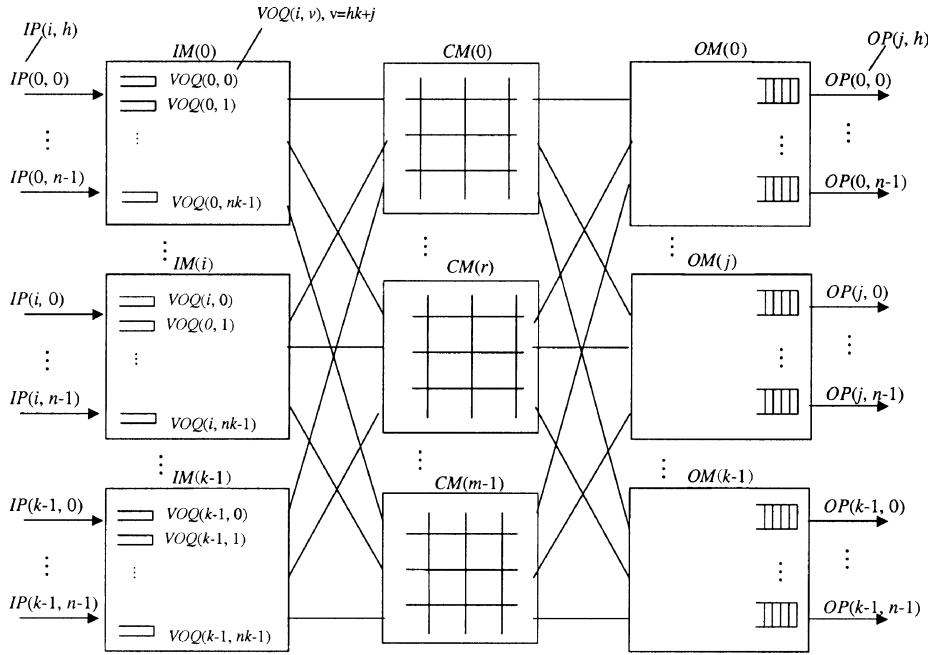


Fig. 10.8 CRRD switch with virtual output queues in the input modules.

The first stage consists of k IMs, each of which is $n \times m$. The second stage consists of m bufferless CMs, each of which is $k \times k$. The third stage consists of k OMs, each of which is $m \times n$.

IM(i) has nk VOQs to eliminate HOL blocking [10]. VOQ(i, v) stores cells that go from IM(i) to OP(j, h) at OM(j), where $v = hk + j$. A VOQ can receive, at most, n cells from n IPs in each cell time slot. The HOL cell in each VOQ can be selected for transmission across the switch through CM(r) in each time slot. This ensures that cells are transmitted from the same VOQ in sequence.

IM(i) has m output links. An output link $L_i(i, r)$, is connected to CM(r).

CM(r) has k output links, denoted as $L_c(r, j)$, and it is connected to k OMs, denoted as OM(j).

OM(j) has n output ports, denoted as OP(j, h), and has an output buffer. Each output buffer receives at most m cells in one time slot, and each output port of an OM forwards one cell in a FIFO manner to the output line.

10.4.2 Concurrent Round-Robin Dispatching Scheme

Figure 10.9 illustrates the detailed CRRD algorithm. To determine the matching between a request from VOQ(i, v) and the output link $L_i(i, r)$, CRRD adopts an iterative matching within IM(i). An IM has m output link arbiters, each of which is associated with each output link, and each VOQ has a VOQ arbiter as shown in Figure 10.9.

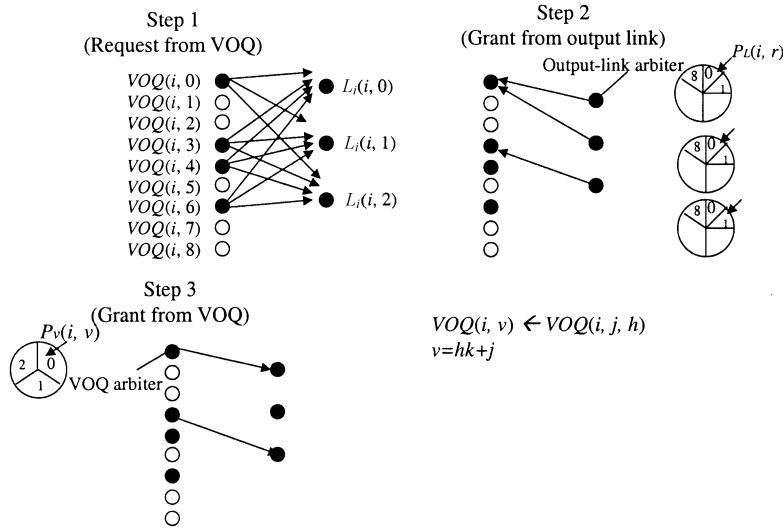


Fig. 10.9 CRRD scheme.

We consider two phases for dispatching from the first stage to the second stage. In phase 1, at most m VOQs are selected as candidates, and the selected VOQ is assigned to an IM output link. A request that is associated with this output link is sent from IM to CM. This matching between VOQs and output links is performed only within the IM. In phase 2, each selected VOQ that is associated with each IM output link sends a request from IM to CM. CMs respond with the arbitration results to IMs so that the matching between IMs and CMs can be done.

- *Phase 1: Matching within an IM*
- *First iteration*
 - *Step 1:* Each nonempty VOQ sends a request to every output link arbiter, each of which is associated with $L_i(i, r)$, where $0 \leq i \leq k - 1$ and $0 \leq r \leq m - 1$.
 - *Step 2:* Each output link $L_i(i, r)$, where $0 \leq i \leq k - 1$ and $0 \leq r \leq m - 1$, independently seeks a request among nk nonempty VOQs. Each output link arbiter associated with $L_i(i, r)$ has its own pointer $P_L(i, r)$, where $0 \leq i \leq k - 1$ and $0 \leq r \leq m - 1$. The output link arbiter seeks one nonempty VOQ request from the $P_L(i, r)$ in a RR fashion. Each output link arbiter sends the grant to a requesting VOQ. Each VOQ has its own RR arbiter and one pointer $P_V(i, v)$, where $0 \leq v \leq nk - 1$, to choose one output link. The VOQ arbiter seeks one grant among several grants that are given by the output link arbiters from the position of $P_V(i, v)$.
 - *Step 3:* The VOQ that chooses one output link $L_i(i, r)$ by using the RR arbiter sends the grant to the selected output link. Note that

the pointer $P_L(i, r)$ that is associated with each output link and the pointer $P_V(i, v)$ that is associated with each VOQ are updated to one position after the granted position, but only if they are matched and the request is also granted by the CM in phase 2.

- *i*th iteration ($i > 1$)
 - *Step 1*: Each unmatched VOQ at the previous iterations sends a request to all the output link arbiters again.
 - *Steps 2 and 3*: Follow the same procedure as in the first iteration.

Phase 2: Matching Between IM and CM

- *Step 1*: After phase 1 is completed, output link $L_i(i, r)$ sends the request to the CM. Then contention control in the CM is performed. CM(r) has k pointers $P_c(r, j)$, where $0 \leq r \leq m - 1$ and $0 \leq j \leq k - 1$, each of which corresponds to each OM(j). The CM makes its arbitration using the pointer $P_c(r, j)$ in a RR fashion, and sends the grants to $L_i(i, r)$ of IM(i). The pointer $P_c(r, j)$ is updated when the CM sends the grant to the IM.
- *Step 2*: If the IM receives the grant from the CM, it sends a corresponding cell from that VOQ in the next time slot. Otherwise, the IM will not send a cell in the next time slot. The request that is not granted from the CM will be dispatched again at the next time slot, because the pointers that are related to the ungranted requests are not updated.

The CRRD algorithm has to be completed within one time slot to provide the matching result in every time slot.

Figure 10.9 shows an example with $n = m = k = 3$, where CRRD is operated at the first iteration in phase 1. At step 1, VOQ($i, 0$), VOQ($i, 3$), VOQ($i, 4$), and VOQ($i, 6$), which are nonempty VOQs, send requests to all the output link arbiters. At step 2, output link arbiters associated with $L_i(i, 0)$, $L_i(i, 1)$, and $L_i(i, 2)$, select VOQ($i, 0$), VOQ($i, 0$), and VOQ($i, 3$), respectively, according to their pointers' positions. At step 3, VOQ($i, 0$) receives two grants from both output link arbiters of $L_i(i, 0)$ and $L_i(i, 1)$, selects $L_i(i, 0)$ by using its own VOQ arbiter, and sends a grant to the output link arbiter of $L_i(i, 0)$. Since VOQ($i, 3$) receives one grant from an output link arbiter $L_i(i, 2)$, it sends a grant to the output link arbiter. With one iteration, $L_i(i, 1)$ cannot be matched with any nonempty VOQs. At the next iteration, the matching between unmatched nonempty VOQs and $L_i(i, 1)$ will be performed.

10.4.3 Desynchronization Effect of CRRD

The ATLANTA switch suffers contention at the CM [10]; CRRD decreases the contention because pointers $P_V(i, v)$, $P_L(i, r)$, and $P_c(r, j)$, are desynchronized.

	T	0	1	2	3	4	5	6	7
$IM(0)$	$P_V(0, 0)$	0	1	0	0	0	1	0	0
	$P_V(0, 1)$	0	0	1	0	0	0	1	0
	$P_V(0, 2)$	0	0	0	1	0	0	0	1
	$P_V(0, 3)$	0	0	0	0	1	0	0	0
$IM(1)$	$P_V(1, 0)$	0	0	1	0	0	0	1	0
	$P_V(1, 1)$	0	0	0	1	0	0	0	1
	$P_V(1, 2)$	0	0	0	0	1	0	0	0
	$P_V(1, 3)$	0	0	0	0	0	1	0	0
$IM(0)$	$P_L(0, 0)$	0	1	2	3	0	1	2	3
	$P_L(0, 1)$	0	0	1	2	3	0	1	2
$IM(1)$	$P_L(1, 0)$	0	0	1	2	3	0	1	2
	$P_L(1, 1)$	0	0	0	1	2	3	0	1
$CM(0)$	$P_C(0, 0)$	0	1	0	1	0	1	0	1
	$P_C(0, 1)$	0	0	1	0	1	0	1	0
$CM(1)$	$P_C(1, 0)$	0	0	1	0	1	0	1	0
	$P_C(1, 1)$	0	0	0	1	0	1	0	1

The request is granted by CM.

Fig. 10.10 Example of desynchronization effect in CRRD ($n = m = k = 2$).

How the pointers are desynchronized is demonstrated by using a simple example. Let us consider $n = m = k = 2$ as shown in Figure 10.10. We assume that every VOQ is always occupied with cells. Each VOQ sends a request to be selected as a candidate at every time slot. All the pointers are set to be $P_V(i, v) = 0$, $P_L(i, r) = 0$, and $P_c(r, j) = 0$ at the initial state. Only one iteration in phase 1 is considered here.

At time slot $T = 0$, since all the pointers are set to 0, only one VOQ in $IM(0)$, which is VOQ(0, 0, 0), can send a cell with $L_i(0, 0)$ through $CM(0)$. The related pointers with the grant, $P_V(0, 0)$, $P_L(0, 0)$, and $P_c(0, 0)$, are updated from 0 to 1. At $T = 1$, three VOQs, which are VOQ(0, 0, 0), VOQ(0, 1, 0), and VOQ(1, 0, 0), can send cells. The related pointers with the grants are updated. Four VOQs can send cells at $T = 2$. In this situation, 100% switch throughput is achieved. There is no contention at the CMs from $T = 2$, because the pointers are desynchronized.

10.5 THE PATH SWITCH

If we consider each input module and each output module as a node, a particular connection pattern in the middle stage of the Clos network can be

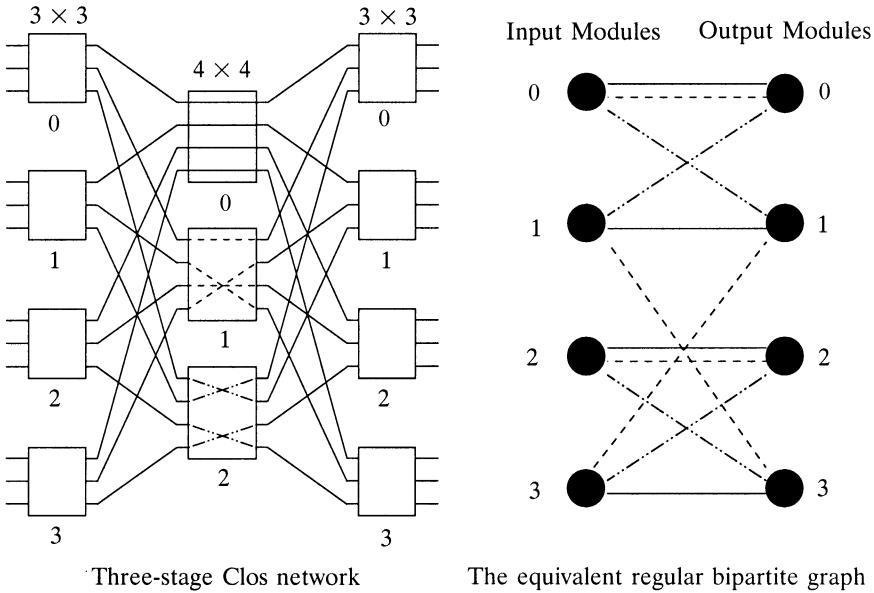


Fig. 10.11 The correspondence between the middle-stage route scheduling in a Clos network and the edge coloring of the regular bipartite multigraph. (©1997 IEEE.)

represented by a regular bipartite multigraph with node degree m as illustrated in Figure 11.11, where each central module corresponds to a group of n edges, each connecting one distinct pair of input/output nodes (modules).

Suppose the routing algorithm of the Clos network is based on dynamic cell switching, and the amount of traffic from input module I_i to output module O_j is λ_{ij} cells per time slot. The connection pattern will change in every time slot according to arrival packets, and the routing will be calculated on slot-by-slot basis. Let $e_{ij}(t)$ be the number of edges from I_i to O_j of the corresponding bipartite multigraph in time slot t . Then the capacity C_{ij} of the virtual path between I_i and O_j must satisfy

$$C_{ij} = \lim_{T \rightarrow \infty} \frac{\sum_{t=1}^T e_{ij}(t)}{T} > \lambda_{ij}. \quad (10.4)$$

On the other hand, the routing of a circuit-switched Clos network is fixed, and the connection pattern will be the same in every time slot. The capacity satisfies

$$C_{ij} = e_{ij}(t) = e_{ij} > \lambda_{ij}, \quad (10.5)$$

which implies that the peak bandwidth C_{ij} is provided for each virtual circuit at call setup time, and it does not take the statistical multiplexing into

consideration at all. We conceived the idea of quasistatic routing, called path switching, using a finite number of different connection patterns in the middle stage repeatedly, as a compromise of the above two extreme schemes. For any given λ_{ij} , if $\sum_i \lambda_{ij} < n \leq m$, and $\sum_j \lambda_{ij} < n \leq m$, we can always find a finite number f of regular bipartite multigraphs such that

$$\frac{\sum_{t=1}^f e_{ij}(t)}{f} > \lambda_{ij}, \quad (10.6)$$

where $e_{ij}(t)$ is the number of edges from node i to node j in the t th bipartite multigraph. The capacity requirement (10.4) can be satisfied if the system provides connections repeatedly according to the coloring of these f bipartite multigraphs, and that finite amount of routing information can be stored in the local memory of each input module to avoid the slot-by-slot computation of route assignments. The path switching becomes circuit switching if $f = 1$, and it is equivalent to cell switching if $f \rightarrow \infty$.

The scheduling of path switching consists of two steps, the capacity assignment and the route assignment. The capacity assignment is to find the capacity $C_{ij} > \lambda_{ij}$ for each virtual path between input module I_i and output module O_j ; it can be carried out by optimizing some objective function subject to $\sum_i C_{ij} = \sum_j C_{ij} = m$. The choice of the objective function depends on the stochastic characteristic of the traffic on virtual paths and the quality of services requirements of connections.

The next step is to convert the capacity matrix $[C_{ij}]$ into edge coloring of a finite number f of regular bipartite multigraphs, each of which represents a particular connection pattern of central modules in the Clos network. An edge coloring of a bipartite multigraph is an assignment of m distinct colors to m edges of each node such that no two adjacent edges have the same color. It is well known that a regular bipartite multigraph with degree m is m -colorable [9, 15]. Each color corresponds to a central module, and the color assigned to an edge from input module i to output module j represents a connection between them through the corresponding central module.

Suppose that we choose a sufficiently large integer f such that fC_{ij} are integers for all i, j , and form a regular bipartite multigraph, called the *capacity graph*, in which the number of edges between node i and node j is fC_{ij} . Since the capacity graph is regular with degree fm , it can be edge-colored by fm different colors [15]. Furthermore, it is easy to show that any edge coloring of the capacity graph with degree fm is the superposition of the edge colorings of f regular bipartite multigraphs of degree m . Consider a particular color assignment $a \in \{0, 1, \dots, fm - 1\}$ of an edge between input node I_i and output node O_j of the capacity graph; let

$$a = rf + t, \quad (10.7)$$

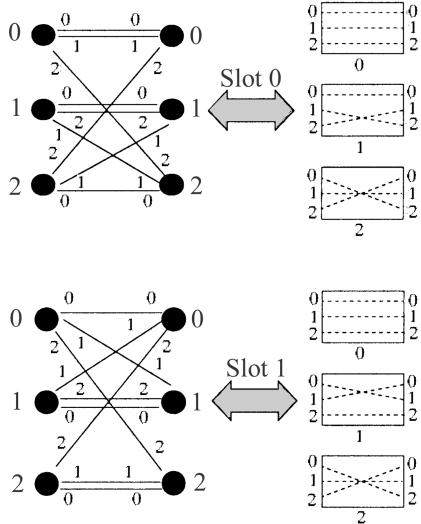
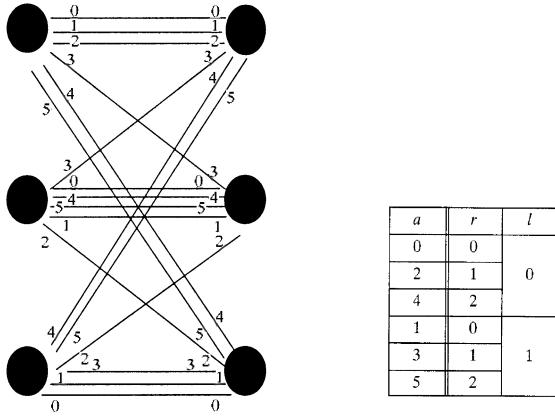


Fig. 10.12 Illustration of time-space interleaving principle. (©1997 IEEE.)

where $r \in \{0, 1, \dots, m - 1\}$ and $t \in \{0, 1, \dots, f - 1\}$ are the quotient and the remainder on dividing a by f , respectively. The mapping $g(a) = (t, r)$ from the set $\{0, 1, \dots, fm - 1\} \rightarrow \{0, 1, \dots, f - 1\} \times \{0, 1, \dots, m - 1\}$ is one-to-one and onto, that is,

$$a = a' \iff t = t' \text{ and } r = r'.$$

That is, the color assignment a , or equivalently the assignment pair (t, r) , of the edge between I_i and O_j indicates that the central module r has been

assigned to a route from I_i to O_j in the t th time slot of every cycle. Adopting the convention in TDMA system, each cycle will be called a *frame*, and the period f the *frame size*. As illustrated by the example shown in Figure 10.12, where $m = 3$ and $f = 2$, the decomposition of the edge coloring into assignment pairs guarantees that route assignments are either space-interleaved or time-interleaved. Thus, the relation (10.7) will be called the *time-space interleaving principle*.

10.5.1 Homogeneous Capacity and Route Assignment

For uniform traffic, where the distribution of traffic loading between input modules and output modules is homogeneous, the fm edges of each node can be evenly divided into k groups, where k is the total number of input or output modules. Each group contains $g = fm/k$ edges between any input–output pair, where the frame size f should be chosen to make the group size g an integer. The edges of this capacity graph can be easily colored by the *Latin square* given in Table 10.1, where each A_i , $0 \leq i \leq k - 1$, represents a set of distinct colors, e.g.,

$$A_0 = \{0, 1, \dots, g - 1\}, \quad A_1 = \{g, g + 1, \dots, 2g - 1\}, \dots$$

$$A_{k-1} = \{(k - 1)g, (k - 1)g + 1, \dots, kg - 1\}$$

Since each number in the set $\{0, 1, \dots, fm - 1\}$ appears only once in any row or column in the table, it is a legitimate edge coloring of the capacity graph. The assignment $a = (t, r)$ of an edge between the I_i – O_j pair indicates that the central module r will connect input module i to output module j in the t th slot of every frame. As an example, for $m = 3$ and $k = 2$, we can choose $f = 2$ and thus $g = 3$. Then, the groups of colors are $A_0 = \{0, 1, 2\}$ and $A_1 = \{3, 4, 5\}$, respectively. The procedure described above is illustrated in Table 10.2, and the correspondence between the route assignments and the connection patterns in the middle stage is shown in Figure 10.13.

In the above example, since the number of central modules (m) is greater than the number of input modules (k), it is possible that more than one central module is assigned to some input–output pair in one time slot. In the case that $m < k$, there are not enough central modules for all input–output

TABLE 10.1 Latin Square Assignment

	O_0	O_1	O_2	...	O_{k-1}
I_0	A_0	A_1	A_2	...	A_{k-1}
I_1	A_{k-1}	A_0	A_1	...	A_{k-2}
\vdots	\vdots	\vdots	\vdots		\vdots
I_{k-1}	A_1	A_2	A_3	...	A_0

TABLE 10.2 Route Assignment by Latin Square for Uniform Traffic

Color	O_0	O_1	Color $a = rf + t$	0	1	2	3	4	5
I_0	0, 1, 2	3, 4, 5	Central module $r = \lfloor a/f \rfloor$	0	0	1	1	2	2
I_1	3, 4, 5	0, 1, 2	Time slot $y = a \bmod f$	0	1	0	1	0	1

Latin square: edge coloring

↓

Transformation from color assignment into time-space pair

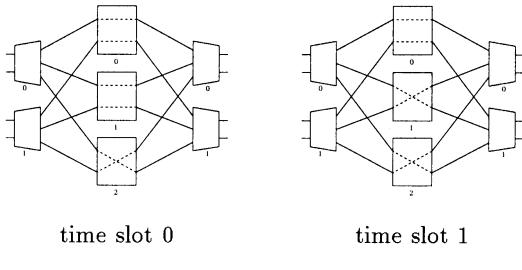
Central Module	O_0	O_1	Central Module	O_0	O_1
I_0	0, 1	2	I_0	0	1, 2
I_1	2	0, 1	I_1	1, 2	0

Time slot 0

Time slot 1

Central Module	Connected I/O pairs at Time Slot	
	0	1
0	$I_0/O_0, I_1/O_1$ (BAR)	$I_0/O_0, I_1/O_1$ (BAR)
1	$I_0/O_0, I_1/O_1$ (BAR)	$I_0/O_1, I_1/O_0$ (CROSS)
2	$I_0/O_1, I_1/O_0$ (CROSS)	$I_0/O_1, I_1/O_0$ (CROSS)

connection pairs in central modules

**Fig. 10.13** Route scheduling in the middle stage for uniform traffic. (©1997 IEEE.)

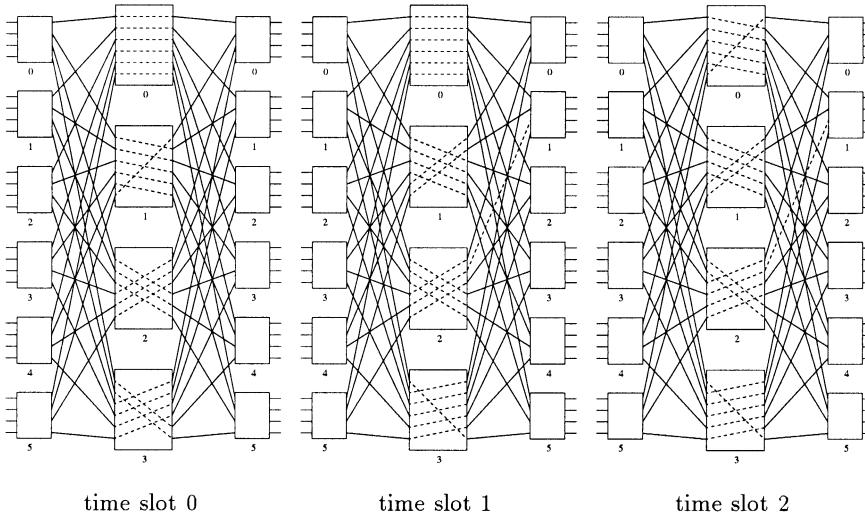


Fig. 10.14 Route scheduling in central modules for the second example of uniform traffic. (©1997 IEEE.)

pairs in one time slot assignment. Nevertheless, the total number of central modules assigned to every input–output pair within a frame should be the same for uniform input traffic to fulfill the capacity requirement, and that number is equal to $g = fm/k$. This point is illustrated in the following example. For $m = 4$ and $k = 6$, we choose $f = 3$ and $g = 2$. The same method will result in the connection patterns shown in Figure 10.14. It is easy to verify that the number of central modules (paths, edges) assigned for each input–output pair is equal to $g = 2$ per $f = 3$ slots.

10.5.2 Heterogeneous Capacity Assignment

The capacity assignment in a cross-path switch is virtual-path-based. It depends on the traffic load on each virtual path to allocate the capacity and determine the route assignment. The Latin square offers a legitimate capacity assignment with homogeneous traffic, but it may no longer be effective with heterogeneous traffic (nonuniformly distributed traffic load over the virtual paths). A more general assignment method is therefore introduced, and the procedure is illustrated in Figure 10.15. The assignment procedure has four steps, each of which will be explained along with an example in the following sub-subsections.

10.5.2.1 Virtual Path Capacity Allocation (VPCA) This step is to allocate capacity to each virtual path based on the traffic loads. It can be formulated as an optimization problem with some traffic modeling.

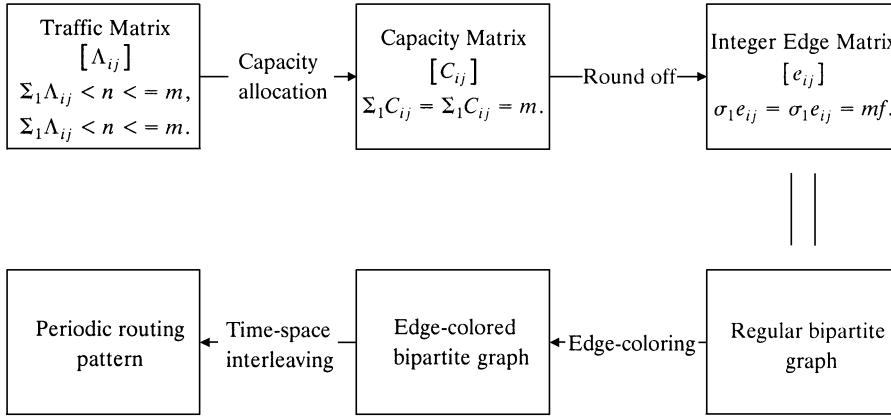


Fig. 10.15 Procedure of capacity and route assignment.

Consider the cross-path switch with parameters $n = 3$, $k = 3$, and $m = 4$. Suppose the traffic matrix is given by

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad (10.8)$$

and the capacity assignment matrix calculated by the minimization of input-stage delay with $M/D/1$ model is

$$\mathbf{C} = \begin{bmatrix} 1.34 & 1.28 & 1.38 \\ 2.66 & 1.34 & 0 \\ 0 & 1.38 & 2.62 \end{bmatrix}. \quad (10.9)$$

10.5.2.2 The Roundoff Procedure Some elements in the resulting capacity matrix may be nonintegers. When they are rounded to the integers required in the route assignment, roundoff error arises. The concept of frame size is used to reduce the roundoff error. Each element in the capacity matrix is multiplied by the frame size; then the capacity per slot is translated into capacity per frame. After that, we round the matrix into an integer matrix:

$$\begin{aligned} \mathbf{C} = \begin{bmatrix} 1.34 & 1.28 & 1.38 \\ 2.66 & 1.34 & 0 \\ 0 & 1.38 & 2.62 \end{bmatrix} &\xrightarrow{\times(f=3)} \begin{bmatrix} 4.02 & 3.84 & 4.14 \\ 7.98 & 3.82 & 0 \\ 0 & 4.14 & 7.86 \end{bmatrix} \\ &\xrightarrow{\text{rounding}} \begin{bmatrix} 4 & 4 & 4 \\ 8 & 4 & 0 \\ 0 & 4 & 8 \end{bmatrix} = \mathbf{E}. \end{aligned} \quad (10.10)$$

The roundoff error is inversely proportional to f . Thus, the error can be arbitrary small if the frame size is sufficiently large. However, since the amount of routing information stored in the memory is linearly proportional to f , the frame size is limited by the access speed and the memory space of input modules. In practice, the choice of frame size f is a compromise between the roundoff error and the memory requirement. In general,

$$\mathbf{E} = \begin{bmatrix} e_{0,0} & e_{0,1} & \cdots & e_{0,k-1} \\ e_{1,0} & e_{1,1} & \cdots & e_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ e_{k-1,0} & e_{k-1,1} & \cdots & e_{k-1,k-1} \end{bmatrix} \approx f\mathbf{C},$$

and

$$\sum_j e_{ij} = \sum_i e_{ij} = fm. \quad (10.11)$$

In the above matrix \mathbf{E} , each element e_{ij} represents the number of the edges between the input module i and output module j in the $k \times k$ capacity graph, in which each node has degree fm .

10.5.2.3 Edge Coloring As mentioned previously, this capacity graph can be colored by fm colors, and each color represents one distinct time-space slot based on the time-space interleaving principle (10.7). Coloring can be found by complete matching, which is repeated recursively to reduce the degree of every node one by one. One general method to search for a complete matching is the so-called Hungarian algorithm or alternating-path algorithm [9, 12]. It is a sequential algorithm with worst time complexity $O(k^2)$, or totally $O(fm \times k^2)$ because there are fm matchings. If each of fm

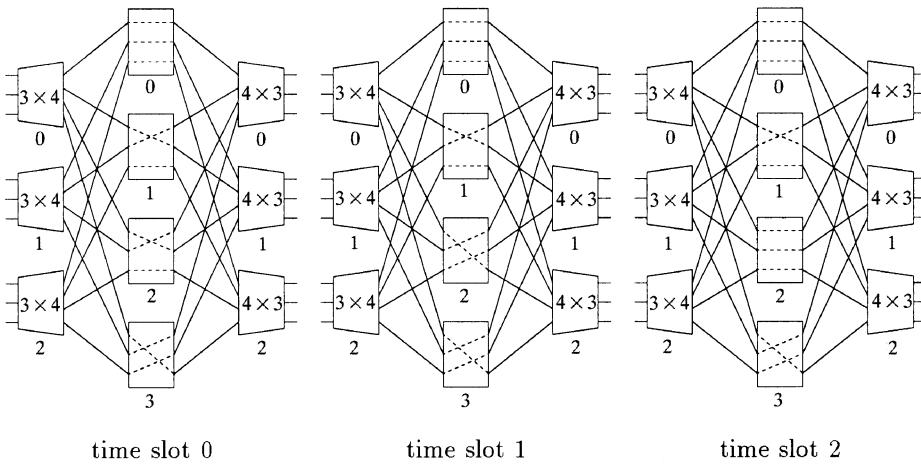


Fig. 10.16 Route scheduling example (heterogeneous traffic). (©1997 IEEE.)

and k is a power of two, an efficient parallel algorithm proposed in [7] for conflict-free route scheduling in a three-stage Clos network with time complexity of $O(\log^2(fmk))$ can be used. Through the time-space interleaving, the middle-stage routing pattern in Figure 10.16 is obtained.

REFERENCES

1. H. J. Chao and J.-S. Park, "Centralized contention resolution schemes for a large-capacity optical ATM switch," *Proc. IEEE ATM Workshop-97*, Fairfax, VA, May 1998.
2. H. J. Chao , "Saturn: a terabit packet switch using dual round-robin," *IEEE Commun. Mag.*, vol. 38, no. 12, pp.78–84, Dec. 2000.
3. K. Y. Eng and A. S. Acampora, "Fundamental conditions governing TDM switching assignments in terrestrial and satellite networks," *IEEE Trans. Commun.*, vol. 35, pp. 755–761, Jul. 1987.
4. K. Y. Eng, M. J. Karol, and Y.-S. Yeh, "A growable packet (ATM) switch architecture: design principles and applications," *IEEE Trans. Commun.*, vol. 40, no. 2, pp. 423–430, Feb. 1992.
5. M. J. Karol and C.-L. I, "Performance analysis of a growable architecture for broadband packet (ATM) switching," *Proc. IEEE GLOBECOM '89*, pp. 1173–1180, Nov. 1989.
6. C. H. Lam, "Virtual path traffic management of cross-path switch," Ph.D. dissertation, Chinese University of Hong Kong, Jul. 1997.
7. T. T. Lee and S. Y. Liew, "Parallel algorithm for Benes networks," *Proc. IEEE INFOCOM '96*, Mar. 1996.
8. T. T. Lee and C. H. Lam, "Path switching—A quasi-static routing scheme for large-scale ATM packet switches," *IEEE J. Select. Areas Commun.*, vol. 15, no. 5, pp. 914–924, Jun. 1997.
9. F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*, Morgan Kaufmann, 1992.
10. F. M. Chiussi, J. G. Kneuer, and V. P. Kumar, "Low-cost scalable switching solutions for broadband networking: the ATLANTA architecture and chipset," *IEEE Commun. Mag.*, pp. 44–53, Dec. 1997.
11. N. McKeown, "The iSLIP scheduling algorithm for input-queue Switches," *IEEE Trans. Networking*, vol. 7, no. 2, pp. 188–200, Apr. 1999.
12. R. J. McEliece, R. B. Ash, and C. Ash, *Introduction to Discrete Mathematics*, McGraw-Hill, 1989.
13. N. McKeown, "Scheduling algorithm for input-queued cell switches," Ph.D. Thesis, University of California at Berkeley, 1995.
14. E. Oki, Z. Jing, R. Rojas-Cessa, and H. J. Chao, "Concurrent round-robin dispatching scheme in a Clos-network switch," *IEEE ICC 2001*, pp. 107–112, Helsinki, Finland, Jun. 2001.
15. R. J. Wilson, *Introduction to Graph Theory*, Academic Press, New York, 1972.
16. Y.-S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: a simple, modular architecture for high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. 5, no. 8, pp. 1274–1283, Oct. 1987.

CHAPTER 11

OPTICAL PACKET SWITCHES

Introduction of optical fibers to communication networks has caused a tremendous increase in the speed of data transmitted. The virtually unlimited bandwidth of optical fibers comes from the carrier frequency of nearly 200 THz [1]. Optical networking technology, such as add-drop multiplexers [2, 3], reconfigurable photonic switches [4], and wavelength division-multiplexing (WDM), has progressed well and facilitated optical networking [5, 6]. Especially, recent advances in dense wavelength multiplexing division (DWDM) technology have provided tremendous bandwidth in optical fiber communications [7]. However, the capability of switching and routing packets at this high bandwidth (e.g., 1 Tbit/s) has lagged far behind the transmission capability. Building a large-capacity packet switching system using only electronic technology may lead to a system bottleneck in interconnecting many electronic devices or modules, mainly caused by the enormous number of interconnection wires and the electromagnetic interference they would generate. With the advancement of optical devices technology, several packet switch architectures based on WDM technology have been proposed for large-capacity packet switches. Although today's optical packet switching technology is still very primitive and cannot compete with electronic switching technology, optical packet switches have great potential to scale up their switching capacity as the technology of some key optical devices becomes mature.

A photonic packet switch requires optical devices such as lasers, filters, couplers, memories, multiplexers, and demultiplexers. At the present time, some of these devices are either too power-consuming or too slow in switching to compete with electronic devices. However, it is possible to design

high-capacity switches by the use of both electronic and optical technologies. In such switches, data transfer can be achieved through optical media, and complicated functions such as contention resolution and routing control can be performed electronically. These switches are called *hybrid* switches. Hybrid switches that only convert packet cell headers into electronics for processing and controlling and leave the entire cell to be handled in the optical domain are called *optically transparent*.

The ongoing research in photonic ATM switches is to develop faster and larger optical switches and new techniques that can be used to enhance the existing optical switch architectures. There are many issues to be considered when designing an optical packet switch, such as the characteristics of the optical devices employed, scalability of the switch, power budget of the system, synchronization between electrical and incoming optical signals, performance of the switch under various traffic patterns, and so on. In addition, some of the techniques developed for optical ATM switches might be applied to large-scale ATM switches where small electronic ATM modules are interconnected by an optical interconnection network.

The techniques of space-division multiplexing (SDM), time-division multiplexing (TDM), and WDM have been used in designing optical switches. SDM requires a large number of binary switching elements. From the switch size and cost point of view, it is not an ideal approach for photonic switching. TDM is a classical technique used in communications [8]. When it is applied to optical switching, complicated temporal compression and temporal expansion circuits are required. The throughput of such a switch is limited by the speed of the demultiplexer, which in practice is controlled by electronics for the time being.

WDM is made possible by the range of wavelengths on an optical fiber. It splits the optical bandwidth of a link into fixed, nonoverlapping spectral bands. Each band has a wavelength channel that can be used for a specific bit rate and transmission technique, independent of the choices for other channels.

In this chapter, we review several approaches to building a large-capacity packet switch and discuss their advantages and disadvantages. These switch architectures are classified into all-optical packet switches (described in Section 11.1) and optoelectronic packet switches (described in Section 11.2), depending on whether the contended packets are stored in the optical or in the electrical domain. Two optical packet switches will be described in detail in Sections 11.3 (using optical memory) and 11.4 (using optics for interconnection only) to better understand switching operations and implementation complexity. In all the architectures presented here, switch control is achieved electronically, since for the time being it is complicated to realize logical operations optically. The capacity of electronic control units and the tuning speed of optical devices are the main performance-limiting factors in these architectures.

11.1 ALL-OPTICAL PACKET SWITCHES

In optical packet switches, logical control and contention resolution are handled by an electronic controller, and packets are carried and stored in optical memories. There are two kinds of optical memory used in all-optical packet switches: one is the traveling type based on fiber delay lines, and the other is the fiber-loop type where packets carried at different wavelengths coexist in the fiber loop.

11.1.1 The Staggering Switch

The staggering switch [9] is one of the optically transparent switches. The major components of the switch are splitter detectors, rearrangeable non-blocking switches, and a control unit. The switch architecture is based on two stages: the scheduling stage and the switching stage, as shown in Figure 11.1. These two stages can be considered as rearrangeably nonblocking networks. The scheduling stage and the switching stage are of size $N \times M$ and $M \times N$, respectively, where M is less than N . These two stages are connected by a set of optical delay lines having unequal delay. The idea behind this architecture is to arrange incoming cells in the scheduling stage in such a way that there will be no output port collision in the switching stage. This is achieved by holding the cells that cause output port collision on the delay lines. The delay on the delay line d_i is equal to i cell slots. The arrangement of incoming cells

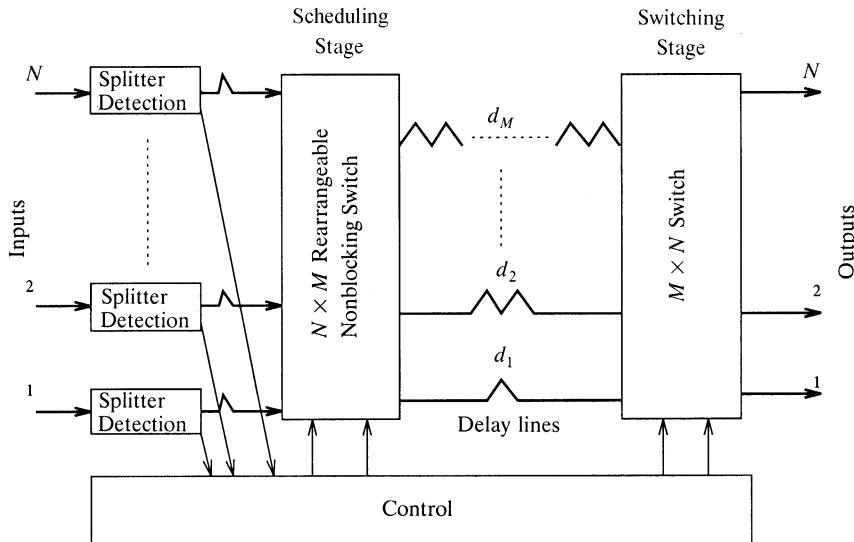


Fig. 11.1 Block diagram of the staggering switch. (© 1993 IEEE.)

is accomplished electronically by the control unit according to the output port requests of incoming cells.

When a cell arrives at the switch, its header information is converted into an electrical signal and sent to the control unit by the corresponding splitter/detector. After evaluating the current destination requests considering the previous requests, the control unit sends the information related to the current schedule to the scheduling stage. The cell is routed through the scheduling stage with respect to the information sent by the control unit. Due to the statistical properties of the incoming cells, it is possible to lose some cells in the scheduling stage. After waiting for a certain period of time on the assigned delay line, the cell reaches the switching stage. No contention occurs in the switching stage, on account of the precautions taken by the control unit, and the cell reaches the requested output port. In this architecture, cells arriving at the same input port may arrive at output ports in the reverse order, since they are assigned to different delay lines. Ordered delivery of cells at the output ports can be achieved by some additional operations in the control unit.

The main bottleneck in this switch architecture is the control unit. The proposed collision resolution algorithm is too complicated to handle large switch size or high input line rate. Some input buffers may be necessary in order to keep newly arriving cells while the control unit makes its arrangements.

11.1.2 ATMOS

Chiaroni et al. proposed a 16×16 photonic ATM switching architecture [10] for bit rates up to 10 Gbit/s. Basically, this switch consists of three main blocks: the wavelength encoding block, the buffering and time switching block, and the wavelength selection block, as shown in Figure 11.2. In the wavelength encoding block, there are N wavelength converters, one per input. Each input is assigned a fixed wavelength by its wavelength converter. When a cell arrives, a small fraction of the optical signal power is tapped by a coupler and converted to an electronic signal. A controller processes these converted data and extracts the routing information for the cell. The arriving cells with different wavelengths are wavelength-division multiplexed in the buffering and switching block by a multiplexer. The buffering-and-time-switching block contains K fiber delay lines to store the payloads of the incoming cells. There is also a space switch, which is made of semiconductor optical amplifier (SOA) gates. These gates are used to select the cells from the fiber delay lines and route them to the requested output ports. The wavelength selection block consists of multiplexer/demultiplexer and SOA gates in order to select a specific wavelength destined to an output port in a cell time slot. This switch can perform the multicast function by using a broadcast-and-select approach.

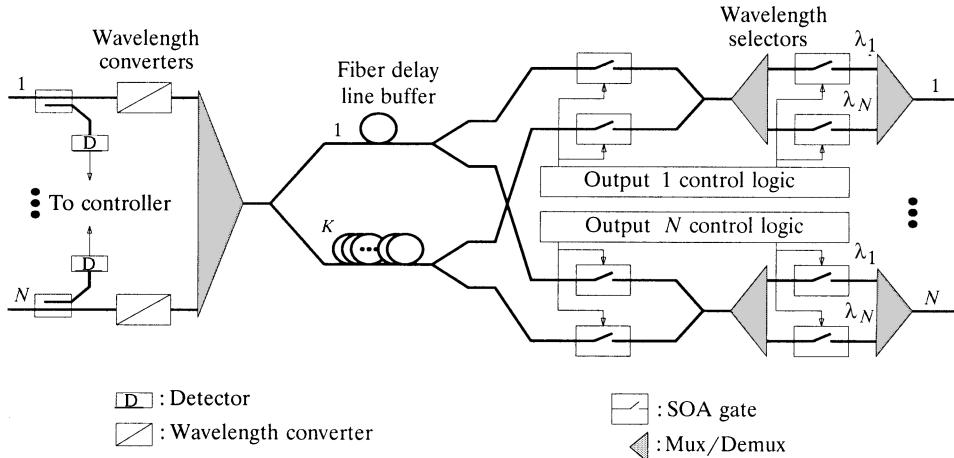


Fig. 11.2 Architecture of the ATMOS switch.

The cell contention problem is solved by the fiber delay lines. However, this approach cannot provide for sharing, so that a great number of delay lines are necessary to meet the cell loss requirement. The architecture is bulky in structure, and the switch size is limited by the available number of wavelengths.

11.1.3 Duan's Switch

Duan et al. introduced a 16×16 photonic ATM switching architecture [11], as shown in Figure 11.3, where each output port is assigned a fixed wavelength. This switch consists of three main blocks: wavelength encoding block, spatial switch block, and wavelength selection block. In the wavelength encoding block, there are N wavelength converters, one per input, each being tuned to the destined output port. When a cell arrives, a small fraction of the optical signal power is tapped by a coupler and sent to the electronic control unit, which processes the routing information of the cell. In a specific cell time slot, cells destined to different outputs are tuned to different wavelengths. These cells with different wavelengths are routed through the shortest path, which is selected by the SOA gates in the spatial switch. The spatial switch block contains K fiber delay lines to store the payloads of the cells for contention resolution. Each fiber delay line can store up to N different wavelengths. In the wavelength selection block, in each cell slot time, multiple wavelengths are broadcast to all output ports by a star coupler. There is a fixed-wavelength filter at each output port. These filters select the cells associated with their wavelengths and send them to the corresponding output ports.

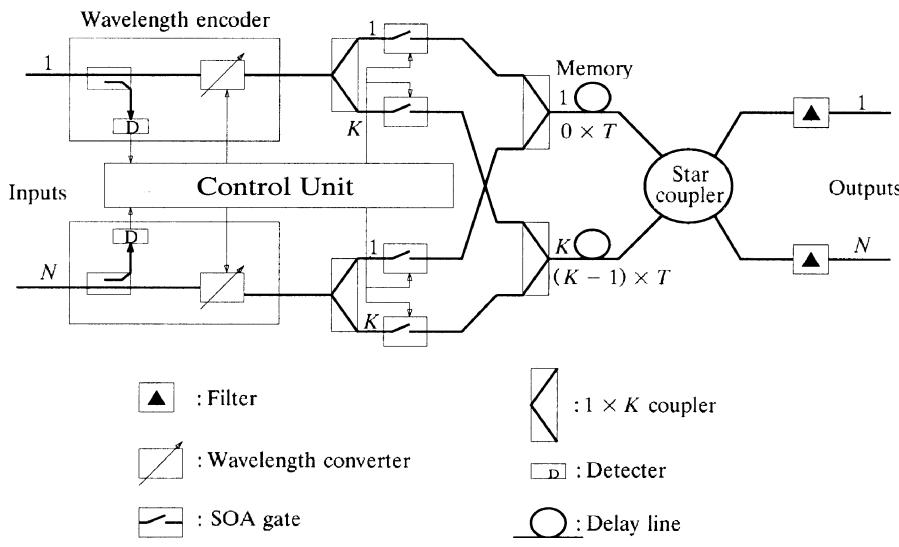


Fig. 11.3 Architecture of the ATM wavelength routing system.

This switch cannot perform multicast functions, because of the fixed-wavelength filters at the output ports. Furthermore, if there is more than one cell destined to the same output port, an arbitration mechanism is necessary in order to assign the incoming cells with the same wavelength to different fiber delay lines. Such a requirement increases the control complexity. In order to meet the cell loss requirement, more fiber delay lines are necessary. Moreover, the electronic controller always has to monitor the status of fiber delay lines to preserve the cell sequence.

11.2 OPTOELECTRONIC PACKET SWITCHES

For the optoelectronic packet switches, optical switching networks are used for interconnection and transmission between electronic input and output modules. Logical control, contention resolution, and packet storage are handled electronically.

11.2.1 HYPASS

HYPASS [14] in Figure 11.4 is an optoelectronic hybrid cell switch in which electronic components are used for memory and logic functions, and optical components are used for routing and transporting data. In this figure, bold continuous lines represent optical paths, bold dashed lines represent serial data paths, dotted lines are tuning current paths, and thin continuous lines are control signal paths. The switch is composed of two networks: the transport network and the control network. The architecture is based on the *broadcast-and-select* approach in both of the networks. There is a unique

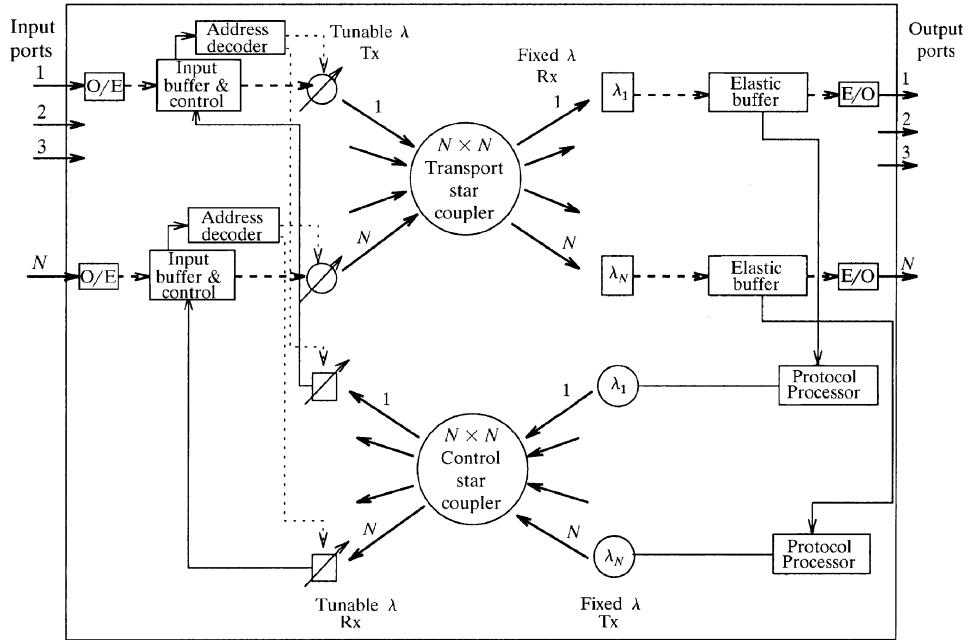


Fig. 11.4 Global diagram of the HYPASS implementation. (© 1988 IEEE.)

optical wavelength associated with each of the output ports. As shown in Figure 11.4, the transport network has tunable-wavelength laser transmitters at the input side, fixed-wavelength receivers at the output side, and an $N \times N$ star coupler, which transfers the incoming data from inputs to outputs. In order to transfer control information from output ports to the input ports, a similar network is used.

When a cell arrives at an input port of the switch, first it is converted from optical to electronic and its destination address is obtained. Then the cell is temporarily stored in the corresponding input buffer. The tunable-wavelength laser transmitter of the corresponding input port is tuned to the wavelength of the requested output port. When a *request-to-send* signal (or *poll*) is received from the corresponding output port via the control network, the cell is transmitted through the transport network. The acknowledgment representing successful delivery of the cell is also transmitted through the control network. If there are multiple cells for the same output port, contention occurs. Power threshold detection or multiple bit detection on the cell preamble could be used to detect collision. The cells which do not get acknowledgments in a slot time are kept to retry later. In order to resolve contention and provide successful transmission of cells, the tree-polling algorithm, which is explained in [14], is employed in the selection of inputs in the following cell slots. The cells that reach the output ports successfully are stored in the elastic buffers and transmitted over the optical fiber trunks after the necessary electrical-to-optical conversion.

The HYPASS architecture has advantages due to its parallel structure. However, since a slot time is based on the length of the polling step, transmission of a cell, and receipt of the acknowledgment, the time overhead for the electronic control and optical tuning operations are the factors limiting its capacity. The switch does not have multicasting capability, due to the use of fixed wavelength receivers at the output ports.

11.2.2 STAR-TRACK

STAR-TRACK [15] is another hybrid switch architecture. It is based on a two-phase contention resolution algorithm. It also supports multicasting. As shown in Figure 11.5, the switch is composed of two internal networks: an optical star transport network, and an electronic control track surrounding the star network. The optical transport network has fixed-wavelength optical transmitters at the input port side, and wavelength-tunable optical receivers at the output port side. There is a unique wavelength associated with each input port. Input and output ports are connected through an optical star coupler. Output port conflicts are resolved by the ring reservation technique [16]. The electronic control network that implements the ring reservation technique is the major track linking input ports, output ports, and a token generator sequentially.

Cells arriving at the input ports are stored in the input buffers after optical-to-electronic conversion. There are two control phases in a cell transmission cycle. In the first phase, input ports write their output port requests into the tokens circulating in the control network. In the second

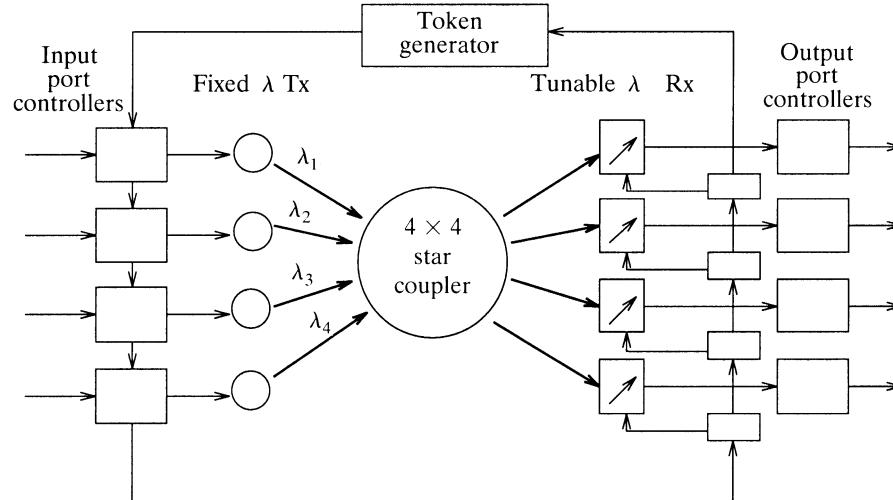


Fig. 11.5 STAR-TRACK architecture (basic single track).

phase, the output ports read the tokens and tune their receivers to the appropriate input port wavelengths. Then, the cell transmission starts over the star transport network. The transmission and control cycles are overlapped in time in order to increase throughput. Since each input has a unique wavelength and there is input–output port pair scheduling prior to transmission, cells are transmitted simultaneously without causing contention.

This architecture allows multicasting. However, the throughput of the switch may degrade as the number of multicasting connections increases, due to output port collisions in the first phase. It is shown that this problem can be alleviated by call splitting (i.e., allowing a multicast call to be completed in multiple cell slots). This architecture can support different priority levels for the cell by adding minor tracks into the control network. However, in that case, the token should recirculate among the input ports more than once, depending on the number of priority levels. This will increase the length of the write phase and result in longer cell processing time.

The main drawback of the switch is the sequential processing of the token by input and output ports. As a result, the time taken for the token to travel through the entire ring increases as the size of the switch increases. In the case of multiple priority levels, the recirculation period for the token becomes even longer. Here, HOL blocking is another factor that degrades throughput.

11.2.3 Cisneros and Brackett's Architecture

Cisneros and Brackett [17] proposed a large ATM switch architecture that is based on memory switch modules and optical star couplers. The architecture consists of input modules, output modules, optical star couplers, and a *contention resolution device* (CRD). Input and output modules are based on electronic shared memories. The architecture requires optical-to-electronic and electronic-to-optical conversions in some stages. Each output module has an associated unique wavelength. As shown in Figure 11.6, the input ports and output ports are put in groups of size n , and each group is connected to $n \times m$ or $m \times n$ memory switches, respectively. The interconnection between the input and output modules is achieved by k optical star couplers. There are k tunable laser transmitters and k fixed-wavelength receivers connected to each optical star coupler. In the Figure 11.6, for simplicity, the optical transmitters and receivers are not shown. The cells transmitted through the switch are buffered at the input and output modules. In the proposed architecture, input and output lines transmit cells at the rate of 155.52 Mbit/s. The lines that interconnect the input modules to the optical stars, and the optical stars to the output modules, run at 2.5 Gbit/s. The values of n , k , N , and m are 128, 128, 16,384, and 8, respectively.

The internal routing header of a cell is composed of two fields. One specifies the output module, and the other the port number in that output module. Each input module handles a single queue, in which the incoming

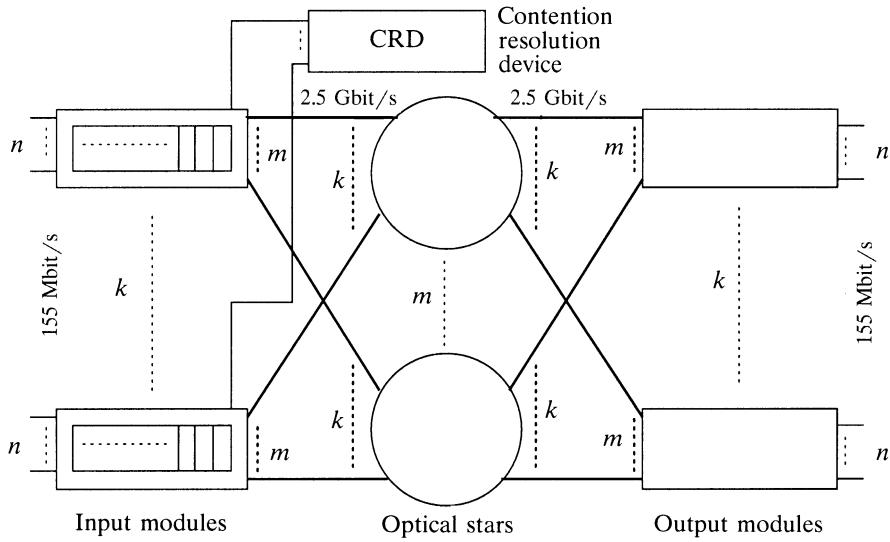


Fig. 11.6 The switch architecture proposed by Cisneros and Bracket. (© 1991 IEEE.)

cells are kept in sequence. The input modules, the optical stars, and the output modules are connected as in a three-stage Clos network. However, the working principle is not the same as in the Clos network. Here, each input module sends the output module request of its HOL cell to the CRD. The CRD examines the requests, chooses one cell for each output module, and responds. The cells that won the contention are routed through the first $k \times k$ optical star, and their HOL pointers are advanced. This process is repeated cyclically for each optical star. Cells at the output modules are kept in a sequence depending on which optical star they arrive in. In this architecture, all optical stars are kept busy if the CRD is m times faster than the time for cell transfer by the optical stars. The maximum number of optical couplers is determined from the time required to transfer a cell through an optical star and the time required by the CRD to resolve contention.

In the architecture, the time required for optical-to-electronic conversion, electronic-to-optical conversion, and tuning optical laser transmitters is not considered. All the calculations are mainly based on the time required to transfer a cell through an optical star. The output port contention resolution scheme is very complex, and the electronic controller can become a bottleneck. The switch does not have multicast capability, due to the fixed-wavelength receivers. Moreover, the maximum throughput of the switch is limited to 58% because of the HOL blocking [18].

11.2.4 BNR Switch

Munter et al. introduced a high-capacity packet switch based on advanced electronic and optical technologies [20]. The main components of the switch are input buffer modules, output buffer modules, a high-speed switching core, and a central control unit, as shown in Figure 11.7. The core switch contains a 16×16 cross-connect network using optical links running at 10 Gbit/s.

The central control unit receives requests from input buffer modules and returns grant messages. Each request message indicates the number of queued packets in the input buffer module, which is later used to determine the size of burst allowed to transmit to the switch fabric. A connection can only be made when both input and output ports are free. A control bus is used by the free input ports to broadcast their requests, and by the free output ports to return grant messages.

An arbitration frame consists of 16 packet time slots for a 16×16 core switch. In each slot, the corresponding output port polls all 16 inputs. For example, in time slot 1, output port 1 (if it is idle) will choose the input that has the longest queue destined for output port 1. If the input is busy, another input port that has the second longest queue will be examined. This operation repeats until a free input port is found. If a match is found (free input, free output, and outstanding request), a connection is made for the duration corresponding to the number of packets queued for this connection. So the switch is a burst switch, not a packet switch. In time slot 2, output port 2

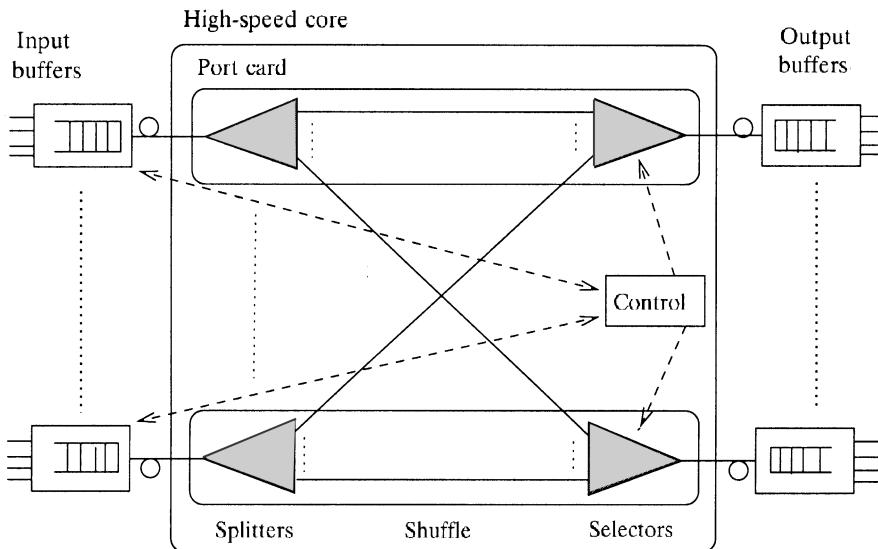


Fig. 11.7 Diagram of BNR switch.

repeats the above operation. The switch capacity is limited by the speed of the central control unit. Packet streams can have a long waiting time in the input buffer modules under a high traffic load.

11.2.5 Wave-Mux Switch

Nakahira et al. introduced a photonic ATM switch based on input–output buffering [21]. Basically, this switch consists of three kinds of modules: the input group module (IGM), switching module (SWM), and output group module (OGM), as shown in Figure 11.8. They are connected by means of fiber optical lines. The inputs are divided into p groups of size n_1 , and each group is connected to an IGM. The cells arriving through optical lines are first converted to electronic signals by optical-to-electronic converters, and their header information is electrically processed at the header converter in IGMs. Both the header and the payload of the arriving cell are processed and stored in an electronic random access memory (RAM). An optical sorter in each IGM is used to sort the cells with respect to their OGM requests and delivers them to the SWM in one cell slot time.

There are p optical switches in the SWM. Each optical switch transmits optical wavelength multiplexed cells from IGM to OGM. In each cell time slot, these p optical switches deliver at most p trunks of wavelength-multiplexed cells from IGMs, which are destined to the different OGMs. In each

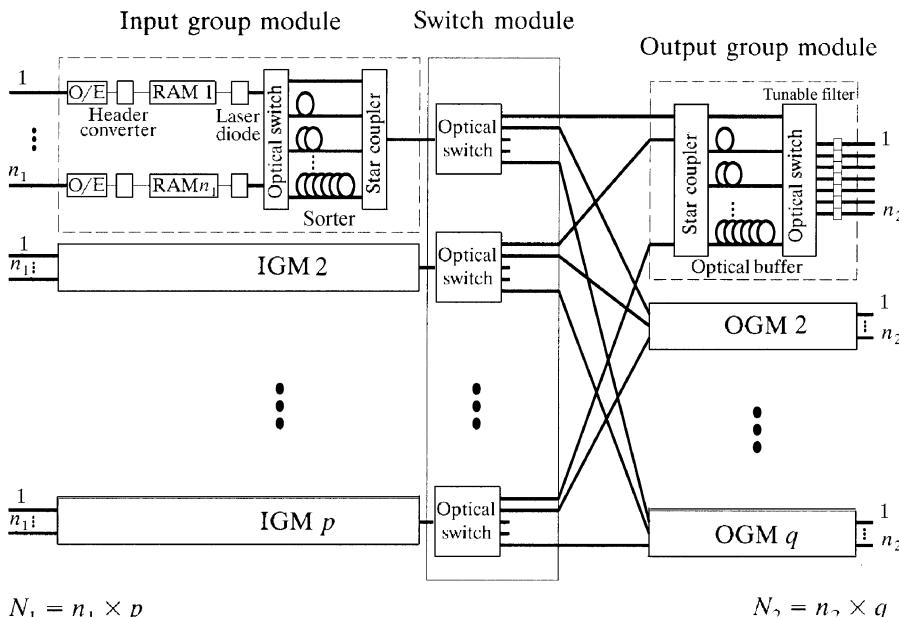


Fig. 11.8 Architecture of the wave-mux switch. (©1995 IEEE.)

OGM, it is possible to have the cells with different wavelengths but with the same output port request. This is called output port contention and is solved by the use of an optical buffer. This optical buffer in the OGM is based on the fiber delay line principle. If no competing cells to the same output port are present in the optical buffer, the incoming wavelengths will be sent through the shortest optical fiber line. They are distributed to the tunable filters by an optical switch. Each tunable filter is then tuned to the wavelength of the requested output port.

The proposed optical switch architecture needs complex arbitration to solve the contention problem, not only for the cells in the same IGM but for the cells in the different IGMs as well. This will increase the complexity of control and thus limit the switch size. In this switch, in order to avoid HOL blocking, cells destined to the same OGM can be read out of the RAM with a speedup factor of two. Many optical-to-electronic and electronic-to-optical converters are required in the switching path, increasing the implementation cost.

11.3 THE 3M SWITCH

The WDM ATM multicast (3M) switch is an optically transparent ATM switch [22, 23]. By taking advantage of both optical and electronic technologies, it is proposed to route ATM cells through an optical switching plane, while extracting and processing their headers in an electronic plane that controls the optical devices and routes the cells to the proper output port(s).

11.3.1 Basic Architecture

Figure 11.9 shows the architecture of an enhanced $N \times N$ 3M switch, where incoming cells running at 2.5 Gbit/s are optically split into two paths. Cells on the top path remain in the optical domain and are routed through the optical switch plane. Cells on the bottom path are converted to the electronic domain, where their headers are extracted for processing (e.g., finding the output ports for which the cells are destined and finding new VPI/VCI values to replace the old ones). An electronic central controller, as shown in Figure 11.9, performs cell delineation, VCI overwrite, cell synchronization, and routing. The first three functions are implemented in the photonic ATM front-end processor, while the last one is handled by a route controller that routes cells to proper output ports.

As shown in Figure 11.10, the cell format adopted in the system has 64 bytes with 5 bytes of header, 48 bytes of payload, and two guard time fields (with all ones), which are 6 and 5 bytes long, respectively. The guard times are used to accommodate the slow switching of optical devices such as optical tunable filters. The lengths of the guard times between the cells, and between the cell header and the payload, were chosen arbitrarily. Cells are transmitted back-to-back and not carried in SONET frames. Not using SONET

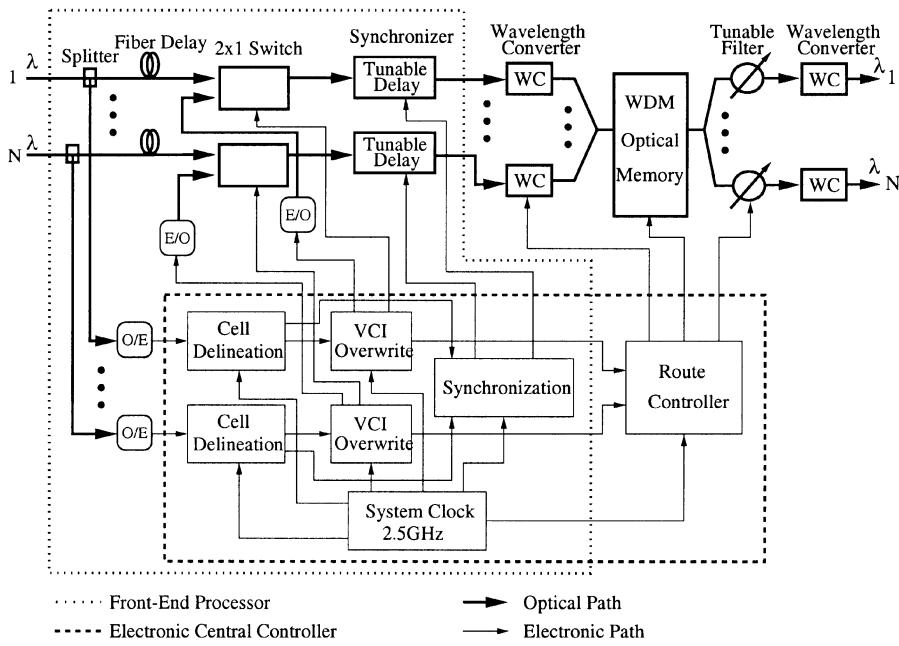


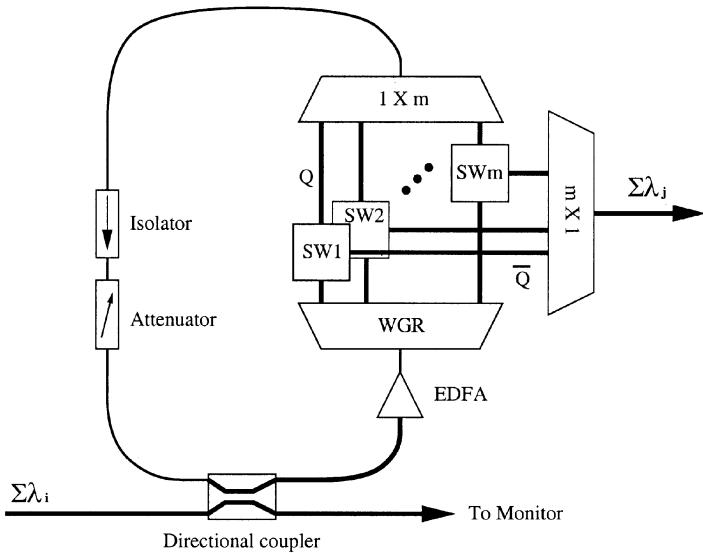
Fig. 11.9 Architecture of the WDM ATM multicast (3M) switch. (©2000 IEEE.)

Header	Guard time	Payload	Guard time
5	6	48	5 (in byte)

Fig. 11.10 Cell format adopted in the system.

frames eliminates the possibility of having variable gaps between or within cells caused by the need to carry SONET transport and path overhead ranging from 1 to 49 bytes.

The incoming optical cells are first delayed by fiber lines, processed for their headers, and synchronized in the front-end processor before they are sent to the switch fabric. In the switch fabric, cells are converted to different wavelengths by wavelength converters (WCs) that are controlled by the route controller, which keeps track of the available wavelengths in the WDM optical shared memory. It is a fiber-loop memory, as shown in Figure 11.11, and is used to store optical cells until they are ready to be transmitted to the next node. Using a 3-dB directional coupler, cells are coupled into the optical memory and coexist with the existing cells. Accessing cells in the optical memory is done by controlling the 1×2 space switches (SW_i)—for example, SOA gates. The wavelength-division multiplexed cell stream is amplified by an erbium-doped fiber amplifier (EDFA) to compensate power loss when looping in the memory. The cell stream is then demultiplexed by a waveguide



SW : Space Switch (e.g., 1 X 2 SOA gate)

WGR : Waveguide Grating Router

EDFA : Erbium Doped Fiber Amplifier

Fig. 11.11 An optical random access memory. (©2000 IEEE.)

grating router (WGR) into m different channels, each carrying one cell. The maximum number of cells (i.e., wavelengths) simultaneously stored in this memory has been demonstrated to be 23 circulations at 2.5 Gbit/s. Cells read from the WDM optical shared memory are broadcast to all N output ports by a $1 \times N$ splitter and selected by the destined output port (or ports, if multicast) through tunable filters that are tuned by the route controller on a per-cell basis. The final WC stage converts cells to their predetermined wavelengths. Other optical loop memory can be found in [25, 26, 27, 28, 29].

Figure 11.12 shows how the shared memory is controlled by a route controller. Signals R_1 to R_4 carry the output port addresses for which the cells are destined. An idle-wavelength FIFO keeps track of available wavelengths in the memory. When up to four incoming cells arrive, free wavelengths are provided by the idle-wavelength FIFO, and are used to convert incoming cells' wavelengths so they can be written to the loop memory at the same time. These wavelengths are stored in the FIFOs (FIFO 1 to FIFO 4) according to the R_1 to R_4 values. Since the 3M switch supports multicasting, the same wavelength can be written into multiple FIFOs. All the FIFOs (including the idle wavelength one) have the same depth, storing up to m wavelengths. While the wavelength values are written sequentially (up to four writes in each cell slot) to the FIFOs, the wavelengths of the HOL cells of the FIFOs are read simultaneously, so that up to four cells can be read out

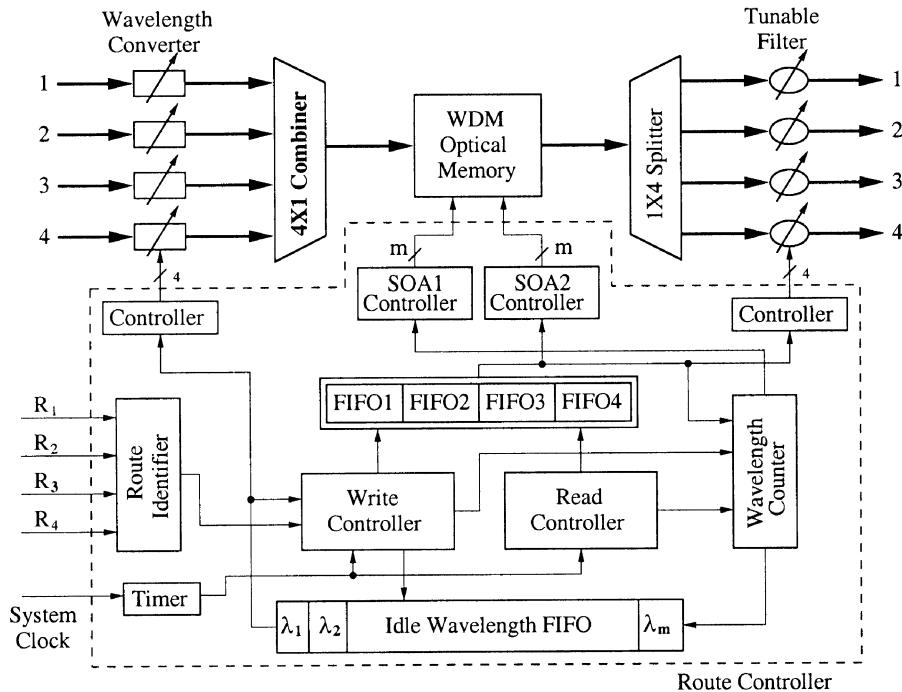


Fig. 11.12 An optical shared memory controlled by a route controller. (©2000 IEEE.)

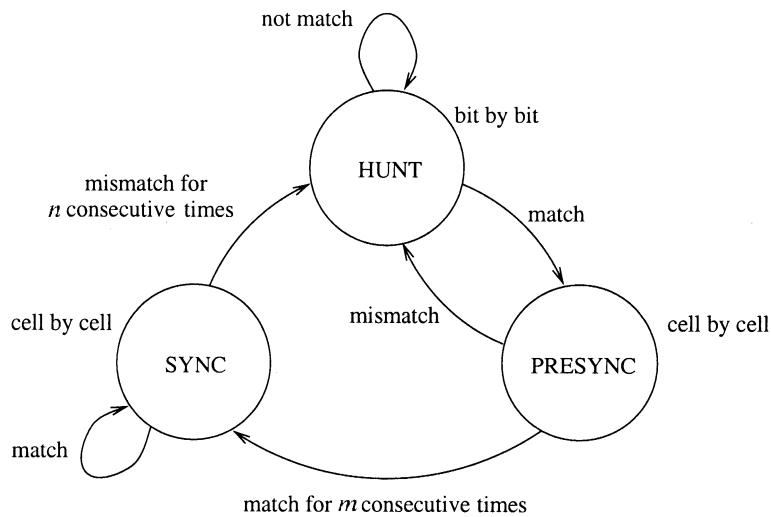
simultaneously. They are, in turn, used to control the tunable filters to direct the cells to the proper output ports. The write controller and read controller generate proper signals to coordinate all functional blocks.

The header of an ATM cell carries all necessary information for routing. The photonic ATM front-end processor is designed to extract the cell header and perform the functions, including cell delineation, VCI overwrite, and cell synchronization. The three basic units to perform these functions are described in the following.

11.3.2 Cell Delineation Unit

As shown in Figure 11.9, an optical cell stream is tapped from each input line, converted to electronic format, and sent to the cell delineation unit. Cell delineation is a process used to identify cell boundaries so that the incoming cell stream can be further processed at the cell level by the following units, such as VCI overwrite.

The standard HEC checking mechanism to find cell boundaries is adopted. It takes advantage of the inherent cyclic redundancy check (CRC) coding correlation between the cell header to be protected (the first 4 bytes) and HEC byte (the fifth byte of the cell header). Figure 11.13 shows the state



HUNT state : Search for cell boundary

PRESYNC state : Cell boundary was found, but not confirmed

SYNC state : Cell boundary was confirmed

Fig. 11.13 State diagram of cell delineation.

diagram of cell delineation. Initially, a cell boundary is arbitrarily assumed and checked by performing a polynomial division bit by bit in the HUNT state (Figure 11.13). If the remainder (the *syndrome*) for a complete calculation is zero, then this boundary is assumed to be correct. Otherwise, shift a bit from the data stream and repeat the operation until the syndrome is zero. Once a cell boundary is determined, it is then confirmed cell by cell for eight consecutive times in the PRESYNC state before the cell boundary is claimed to be found. It then goes to the SYNC state. Once in the SYNC state, the cell boundary is claimed to be lost when seven consecutive mismatches occur. As a result, the above procedure for cell delineation will start over again (from the HUNT state).

As shown in Figure 11.14, to reduce the high-speed circuit requirement, the serial bit stream at 2.5 Gbit/s is first converted to 16-bit parallel words (155 Mbit/s) through a serial-to-parallel converter. A 16-bit parallel format of the CRC circuit is used to perform polynomial division, and the syndrome is checked every three word clock cycles. An HPS (HUNT, PRESYNC, and SYNC) finite state machine performs the state transition between HUNT, PRESYNC, and SYNC states in Figure 11.13. If a syndrome equals zero, then the finite state machine goes to the PRESYNC state from the HUNT state and disables a set of control and shift circuits by a signal Y . Otherwise,

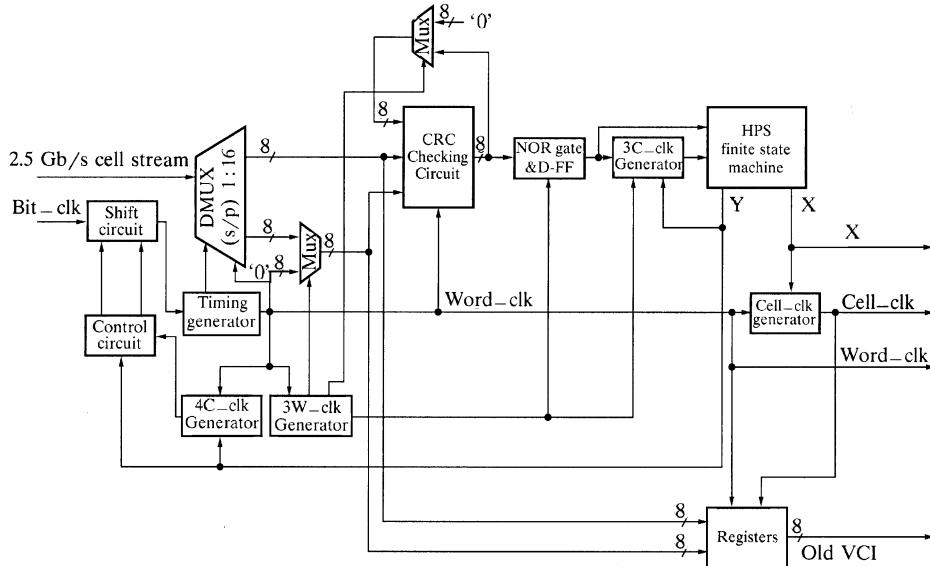


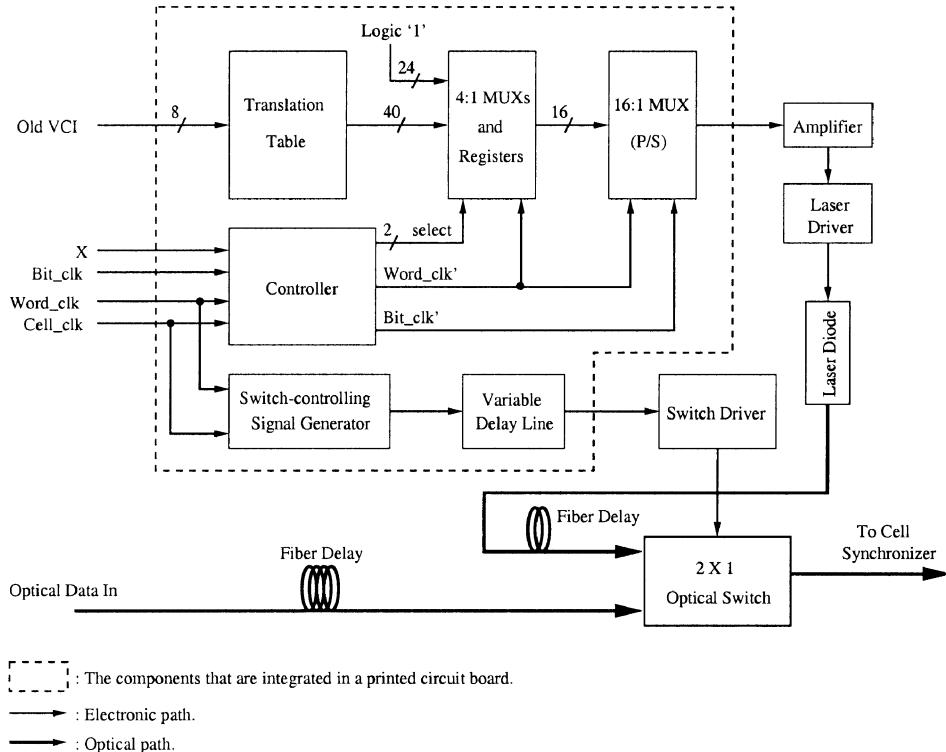
Fig. 11.14 Block diagram of the cell delineation unit. (©2000 IEEE.)

the finite state machine notifies the control and shift circuits to inhibit a bit or a byte every three cell clock cycles. Once the cell boundary is confirmed, the state machine goes to the SYNC state and sends a signal X to create a cell clock, which indicates the location of the cell boundary. The cell clock and signal X are passed to the VCI-overwrite unit together with the old VPI/VCI and the word clock.

Identifying cell boundaries for a back-to-back cell stream at 2.5 Gbit/s without using SONET frames is much more difficult than in the case where cells are carried over SONET frames. When an error HEC is detected in the HUNT state, a clock pulse at 2.5 GHz (or 200 ps) is masked and the HEC hunting process is restarted, rather than shifting one byte as in the case where cells are carried over SONET frames.

11.3.3 VCI-Overwrite Unit

Once cell boundaries are recognized and confirmed by the cell delineation unit, the state machine moves to the SYNC state and enables the VCI-overwrite unit with the cell clock and signal X , as shown in Figure 11.15. The main function of this unit is to overwrite the VPI/VCI field of the incoming cell header in the optical domain. The VCI-overwrite unit performs table lookups in the electronic domain, converts the new VPI/VCI to an optical signal, and replaces the old VPI/VCI by using a 2×1 optical switch. The routing table (i.e., VPI/VCI translation table) is preprogrammed manually.



Dashed box: The components that are integrated in a printed circuit board.

→ : Electronic path.

→ : Optical path.

Fig. 11.15 Block diagram of the VCI-overwrite unit.

The challenge is to handle the high-speed overwriting at the bit rate of 2.5 Gbit/s with each bit only 400 ps long. It is solved by using electronic variable delay lines (programmable delay) to compensate for the time difference between the old header and the new header.

As shown in Figure 11.15, the new header obtained from the table lookup is converted to a serial format by a parallel-to-serial converter. It is then used to control a laser driver to drive a DFB laser diode that generates the cell header in the optical domain. The new header replaces the old one using a 2×1 optical switch that is controlled by a 6-byte-wide pulse in every cell time slot. The successfully overwritten cells are sent to fiber delay lines in the cell synchronization unit.

11.3.4 Cell Synchronization Unit

The cell synchronization unit is used to align the phases of incoming ATM cells in the optical domain. The synchronization issue is also addressed in [30]. Burzio et al. [31] and Zucchelli et al. [32] have implemented a two-stage cell synchronizer (coarse and fine synchronizer) at the rate of 622 Mbit/s. The former uses a few slow thermo-optic 2×2 switches to control the cells

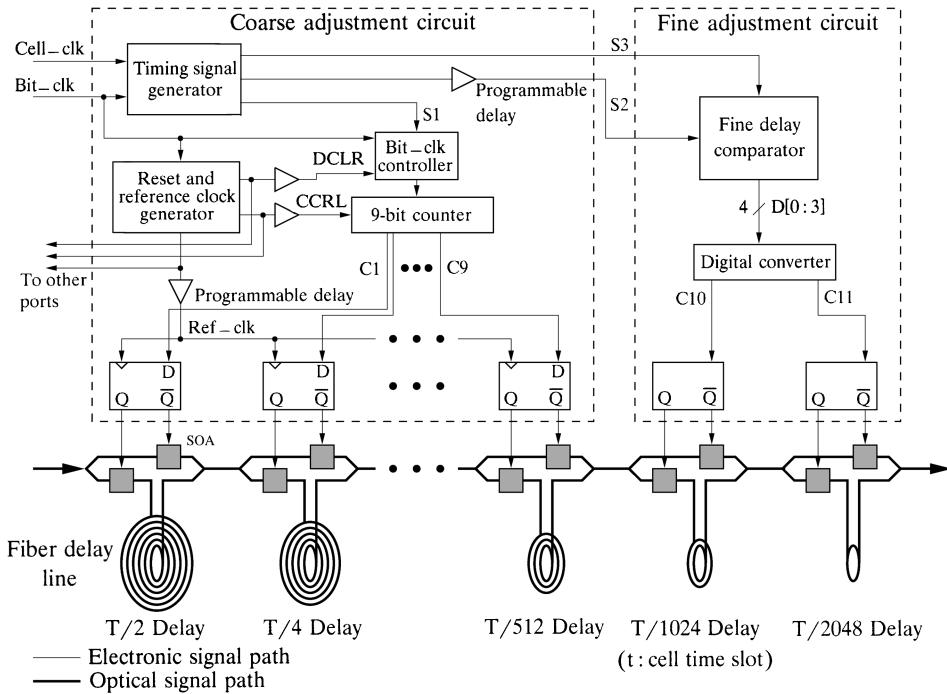


Fig. 11.16 Block diagram of the cell synchronization unit.

through different fiber delay lines. The latter uses a tunable wavelength converter to convert the wavelength of the cells so as to finely adjust the delay along a highly dispersive fiber.

In the 3M switch, synchronization is done at two levels. The cell contention resolution, VPI/VCI overwrite, and cell read (write) from (to) the loop memory are executed at the cell level, while the interaction between electronic and optical signals is at the bit level. For instance, cell header overwrite is done optically at the bit level. Furthermore, synchronization among the incoming optical cells is achieved at $\frac{1}{4}$ bit.

The cell synchronization unit in Figure 11.16 is used to optically align cells from different inputs to the extent of $\frac{1}{4}$ bit (100-ps, or 2-cm, optical delay line at 2.5 Gbit/s) before they are further processed in the switch fabric. Because of the stringent timing requirement, control is divided into two steps. A coarse adjustment circuit controls the first nine stages of the optical delay elements and adjusts the phases of incoming cells down to the bit level. A fine adjustment circuit controls the last two stages and further adjusts the phase down to $\frac{1}{4}$ bit.

Each stage of the optical delay element consists of a Y-junction SOA gate, a combiner, and a fiber delay line with a delay of $T/2^n$ (where T is one cell time slot and n is from 1 to 11). There are challenging issues in the optical delay

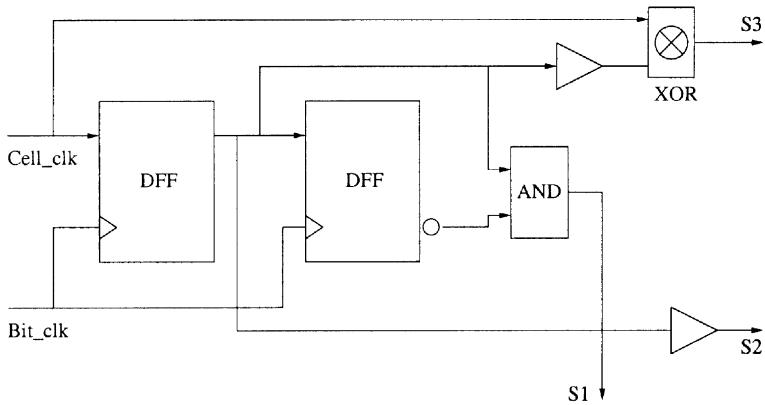


Fig. 11.17 Block diagram of the timing signal generator.

line fabrication, such as polarization, noise reduction, coherent crosstalk, and power stabilization. InP-based semiconductor Y-junction switches have been implemented for the delay units.

The cell clock generated in the cell delineation unit indicates the cell boundaries. By comparing the cell clock with a reference clock generated by the system, a 9-bit digitized timing difference can be obtained by using a 9-bit counter in the coarse adjustment circuit. Signals C1 to C9 control the nine switching stages (each consisting of two SOA gates) to determine if each cell should pass the fiber delay line or not. As shown in Figure 11.17, three important timing signals S1 to S3 are generated by the timing signal generator. S1, shown in Figure 11.18, is used to enable the 9-bit counter at the beginning of the Cell_clk. As the 9-bit counter is incremented by the bit clock, its output is sampled and latched at the next rising edge of the Ref_clk. Thus, the latch value (C1 to C9) indicates the phase difference between the Cell_clk and the Ref_clk at bit level.

However, to identify the timing difference for less than one bit is challenging. A novel sampling technique is adopted to adjust the phase down to 100 ps without using a 10-GHz clock. The signal S2, a variant Cell_clk that is aligned with the bit clock as shown in Figure 11.18, is used as a sample clock to determine the phase difference between the Cell_clk and the Ref_clk at the sub-bit level. The signal S3 mimics the Cell_clk but with a shorter duration and is used to produce four similar signals (F0 to F3), each separated by 100 ps. The signal S2 then samples the signals F0 to F3 as shown in the fine delay comparator (Fig. 11.19). The sampled outputs D0 to D3 show the phase difference within a bit. For instance, the sampled value in Figure 11.18 is 1100, corresponding to a phase difference of 100 ps. Note that the actual phase difference can be anywhere between 100 and 200 ps. Since the resolution is 100 ps, the phase adjustment error is limited to 100 ps.

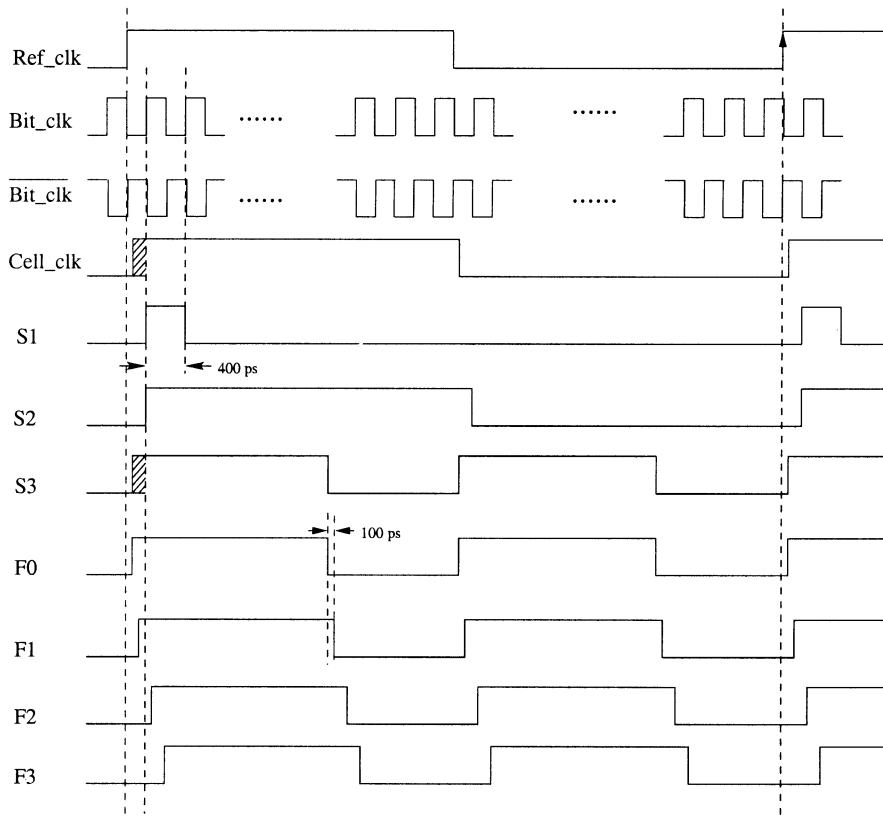


Fig. 11.18 Sample timing of the fine adjustment circuit ($\frac{1}{4}$ bit).

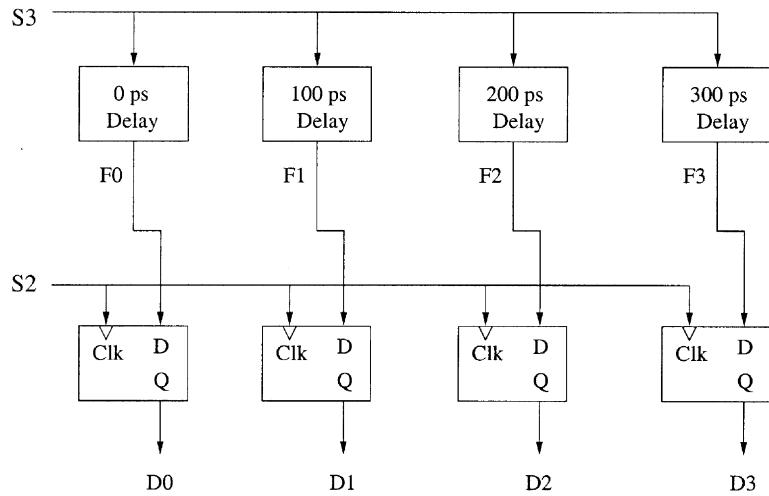


Fig. 11.19 Fine delay comparator.

TABLE 11.1 Conversion Table Between [D0, D1, D2, D3] and [C10, C11]

D0 (0 ps)	D1 (100 ps)	D2 (200 ps)	D3 (300 ps)	C10	C11
0	0	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	1
1	1	1	0	1	0
1	1	1	1	1	1

Table 11.1 shows the mapping of the fine sampled value (D0 to D3) to the last two bits of the delay line control (C10, C11).

With different combinations of C1 to C11, the optical delay elements are tuned to insert the desired delay needed to align the phase of incoming cells. For example, with $[C1, C2, \dots, C11] = [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1]$, a total delay of $T/2 + T/2^3 + T/2^{11}$ (the last term is $\frac{1}{4}$ bit) is added by the cell synchronization unit.

11.4 OPTICAL INTERCONNECTION NETWORK FOR TERABIT IP ROUTERS

11.4.1 Introduction

The tremendous increase in the speed of data transport on optical fibers has stimulated a large demand for gigabit multimedia services such as distance learning and video conferencing. The Internet, capable of supporting various information transport with the robust and reliable Internet protocol (IP), is widely considered as the most reachable platform of next-generation information infrastructure [33]. The key to the success of next-generation Internet (NGI) lies in the deployment of terabit IP routers to meet the exponential growth of multimedia traffic.

Although several large-capacity routers have been proposed [34, 35, 36, 37, 38, 39, 40, 41, 42, 43], building an IP router with multiterabit-per-second capacity still remains to be seen. The challenges of building a terabit IP router include: (1) implementing a large-capacity switch fabric providing high-speed interconnection for a number of smaller-capacity routing modules (RMs), and (2) devising a fast arbitration scheme resolving output contention within stringent time constraint while achieving high throughput and low delay.

By combining the strength of both optical and electronic technologies, a terabit packet switch architecture is proposed to use a nonblocking bufferless optical interconnection network (OIN) to interconnect multiple electronic RMs. The reason that an OIN is used rather than an electronic interconnection network (EIN) is to reduce the number of interconnection wires and to eliminate the problem of electrical–magnetic interference in the EIN. The

OIN takes advantage of the WDM technology by carrying multiple packets on different wavelengths and thus providing a large bandwidth for each interconnection fiber. As a result, the total number of interconnection wires is dramatically reduced.

The OIN uses a hierarchical broadcast-and-select approach to interconnect a number of optical transmitters and receivers. As the dense WDM technology matures, multiplexing 100 wavelengths in the 1550-nm window and amplifying them simultaneously become feasible [7]. In addition, the advance of the optical hybrid integration of planar lightwave circuits (PLCs) and semiconductor devices makes an optical system more cost-effective [44, 45, 46]. Thus, the OIN based on WDM and optical hybrid integration techniques will play an important role in providing terabit-per second switching capacity. One of its challenges is to synchronize optical data signals and electronic control signals. A couple of bits may be used as a guard time to circumvent the slower switching speed of optical devices. An adjustable electronic delay circuit can always be added to the control signals so that they can be tuned to align with the optical data signals.

In order to meet the stringent time constraint imposed on IP forwarding, especially IP table lookup, the operation of IP forwarding is separated from the construction of routing tables and the execution of routing protocols. IP forwarding is implemented in hardware at each RM, while the construction of routing tables and the execution of routing protocols is handled by the route controller (RC). Currently an Internet backbone router can have more than 40,000 routes in its routing table [47]. As the number of routes is still increasing, computing and maintaining the routing tables by the RC becomes more and more challenging. One solution could be using multiple RCs and interconnecting them to achieve both higher processing speed and load balancing. The RMs and the RCs can be interconnected either using dedicated buses, such as the peripheral component interconnect (PCI) bus, or through a core switch fabric, such as the OIN.

Input and output buffering together with internal speedup is used to achieve high switch performance and low loss rate at a reasonable complexity. Purely input buffering suffers HOL blocking [18] and thus has low throughput and large delay. In contrast, purely output buffering has the best delay-throughput performance, but is limited in size by the memory speed constraint. Input/output buffering and doubling of the switch's internal speed are used to achieve almost 100% throughput and low average input buffer delay.

A novel Ping-Pong arbitration (PPA) scheme [19] is proposed to resolve output port contention for both unicast and multicast packets. The basic idea is to divide the inputs into groups and apply arbitration recursively. The recursive arbiter is hierarchically structured, consisting of multiple small-size arbiters at each layer. The arbitration time of an N -input switch is proportional to $\log_4[N/2]$, when every four inputs are grouped at each layer. For a 256-input arbiter, our scheme can reduce the arbitration time to 11 gates delay, less than 5 ns using the current CMOS technology (much less than the

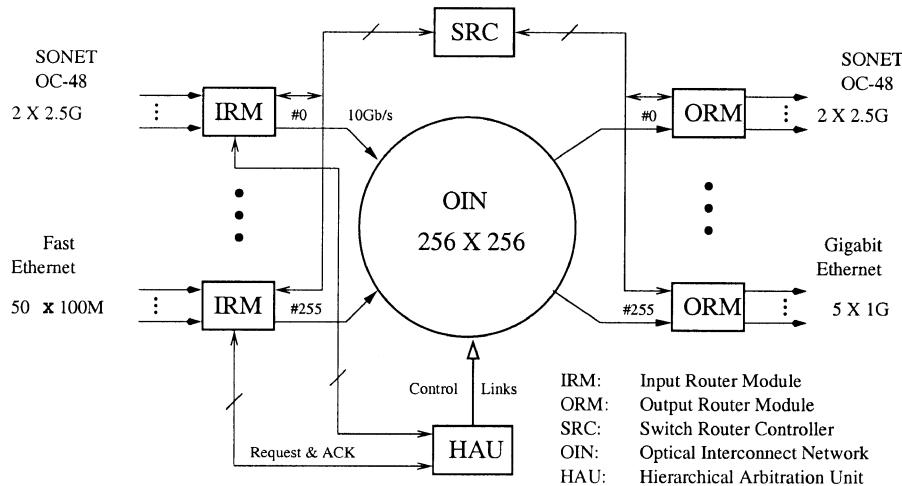


Fig. 11.20 Architecture of a terabit IP router. (©1998 IEEE.)

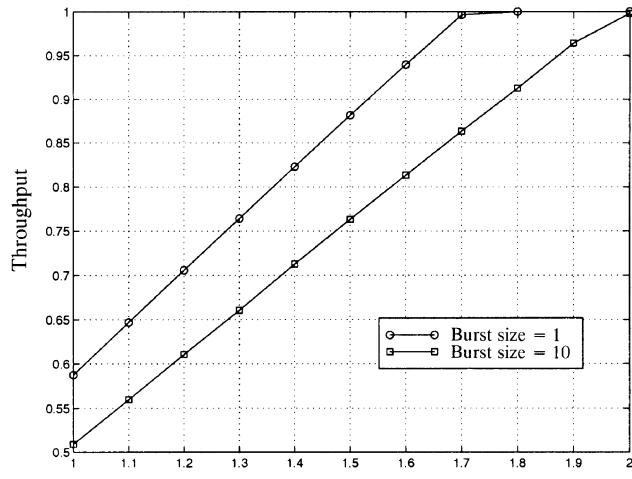
51.2 ns required for 64-byte segments transmitted at 10 Gbit/s), demonstrating that arbitration is no longer a bottleneck that limits the switch capacity.

11.4.2 A Terabit IP Router Architecture

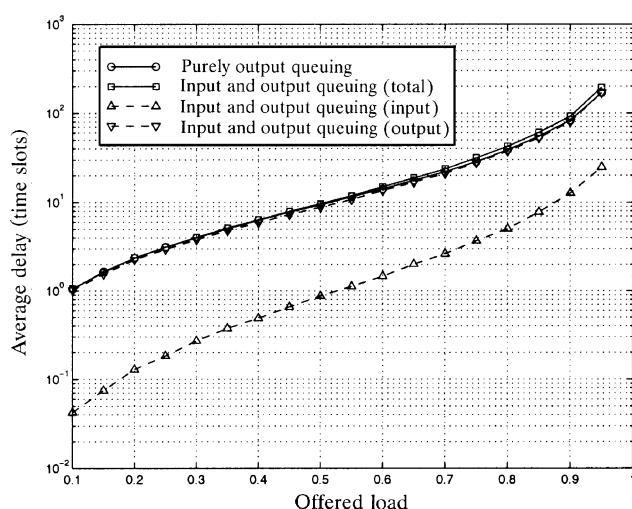
Figure 11.20 depicts four major elements in the terabit IP router [24]: the OIN supporting nonblocking and high-capacity switching, the Ping-Pong arbitration unit (PAU) resolving the output contention and controlling the switching devices, the RMs performing IP packet forwarding, and the RC constructing routing information for the RMs. There are two kinds of RM: input RM (IRM) and output RM (ORM). Both the IRMs and the ORMs implement IP packet buffering, route (table) lookup, packet filtering, and versatile interfaces, such as OC-3, OC12, OC-48, and Gigabit Ethernet. The interconnection between the RC and the RMs can be implemented with dedicated buses or through the OIN. Figure 11.20 simply illustrates the bus-based approach.

11.4.2.1 Speedup The fixed-length segment switching technique is commonly adopted in high-capacity IP routers to achieve high-speed switching and better system performance.

Figure 11.21(a) suggests that a speedup factor of two is required to achieve nearly 100% throughput under bursty traffic with geometric distribution and an average burst size of 10 packet segments. Figure 11.21(b) shows the corresponding average delay. The total average delay of input and output queuing is very close to the theoretical bound of purely output queuing. The input delay is an order smaller than the total delay, hinting that an input-



(a)



(b) Burst size = 10, speedup factor = 2

Fig. 11.21 Switch performance: (a) throughput, (b) average delay.

queued switch with speedup 2, on average, will perform as nearly well as a purely output-queued switch.

The speedup induces more challenges in two aspects: (1) doubling the switch transmission speed to 10 Gbit/s, and (2) halving the arbitration time constraint. The first challenge can be easily met with optical interconnection technology, while the second can be met by the PPA scheme described in Section 11.4.5.

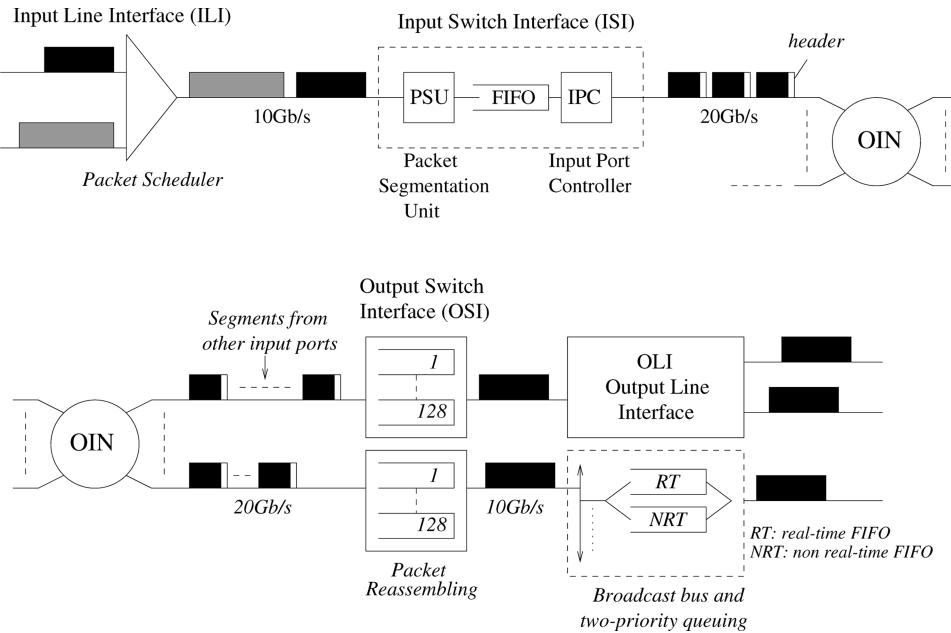


Fig. 11.22 The flow of packets across the router.

11.4.2.2 Data Packet Flow A data segment unit of 64 bytes is chosen to accommodate the shortest IP packets (40 bytes). Variable-length IP packets are segmented before being passed through the switch. Figure 11.22 depicts the flow of packets across the router. A simple round-robin (RR) packet scheduler is used at each input line interface (ILI) to arrange the packet arrivals from different interfaces (see also Fig. 11.20). It uses a FIFO buffer per interface to store incoming packets. Since the output line speed of the scheduler depends on all interfaces, it can be shown that the maximum packet backlog at each input line FIFO is just twice the maximum IP packet size, so the same large buffer can be chosen to avoid any packet loss.

The output packets of the scheduler enter the input switch interface (ISI) in which packet segmentation takes place. While a packet is being segmented, its IP header is first checked by the input packet filter (IPF) for network security and flow classification (i.e., inbound filtering), as shown in Figure 11.20. Afterwards, the header is sent to the input forwarding engine (IFE) for IP table lookup, deciding which ORM(s) this packet is destined for.

Data segments are stored in a FIFO while waiting for arbitration before being forwarded through the OIN. The forwarding sequence is packet by packet, not cell by cell, for each ISI, in order to simplify the reassembly. The input port number is added to each segment before it enters the OIN for ensuring correct packet reassembly at output ports.

Segments of a packet arriving at an output port may be interleaved with those from other input ports. While a packet is being reassembled, its IP

header can be sent to the output packet filter (OPF) for outbound filtering and then to the output forwarding engine (OFE) for another IP route lookup to decide which outgoing interface(s) this packet should be destined for. The packets are then broadcast at the output line interface (OLI) to all desirable interfaces. Each interface can maintain two FIFOs supporting two traffic priorities: real-time (RT) and non-real-time (NRT) packets.

11.4.3 Router Module and Route Controller

The RMs interconnected by the OIN perform packet forwarding in a distributed manner, as shown in Figure 11.20. The tasks of running unicast and multicast routing protocols and other control protocols are carried out by the RC centrally in order to expedite the data transport at each RM. Each RM mainly consists of input/output line interface (ILI/OLI), switch interface (ISI/OSI), packet filter (IPF/OPF), and forwarding engine (IFE/OFE). The line interfaces terminate OC3, OC12, OC-48, or Gigabit Ethernet connections. The switch interfaces bridge the line interfaces and the OIN for switching. The packet filters check inbound and outbound packets for network security and flow classification. The forwarding engines perform distributed IP route lookup. Below the forwarding engines and the switch interfaces are now described.

11.4.3.1 Input and Output Forwarding Engines The IP route lookup function is performed by the IFE and the OFE in each IRM and ORM, respectively, as shown in Figure 11.20. Several high-speed route lookup techniques have recently been proposed [48, 49, 50, 47, 51]. IFEs and OFEs show slight difference in performing packet forwarding.

An IFE, when receiving an incoming packet, will first validate its IP header. If the header is invalid, the packet is simply discarded. Otherwise, the IFE will determine whether this packet belongs to control messages or data, and then proceed accordingly. For control messages, the IFE forwards the packet via a control path, such as a PCI bus, or through the OIN to the RC. For data, depending on whether this packet is for unicast or multicast, the IFE will perform unicast or multicast table lookup, find out its destined ORM(s), and pass this information to the ISI. The ISI will then forward the packet via the OIN to the ORM(s), as will be described in Section 11.4.3.2.

Since the control messages are handled by IFEs, OFEs deal with data only. Refer to Section 11.4.2.2. An OFE performs similar IP route lookup for received packets, determines their outgoing interfaces to the next-hop router or host, and passes this information to the OSI. The OSI will then forward each packet to its outgoing interface(s), as will be described in the following.

11.4.3.2 Input and Output Switch Interfaces Figures 11.23 and 11.24 depict the components and the functions within an ISI and an OSI, respectively. The functions of an ISI include: (1) packet segmentation, (2) buffering

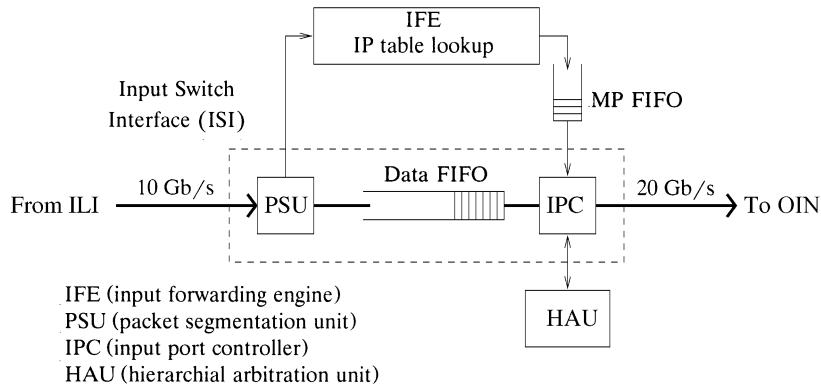


Fig. 11.23 Input switch interface.

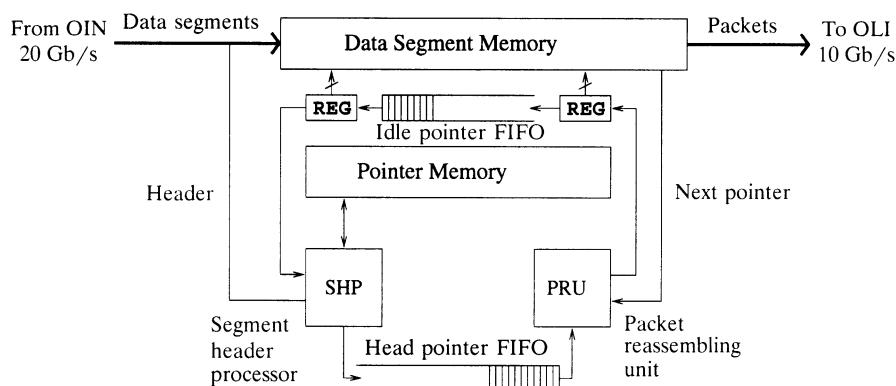


Fig. 11.24 Output switch interface.

of data segments, and (3) controlling the segment forwarding at the input port of the OIN. While a packet starts being segmented, its IP header is extracted and sent to the IPF for inbound filtering, then to the IFE, which handles IP table lookup and outputs the destination information [called the *multicast pattern* (MP)] of the packet into a MP FIFO. Meanwhile, a one-bit header is added to each segment to indicate whether it is the last segment of a packet. This information is used to determine the packet boundary. The segments are buffered in a data FIFO with its HOL segment entering the IPC for transmission. The IPC fetches the multicast pattern from the MP FIFO whenever the first segment of a packet comes. In every time slot, the IPC keeps sending the request signals to the PAU, waiting for the grant signals, and updating the multicast pattern for preparing the next request. If all requests are granted, which means the current segment has been forwarded to all its destinations, then the IPC reads a new segment from the

data FIFO and checks if it is the first segment of a packet. If not, the current MP is used for the segment. If so, the IPC gets a new MP from the MP FIFO and repeats the procedure. Upon being transmitted in the OIN, each segment is marked with a number identifying the input port in order to facilitate packet reassembly at the outputs.

At each OSI, data segments from the OIN are stored in the data segment memory, which is organized as linked lists connecting all segments of each packet as in a shared-memory switch. There is an idle-pointer FIFO keeping the unused memory pointers. A separate pointer memory indexed by the input port number is used to store the head and tail pointers of the linked lists of the packets being transmitted over the OIN to the output. Once all segments of a packet have arrived, the entire linked list is completed and the segment header processor (SHP) will forward its head pointer to the head pointer FIFO for packet departure scheduling. The head and tail pointers of the packet, which are no longer necessary, will be deleted from the pointer memory to prepare for the next packet arrival from the same input port. The head pointer FIFO keeps the packets in the order of their (last segments') arrival times. The packet reassembly unit (PRU) fetches the HOL packet and schedules its segment transmission.

11.4.3.3 Route Controller The RC in the router performs three main tasks: (1) executing routing protocols (such as RIP and DVMRP) and other control protocols (such as ICMP, IGMP, and SNMP), exchanging control messages via the RMs with neighbor routers and network management servers, and maintaining a routing information base (RIB) in the system; (2) based on the RIB, computing, updating, and distributing the IP unicast or multicast forwarding table [also called the forwarding information base (FIB)] for every RM; and (3) performing packet filter management. All the communications are through a control plane, which can be implemented either by a specific bus, such as a PCI bus, or by the core switch, such as the OIN.

Because trace data collected from a major Internet service provider (ISP) backbone router indicate that a few hundred updates can occur per second in the worst case, with an average of 1.04 updates per second [50], the RC needs to perform fast FIB updates. These include the operations of inserting a new route entry, modifying an old entry with new information such as new or additional outgoing interface(s) for this route, and removing an old entry due to timeout or route changes. Besides, since FIBs are usually organized in a treelike data structure in order to facilitate fast searching, changing a single route can affect a large number of relevant entries in an FIB. Many FIBs are to be updated if they all have this route entry.

To resolve the above challenge, it is suggested to let the RC directly access to each FIB rather than periodically download the full table to each RM, especially when there are only minor changes in the table. Consider the unicast case. Suppose each table size is 32 Mbyte [50]. The width of a PCI

bus is 64 bits, its frequency is 66 MHz. Simply downloading all 2×256 tables will take about 31 seconds, which is unacceptably large.

On the other hand, multiple RCs can be arranged in a hierarchical way to provide load balancing. One of the RCs can be responsible for maintaining a common RIB shared by the others in the system. Each RC will handle a number of RMs and this number depends on the processing power of the RC and the bandwidth of the interconnection (i.e., control plane) between the RCs and the RMs. The traffic on the control plane include routing information, network control messages, FIB updates, packet filter management messages, and other local hardware monitoring and diagnosis messages.

Another challenge is to design an efficient control structure for interconnecting the RCs and the RMs. One can either separate the control plane and data plane or let both share a common plane, such as the OIN, where control traffic should have higher priority than data. The advantage of using separate planes is to reduce the complexity of switch interconnection management and to improve the data throughput performance. Besides, the router will have better scalability in that the design of control plane and data plane can be optimized separately.

11.4.4 Optical Interconnection Network

Figure 11.25 shows the proposed 256×256 OIN, which can easily provide the multicast function due to its broadcast-and-select property. The OIN consists of two kinds of optical switching modules: input optical modules (IOMs) and output optical modules (OOMs). There are 16 of each kind in the OIN. Each IOM uses the same set of 16 different wavelengths (λ_1 to λ_{16}); each of the 16 input links at an IOM is assigned a distinct wavelength from the set, which carries packet segments under transmission. In each time slot, up to 16 packet segments at an IOM can be multiplexed by an arrayed-waveguide grating (AWG) router. The multiplexed signal is then broadcast to all 16 OOMs by a passive 1×16 splitter.

At each OOM, a 16×16 fully connected switching fabric performs the multicast switching function by properly controlling the semiconductor optical amplifier (SOA) gates. There are a total of 256 SOA gates in each OOM. At most 16 of them can be turned on simultaneously. The tunable filter, controlled by the PAU, is used to dynamically choose one of the 16 wavelengths in every time slot. As illustrated in Figure 11.26, it is assumed that a packet segment from the k th input link of the i th IOM is destined for the q th and 16th output links of the j th OOM, where $1 \leq i, j, k, q \leq 16$. These two multicast connections are established by turning on the SOA gates with index (i, j, q) and $(i, j, 16)$ only (the others are turned off). The tunable filters at the q th and 16th output links of the j th OOM are turned on with index k , which is provided by the PAU.

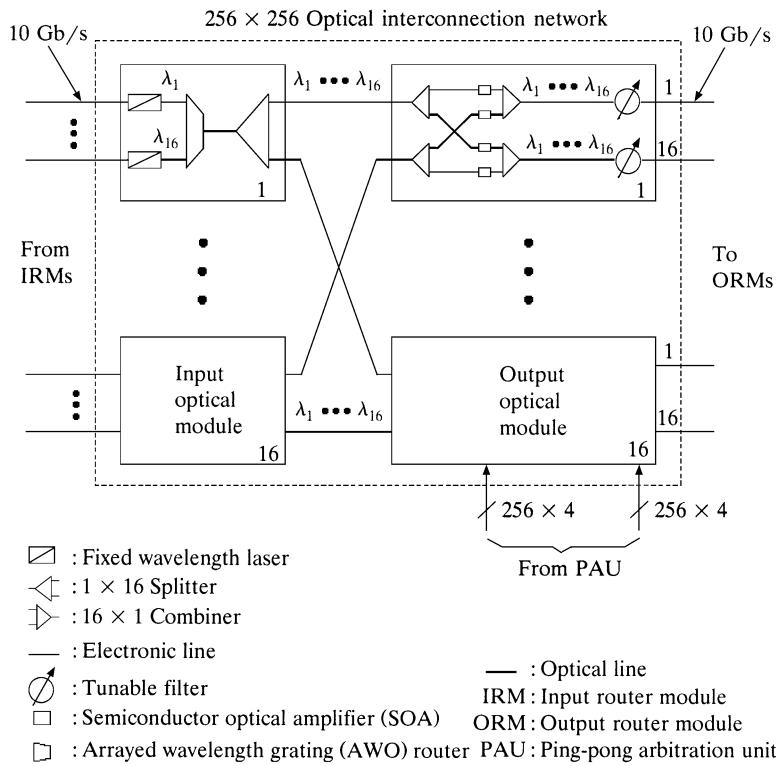


Fig. 11.25 256 × 256 optical interconnection network.

11.4.4.1 Input Optical Module The IOMs carry packets at 10 Gbit/s. At each IOM, distributed Bragg reflector (DBR) or distributed feedback (DFB) laser arrays can be used as the laser sources between 1525 and 1565 nm to match the gain bandwidth of commercially available EDFAs. Each EDFA can amplify multiple wavelengths simultaneously. Each input link of an IOM is connected to a laser with fixed wavelength.

To improve the chirp performance, a DFB laser diode integrated with an external modulator (EM) operating at 10 Gbit/s has been fabricated [52]. To ensure output power levels and chirp performance, an SOA and EMs can be integrated with the DFB laser arrays [53]. This monolithically integrated WDM source is able to provide multiwavelength capability and significantly reduce the cost per wavelength. In addition, it can also eliminate the alignment of fibers to individual lasers, reduce component count and coupling loss between components, and increase the reliability.

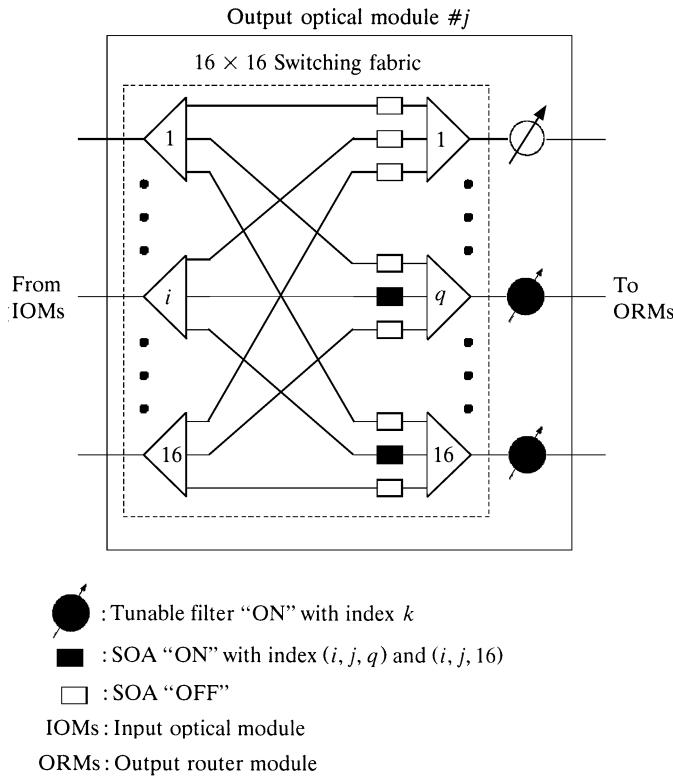
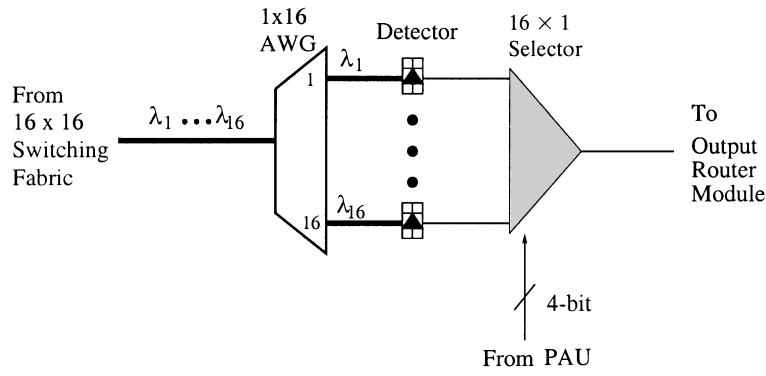


Fig. 11.26 Control in the j th output optical module.

11.4.4.2 Output Optical Module Each 16×16 switching fabric should be capable of connecting simultaneously two or more IOMs to all tunable filters at an OOM, so it needs to have broadcast capability and to be strictly nonblocking. As shown in Figure 11.26, a space switch can meet this requirement simply and can be constructed by using SOA gates.

In addition to their fast switching function (~ 1 ns), SOA gates can provide some gain to compensate the coupling loss and splitting loss caused by the splitters and combiners and the connection between discrete optical devices. Furthermore, SOA gates can be monolithically integrated with the passive couplers to enhance the reliability and loss performance between components.

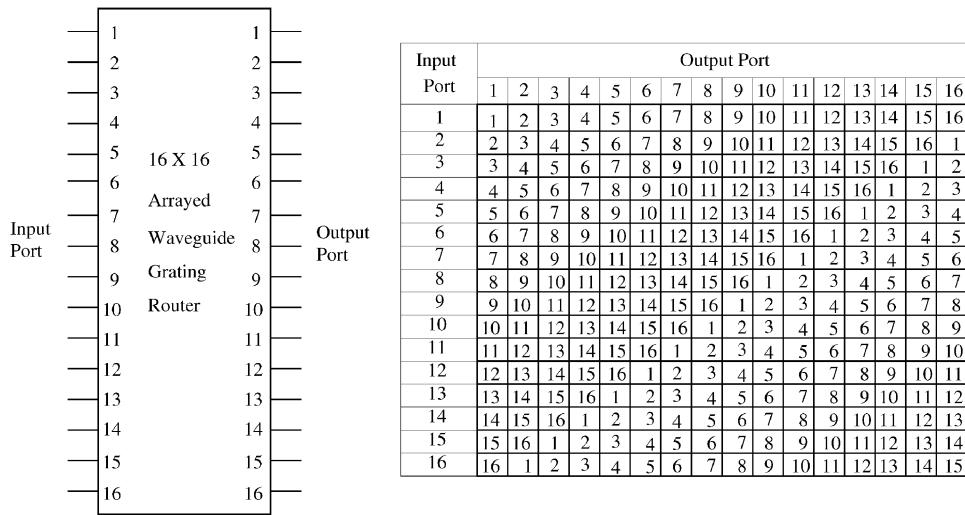
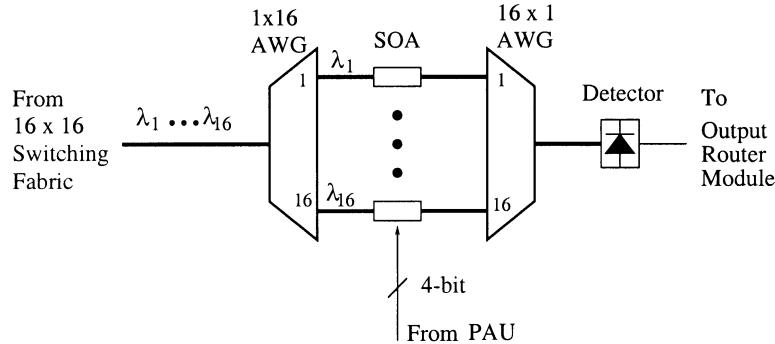
11.4.4.3 Tunable Filters Tunable filters are used to perform wavelength selection in the OIN. Three possible ways to implement the tunable filter are considered here.



AWG: Arrayed-Waveguide Grating
PAU: Ping-Pong Arbitration Unit

Fig. 11.27 Type-I tunable filter.

- **Type-I Tunable Filter:** The type-I tunable filter, as shown in Figure 11.27, performs the wavelength selection in the electrical domain. Each output of a 16×16 switching fabric is connected to a 1×16 AWG router, which is made from high-index indium phosphide (InP) material and is capable of demultiplexing 16 wavelengths in the 1550-nm window. Figure 11.28 shows the connectivity of a 16×16 AWG router. For example, if the WDM signal enters the 7th input port of the AWG router, only the 14th wavelength (λ_{14}) will be sent out through the 8th output port. Each demultiplexed wavelength is detected through a high-speed signal detector. Each detector has a laser waveguide structure and can be monolithically integrated with the AWG router, thus increasing the reliability and reducing the packaging cost of the AWG router. Finally, a 16×1 electronic selector is used to select the desired signal from the 16 detectors. The selector is controlled by the 4-bit control signal from the PAU. An alternative electronic selector is the InP-based optoelectronic integrated circuit (OEIC) receiver array [54], which operates at 10 Gbit/s per channel and integrates 16 p-i-n photodiodes with a heterojunction bipolar transistor (HBT) preamplifier.
- **Type-II Tunable Filter:** The type-II tunable filter, as shown in Figure 11.29, performs the wavelength selection optically. It has two AWGs. The first one, 1×16 , performs the demultiplexing, while the second, 16×1 , performs the multiplexing. By properly controlling the SOA gates, only one of the 16 wavelengths is selected. The selected wavelength passes through the second AWG and then is converted into an electronic signal by a detector. A PLC-PLC direct attachment technique [55] can be used to construct this type of tunable filter and to

Fig. 11.28 16×16 AWG router connectivity.

PAU: Ping-Pong Arbitration Unit

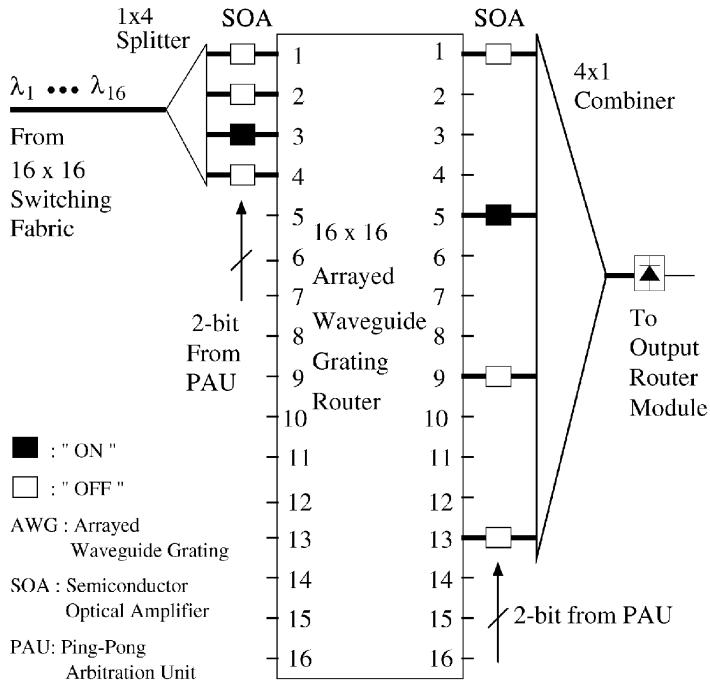
AWG: Arrayed-Waveguide Grating

SOA: Semiconductor Optical Amplifier

Fig. 11.29 Type-II tunable filter.

integrate the AWG routers and the SOA gates. This hybrid integration of PLC and SOA gates can reduce the coupling loss and increase the reliability.

- **Type-III Tunable Filter:** The type-III tunable filter, as shown in Figure 11.30, performs the wavelength selection optically. In contrast with the type-II filter, it uses only one 16×16 AWG router. Any one of the 16 wavelengths can be selected through its specific combination of SOA

**Fig. 11.30** Type-III tunable filter.

Input Port	Output Port			
	1	5	9	13
1	1	5	9	13
2	2	6	10	14
3	3	7	11	15
4	4	8	12	16

Fig. 11.31 Connectivity of type-III tunable filter.

gates at the input and output sides of the AWG router [56]. Figure 11.31 shows a way to choose any one of the 16 wavelengths. The 16×16 AWG router will route a wavelength λ_k from input port x ($x = 1, \dots, 16$) to output port y ($y = 1, \dots, 16$), where $k = (x + y - 1) \bmod 16$. For example, λ_7 will be selected as the output by turning on the 3rd SOA gate at the input side and the 5th SOA gate at the output side of the AWG router, respectively. The number of SOA gates in the type-III tunable filter is reduced by half; only 8 SOA gates (4 at the input and 4

at the output) are used instead of 16 SOA gates in the type-II tunable filter. However, compared to type-I and type-II tunable filters, the type-III tunable filter has more power loss, caused by the 1×4 splitter and the 4×1 combiner.

11.4.5 Ping-Pong Arbitration Unit

As shown in Figure 11.20, a centralized PAU is used in our router. The arbitration is pipelined with packet segment transmission in the OIN. In other words, while a HOL segment is being transmitted via the OIN, the segment next to it is also sending a request to the arbitration unit. In order to minimize the delay of forwarding multicast request signals, 256 parallel arbiters are used, each of which is associated with one output and handles 256 input request signals. The 256 incoming multicast request signals must be handled simultaneously within one time slot, that is, 51.2 ns for a 64-byte data segment sent at 10 Gbit/s.

11.4.5.1 Principles of Ping-Pong Arbitration Consider an N -input packet switch. To resolve its output contention, a solution is to use an arbiter for each output to fairly select one among those incoming packets and send back a grant signal to the corresponding input. The arbitration procedure is as follows:

1. During every arbitration cycle, each input submits a one-bit request signal to each output (arbiter), indicating whether its packet, if any, is destined for the output.
2. Each output arbiter collects N request signals, among which one input with active request is granted according to some priority order.
3. A grant signal is sent back to acknowledge the input.

Here, the second step that arbitrates one input among N possible ones is considered.

A simple RR scheme is generally adopted in an arbiter to ensure fair arbitration among the inputs. Imagine there is a token circulating among the inputs in a certain ordering. The input that is granted by the arbiter is said to grasp the token, which represents the grant signal. The arbiter is responsible for moving the token among the inputs that have request signals. The traditional arbiters handle all inputs together, and the arbitration time is proportional to the number of inputs. As a result, the switch size or capacity is limited by the available amount of arbitration time.

Here, it is suggested to divide the inputs into groups. Each group has its own arbiter. The request information of each group is summarized as a group request signal. Further grouping can be applied recursively to all the group request signals at the current layer, forming a tree structure, as illustrated in Figure 11.32. Thus, an arbiter with N inputs can be constructed using multiple small-size arbiters (ARs) in each layer. Different group sizes can be used.

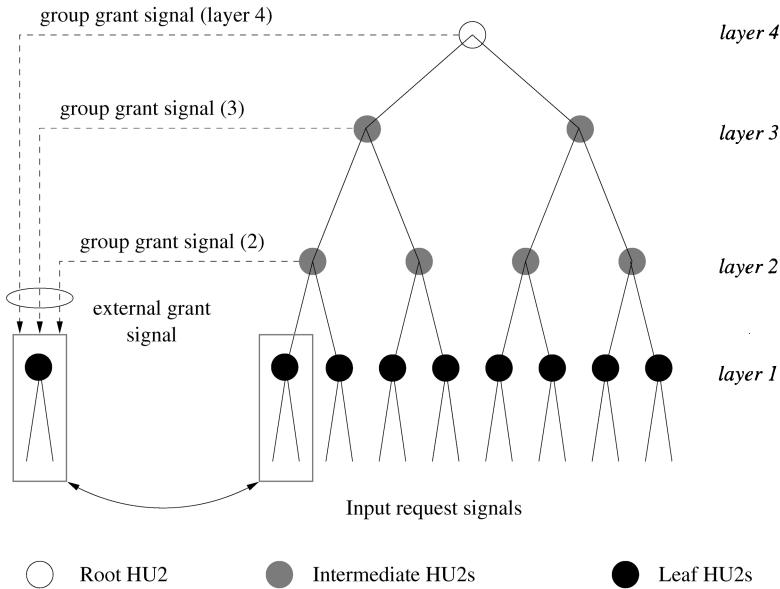
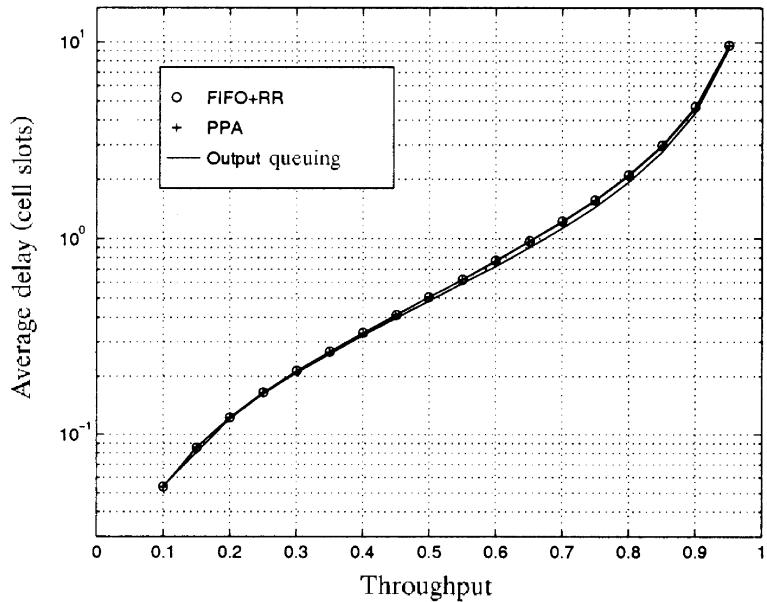


Fig. 11.32 A tree-structured hierarchical arbitration. (©1999 IEEE.)

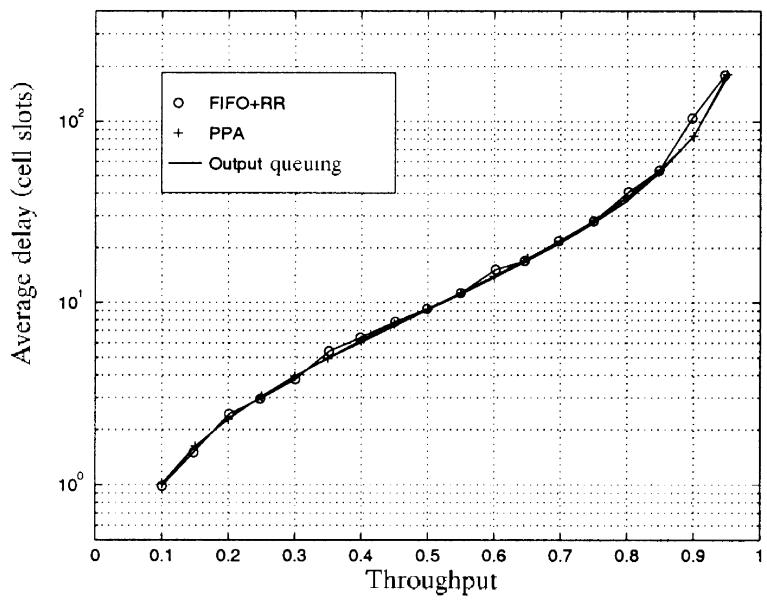
Assume $N = 2^k$. Figure 11.32 depicts a k -layer complete binary tree with a group size of two when $k = 4$. Let AR2 represent a two-input AR. An AR2 contains an internal flag signal that indicates which input is favored. Once an input is granted in an arbitration cycle, the other input will be favored in the next cycle. In other words, the granted request is always chosen between left (input) and right alternately. That is why it is called Ping-Pong arbitration. The first layer consists of 2^{k-1} arbiters called *leaf* AR2s. The next $k - 2$ layers consist of arbiters called *intermediate* AR2s, 2^{k-i} of which are in layer i . Finally, the last layer consists of only one arbiter called the *root* AR2.

Every AR2 has two request signals. An input request signal in layer i is the group request signal of 2^{i-1} inputs and can be produced by OR gates either directly or recursively. The grant signal from an AR2 has to be fed back to all the lower-layer AR2s related to the corresponding input. Therefore, every leaf or intermediate AR2 also has an external grant signal that ANDs all grant signals at upper layers, indicating the arbitration results of upper layers. The root AR2 needs no external grant signal. At each leaf AR2, the local grant signals have to combine the arbitration results of the upper layer (i.e., form its external grant signal) and provide full information on whether the corresponding input is granted or not.

One important use of the external grant signal is to govern the local flag signal update. If the external grant signal is invalid, which indicates that these two input requests as a whole are not granted at some upper layer(s), then



(a) Uniform traffic



(b) Bursty traffic

Fig. 11.33 Comparison of the PPA with FIFO + RR and output queuing: switch throughput and total average delay for a speedup factor of two.

the flag should be kept unchanged in order to preserve the original preference. As shown in Figure 11.32, the external grant signal of a leaf AR2 can be added at the final stage to allow other local logical operations to be finished while waiting for the grant signals from upper layers, which minimizes the total arbitration time.

Suppose N inputs are served in increasing order of their input numbers, i.e., $1 \rightarrow 2 \rightarrow \dots \rightarrow N \rightarrow 1$ under a RR scheme. Each AR2 by itself performs RR service for its two inputs. The Ping-Pong arbitration consisting of the tree of AR2s shown in Figure 11.32 can serve the inputs in the order of $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1$ when $N = 4$, for instance, which is still RR, if each input always has a packet to send and there is no conflict between any of the input request signals. Below its performance is shown by simulations.

11.4.5.2 Performance of PPA The performance of the PPA, FIFO + RR (FIFO for input queuing and RR for arbitration), and output queuing is compared here. A speedup factor of two is used for PPA and FIFO + RR. Simulation results are obtained from a 32×32 switch under uniform traffic (the output address of each segment is equally distributed among all outputs) and under bursty traffic (on-off geometric distribution) with an average burst length of 10 segments. The bursty traffic can be used as a packet traffic model with each burst representing a packet of multiple segments destined for the same output. The output address of each packet (burst) is also equally distributed among all outputs.

Figure 11.33 shows the throughput and total average delay of the switch under various arbitration schemes. It can be seen that the PPA performs comparably to the output queuing and the FIFO + RR. However, the output queuing is not scalable, and the RR arbitration is slower than the PPA. The overall arbitration time of the PPA for an N -input switch is proportional to $\log_4[N/2]$ when every four inputs are grouped at each layer. For instance, the PPA can reduce the arbitration time of a 256×256 switch to 11 gate delays, less than 5 ns using the current CMOS technology.

11.4.5.3 Implementation of PPA Multiple small arbiters can be recursively grouped together to form a large multilayer arbiter, as illustrated in Figure 11.32. Figure 11.34 depicts an n -input arbiter constructed by using $p q$ -input arbiters (AR- q), from which the group request and grant signals are incorporated into a p -input arbiter (AR- p). Constructing a 256-input arbiter starting from two-input arbiters as basic units is shown below.

Figure 11.35 shows a basic two-input arbiter (AR2) and its logical circuits. The AR2 contains an internally feedbacked flag signal, denoted by F_i , that indicates which input is favored.¹ When all G_g inputs are 1 (indicating that the two input requests R_0 and R_1 as a whole are granted by all the upper layers), once one input is granted in an arbitration cycle, the other input will be favored in the next cycle, as shown by the truth table in Figure 11.35(a). This mechanism is maintained by producing an output flag signal, denoted by

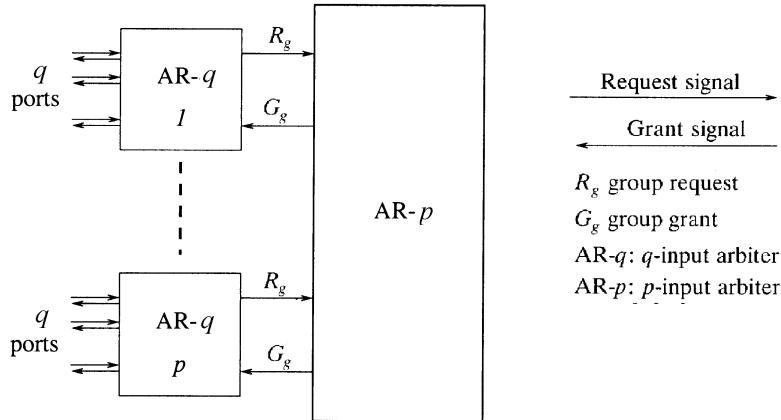


Fig. 11.34 Hierarchy of recursive arbitration with $n = pq$ inputs.

F_o , feedbacked to the input. Between F_o and F_i there is a D flip-flop which functions as a register forwarding F_o to F_i at the beginning of each cell time slot. When at least one of the G_g inputs is 0, indicating the group request of R_0 and R_1 is not granted at some upper layer(s), we have $G_0 = G_1 = 0$, $F_o = F_i$, that is, the flag is kept unchanged in order to preserve the original preference. As shown in Figure 11.35(b), the local grant signals have to be ANDed with the grant signals from the upper layers to provide full information whether the corresponding input is granted or not. G_g inputs are added at the final stage to allow other local logical operations to be finished in order to minimize the total arbitration time.

A four-input arbiter module (AR4) has four request signals, four output grant signals, one outgoing group request, and one incoming group grant signal. Figure 11.36(a) depicts our design of an AR4 constructed by three AR2s (two leaf AR2s and one intermediate AR2; all have the same circuitry), two two-input OR gates, and one four-input OR gate. Each leaf AR2 handles a pair of inputs and generates the local grant signals while allowing two external grant signals coming from upper layers: one from the intermediate AR2 inside the AR4, and the other from outside AR4. These two signals directly join at the AND gates at the final stage inside each leaf AR2 for minimizing the delay. Denote by R_{ij} and G_{ij} the group request signal and the group grant signal between input i and input j . The intermediate AR2 handles the group requests (R_{01} and R_{23}) and generates the grant signals (G_{01} and G_{23}) to each leaf AR2 respectively. It contains only one grant signal, which is from the upper layer for controlling the flag signal.

As shown in Figure 11.36(b), an AR16 contains five AR4s in two layers: four in the lower layer handling the local input request signals, and one in the upper layer handling the group request signals.

¹When the flag is LOW, R_0 is favored; when the flag is HIGH, R_1 is favored.

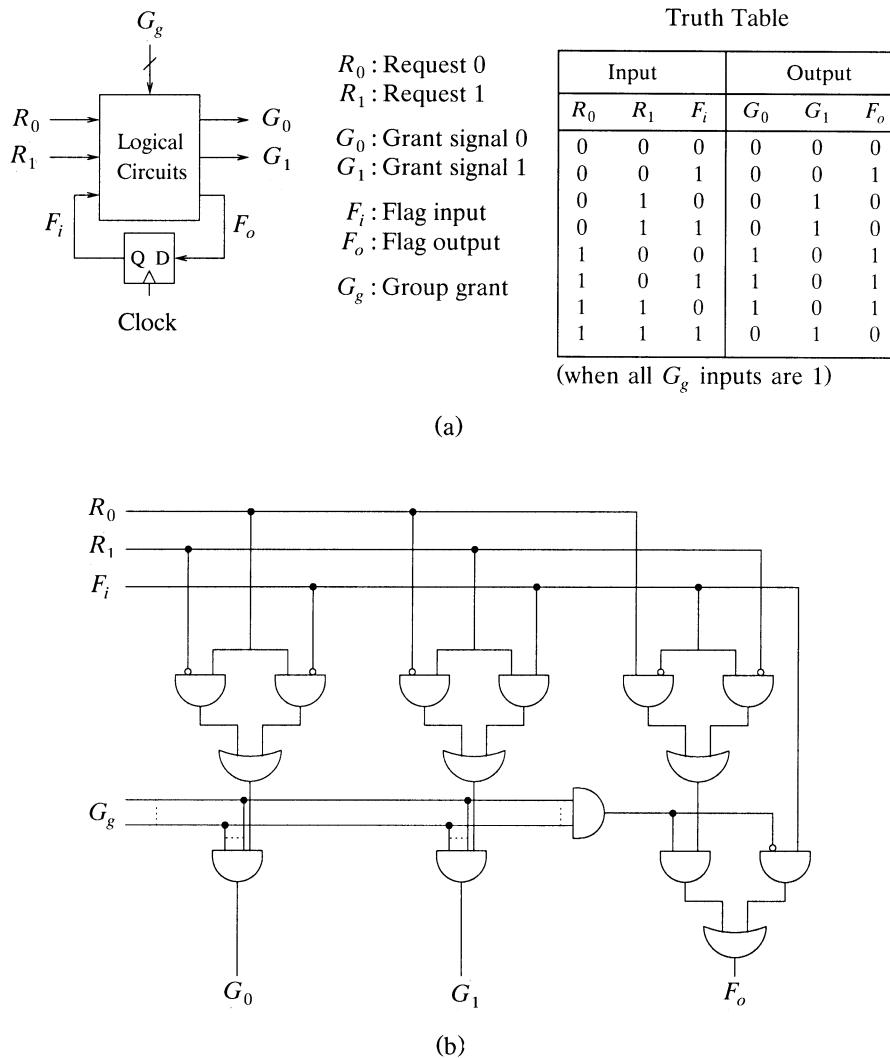


Fig. 11.35 (a) A two-input arbiter (AR2) and its truth table; (b) its logic circuits.
©1999 IEEE.)

Figure 11.37 illustrates a 256-input arbiter (AR256) constructed from AR4s, and its arbitration delay components. The path numbered from 1 to 11 shows the delay from when an input sends its request signal till it receives the grant signal. The first four gates delays (1–4) constitute the time for the input's request signal to pass through the four layers of AR4s and reach the root AR2, where one OR-gate delay is needed at each layer to generate the request signal [see Figure 11.36(a)]. The next three gate delays (5–7) constitute the time while the root AR2 performs its arbitration [see Figure

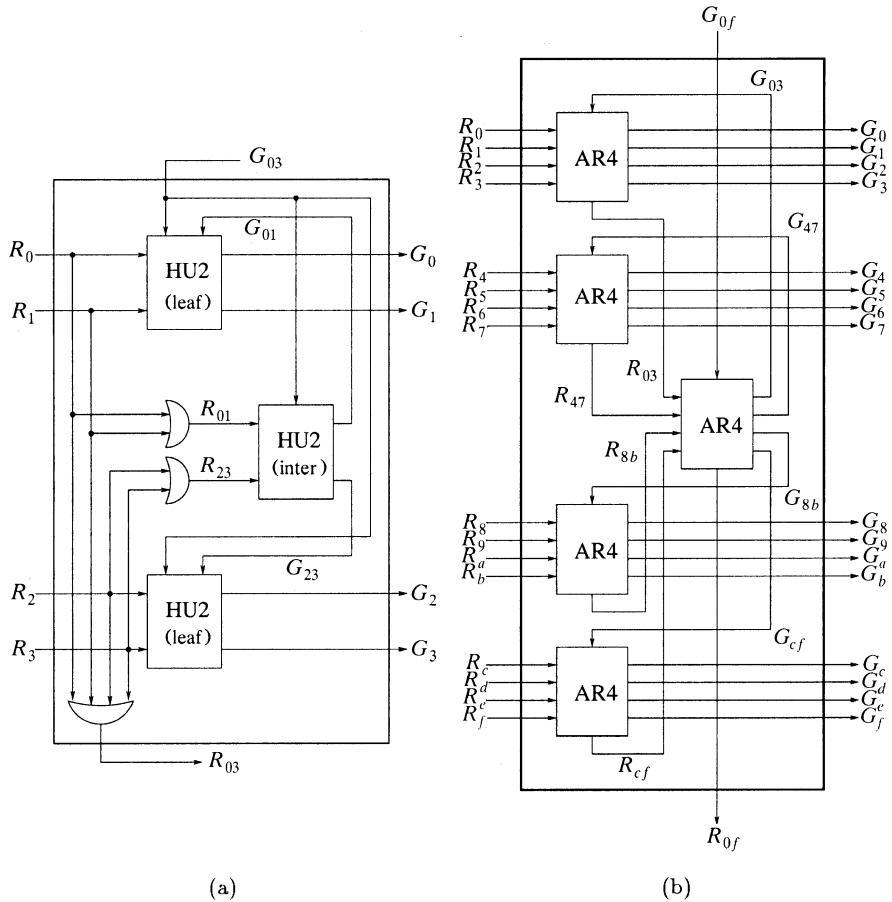


Fig. 11.36 (a) A 4-input arbiter (AR4) and (b) a 16-input arbiter (AR16) constructed from five AR4s. (©1999 IEEE.)

11.35(b)]. The last four gate delays (8–11) constitute the time for the grant signals in the upper layers to pass down to the corresponding input. The total arbitration time of an AR256 is then 11 gate delays. It thus follows that the arbitration time (T_n) of an n -input arbiter using such implementation is

$$T_n = 2 \log_4 \left\lceil \frac{n}{2} \right\rceil + 3. \quad (11.1)$$

11.4.5.4 Priority PPA Among the packets contending for the same output, those from real-time sessions are more delay-sensitive than the others from non-real-time sessions, so they should have higher priority. Therefore, sessions (and thus their packets) with various QoS requirements need to be

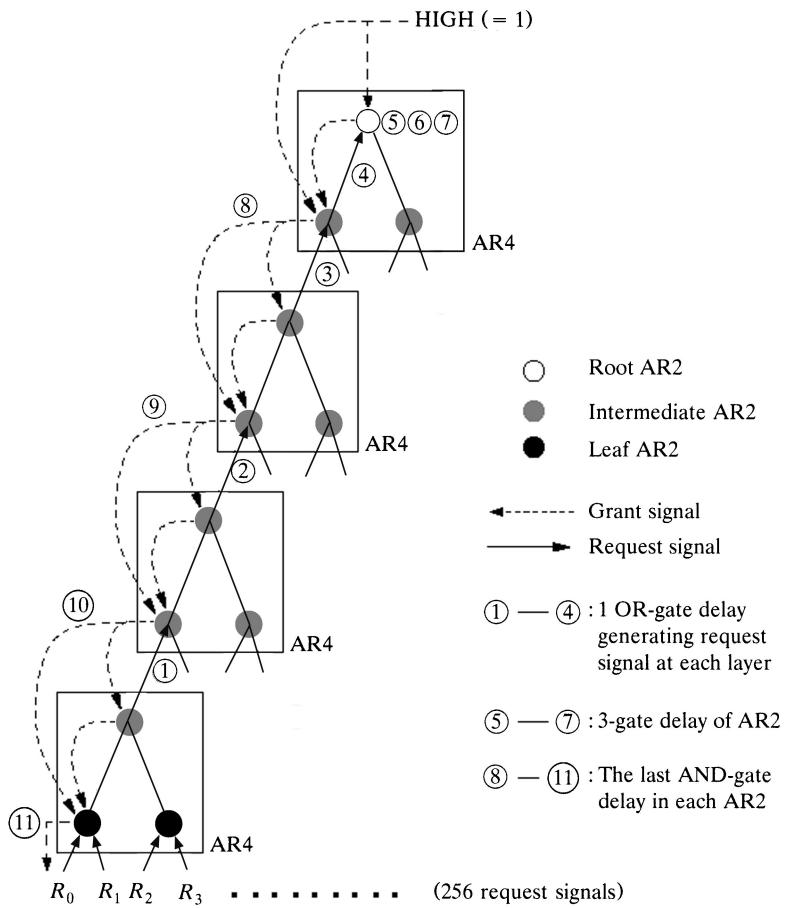


Fig. 11.37 Decomposition of arbitration delay in an AR256. (©1999 IEEE.)

assigned different levels of service priority. Below it is shown how to enhance the PPA for handling priority.

Two priority representations are used in our design for transfer efficiency and arbitration convenience, respectively. Suppose p levels of priority are supported. An input has altogether $p + 1$ states (including the case of no request), which can be represented by using $\lceil \log_2(p + 1) \rceil$ bits. The interlayer request information could be transferred either in series using one line or in parallel using multiple lines, depending on the tradeoff chosen between delay and pin count complexity. The serial-parallel format transformation can be realized by using shift registers.

A group of p lines is used in the second representation. At most one of them is HIGH, indicating that there is one request at the corresponding level of priority. There will be no request if all output lines are LOW.

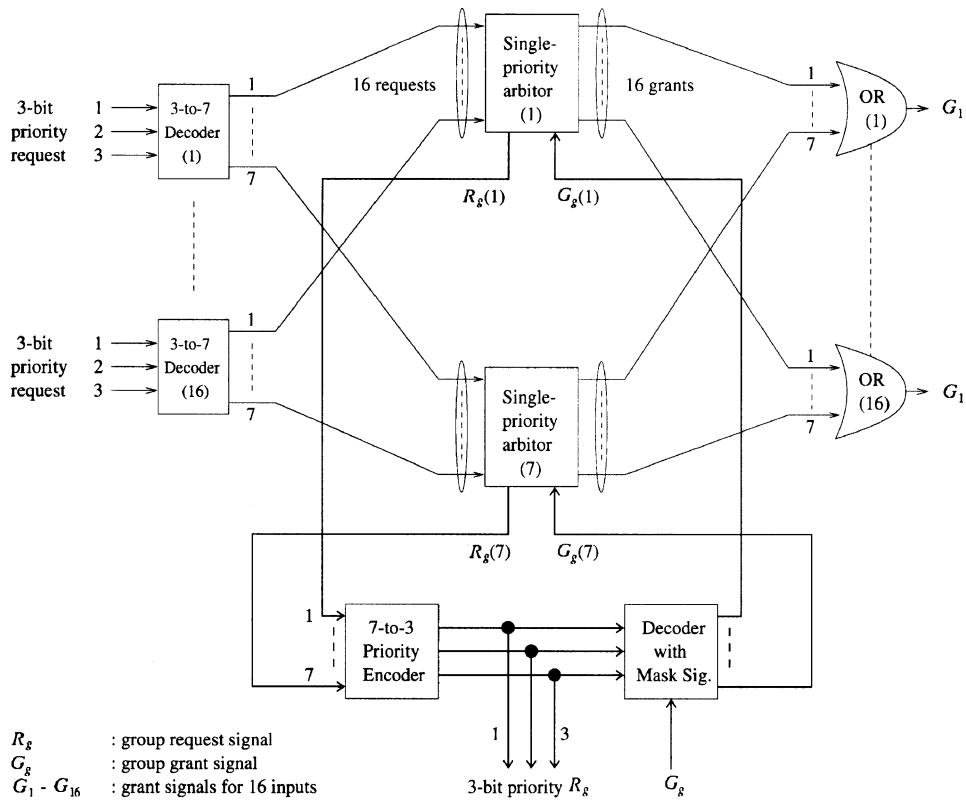


Fig. 11.38 Demonstration of priority handling with parallel arbitration: seven priority levels and 16 inputs.

Our solution to multipriority arbitration relies on a group of parallel single-priority arbiters to resolve the contention at each level of priority simultaneously. Multiple single-priority arbiters are necessary to maintain the arbitration states (states of the flip-flops) for each level of priority, which will be changed only when an input request at this priority level is granted. A preprocessing phase and a postprocessing phase are then added, as demonstrated in Figure 11.38, with a multipriority arbiter, which handles 16 inputs and seven levels of priority. A decoder is used at each input to decode the three-line priority request into seven single lines, each representing the request in the corresponding level of priority and entering the corresponding arbiter for single-priority contention resolution. An OR gate is used at each output to combine all corresponding local grants from the single-priority arbiters to produce the final grants for each input.

Meanwhile, every single-priority arbiter generates a group request signal for the upper layer's arbitration; it receives a group grant signal later, which indicates if this group of requests (at the corresponding level of priority) is granted or not. A priority encoder collects all the group requests from the

single-priority arbiters and indicates among them the highest priority with its three-line output. The outputs, in addition to being forwarded to the upper layer, will also be used to inhibit the arbiters with lower priority from producing any active grants. A decoder with its outputs masked by the upper-layer grant signal is used to decompose the output of the priority encoder into seven single-line grant signals, each for a single-priority arbiter. Only the arbiter at the corresponding level of priority will receive the upper layer's grant signal, while all the others will receive nothing but a LOW grant signal.

11.4.6 OIN Complexity

11.4.6.1 Component Complexity The complexity of the proposed 256×256 OIN is determined by the numbers of optical components and of interconnection lines between optical components and between the optical and electronic components. Figure 11.39 lists the optical components used in the OIN with different available wavelengths. The number of SOA gates dominates the component and interconnection complexity. The total number of SOA gates is equal to 256×32 when using type-II tunable filters with 16 wavelengths (i.e., $W = 16$), or 256×24 when using type-III tunable filters. The component complexity of the 256×256 OIN with respect to 8, 32, and 64 wavelengths is also shown in Figure 11.39. In addition, Figure 11.39 indicates that the total number of SOA gates of the switching fabrics at the OOMs decreases as the wavelength number W increases. However, the number of SOA gates in the type-II tunable filter is proportional to the number of wavelengths. The total number of SOA gates of the OIN based on the type-II tunable filter is equal to $256 \times 256/W + 256W$, where the first term represents the number of SOA gates used in the switching fabrics and the second term represents the number of the SOA gates used in type-II tunable filters. It is also shown that the OIN with 16 wavelengths based on type-II tunable filters has the fewest SOA gates. If type-III tunable filters are applied, the OIN with $W = 32$ or 64 has the fewest SOA gates.

Because the on-off switching of multiple wavelengths is performed at each SOA gate in the switching fabrics, the signal output of one wavelength varies according to the gain fluctuation induced by modulation of other wavelengths, even when the input power of the wavelength is constant. This is crosstalk induced by the gain saturation in the SOA gate and is often referred as cross-saturation. The impact of the cross-saturation on the performance of the OIN will become severe as the wavelength number passing through the SOA gate increases. To reduce this crosstalk in SOA gates, gain-clamped SOA gates [57] are necessary to provide a constant optical gain over a wider range of wavelength power as more wavelengths are used in the OIN.

To avoid the wavelength-dependent and polarization-dependent loss experiencing in the SOA gates, electroabsorption modulators (EAMs) [58] can be

	Component	Quantity	
Input Optical Module	Laser	256	
	8x1 AWG	32	
	1x32 Splitter	32	
Output Optical Module	1x8 Splitter	32x32	
	SOA Gates	256x32	
	32x1 Combiner	256	
	Type-II Tunable Filter	8x1 AWG SOA Gates	256x2 256x8
	Type-III Tunable Filter	8x8 AWG SOA Gates 1x2 Splitter 4x1 Combiner	256 256x6 256 256

$$\Sigma \text{ SOA Gates : (Type-II)} = 256 \times (32+8) = 256 \times 40 \\ (\text{Type-III}) = 256 \times (32+6) = 256 \times 38$$

(a) $W = 8$

	Component	Quantity	
Input Optical Module	Laser	256	
	16x1 AWG	16	
	1x16 Splitter	16	
Output Optical Module	1x16 Splitter	16x16	
	SOA Gates	256x16	
	16x1 Combiner	256	
	Type-II Tunable Filter	16x1 AWG SOA Gates	256x2 256x16
	Type-III Tunable Filter	16x16 AWG SOA Gates 1x4 Splitter 4x1 Combiner	256 256x8 256 256

$$\Sigma \text{ SOA Gates : (Type-II)} = 256 \times (16+16) = 256 \times 32 \\ (\text{Type-III}) = 256 \times (16+8) = 256 \times 24$$

(b) $W = 16$

	Component	Quantity	
Input Optical Module	Laser	256	
	32x1 AWG	8	
	1x8 Splitter	8	
Output Optical Module	1x32 Splitter	8x8	
	SOA Gates	256x8	
	8x1 Combiner	256	
	Type-II Tunable Filter	32x1 AWG SOA Gates	256x2 256x32
	Type-III Tunable Filter	32x32 AWG SOA Gates 1x4 Splitter 8x1 Combiner	256 256x12 256 256

$$\Sigma \text{ SOA Gates : (Type-II)} = 256 \times (8+32) = 256 \times 40 \\ (\text{Type-III}) = 256 \times (8+12) = 256 \times 20$$

(c) $W = 32$

	Component	Quantity	
Input Optical Module	Laser	256	
	64x1 AWG	4	
	1x4 Splitter	4	
Output Optical Module	1x64 Splitter	4x4	
	SOA Gates	256x4	
	4x1 Combiner	256	
	Type-II Tunable Filter	64x1 AWG SOA Gates	256x2 256x64
	Type-III Tunable Filter	64x64 AWG SOA Gates 1x8 Splitter 8x1 Combiner	256 256x16 256 256

$$\Sigma \text{ SOA Gates : (Type-II)} = 256 \times (4+64) = 256 \times 68 \\ (\text{Type-III}) = 256 \times (4+16) = 256 \times 20$$

(d) $W = 64$

Fig. 11.39 Component complexity of 256×256 optical interconnection network with different numbers of wavelengths, W .

used to perform faster switching (~ 100 ps) in multiple wavelength levels instead of the SOA gates in the proposed OIN. To compensate for the high coupling loss caused by the EAM, higher-power EDFA or EAMs integrated with semiconductor optical amplifiers will be necessary [59]. Here, the SOA-based OIN is considered because the SOA can provide high gain and fast switching (~ 1 ns) simultaneously.

11.4.6.2 Interconnection Complexity As a representative interconnection complexity, the interconnection between the electronic controller and the 256×256 OIN is evaluated. The interconnection complexity of the proposed 256×256 OIN is determined by the number of the SOA gates in the OOMs. The types of tunable filters and the number of wavelengths applied to the OIN are two dominant factors of the interconnection complexity, because they determine the number of SOA gates in the OOMs. Because the interconnection complexity with type-I and type-II tunable filters is the same, only type-II and type-III tunable filters are considered in this regard.

Figure 11.40 shows the interconnection complexity for different wavelengths ($W = 8, 16, 32$, and 64) and different types of tunable filters, where X is the complexity introduced by the switching fabrics, Y is the complexity represented by the tunable filters, and $X + Y$ gives the total interconnection complexity. Type-I and type-II tunable filters have the same interconnection complexity. For example, if $W = 16$, $X = Y = 256 \times 4$ when using type-I -II, or -III tunable filters. As shown in Figure 11.40, the total interconnection complexity, $X + Y$, is equal to 256×8 , and is independent of the wavelength number and the type of tunable filter used in the OIN.

11.4.7 Power Budget Analysis

Power loss in the OIN, which is due to splitting loss, depends on the number of wavelengths and the size of the switching fabric of each OOM. To compensate for power loss in the OIN, optical amplifiers, such as EDFA

Wavelength Number	Interconnection Complexity				
	X	Y		X+Y	
		Type-I/II	Type-III	Type-I/II	Type-III
W=8	256 x 5	256 x 3	256 x 3	256 x 8	256 x 8
W=16	256 x 4	256 x 4	256 x 4	256 x 8	256 x 8
W=32	256 x 3	256 x 5	256 x 5	256 x 8	256 x 8
W=64	256 x 2	256 x 6	256 x 6	256 x 8	256 x 8

X: Interconnection complexity with respect to switching fabrics

Y: Interconnection complexity with respect to tunable filters

Fig. 11.40 Interconnection complexity of the 256×256 OIN for different types of tunable filter and different wavelength numbers W .

and SOA gates, have to provide wide bandwidth to accommodate multiple wavelengths with uniform gain in the 1550-nm window. At each IOM, to avoid the problems introduced by direct modulation of lasers, such as large frequency chirp, an external modulator (EM) is used to obtain an optical on-off keying (OOK) signal.

Figure 11.41 shows the optical signal path through the OIN with type-II tunable filters and $W = 16$. All the optical components are represented with their gains or losses. For simplicity of modeling and analysis, the OIN is assumed to be fully loaded and all the possible connections between the input–output pairs established. Second, optical signals are represented by their optical power corresponding to bit mark–space data signals. Third, frequency chirp due to the EM and dispersion effects of all the components are not taken into consideration since signals are transmitted only in the

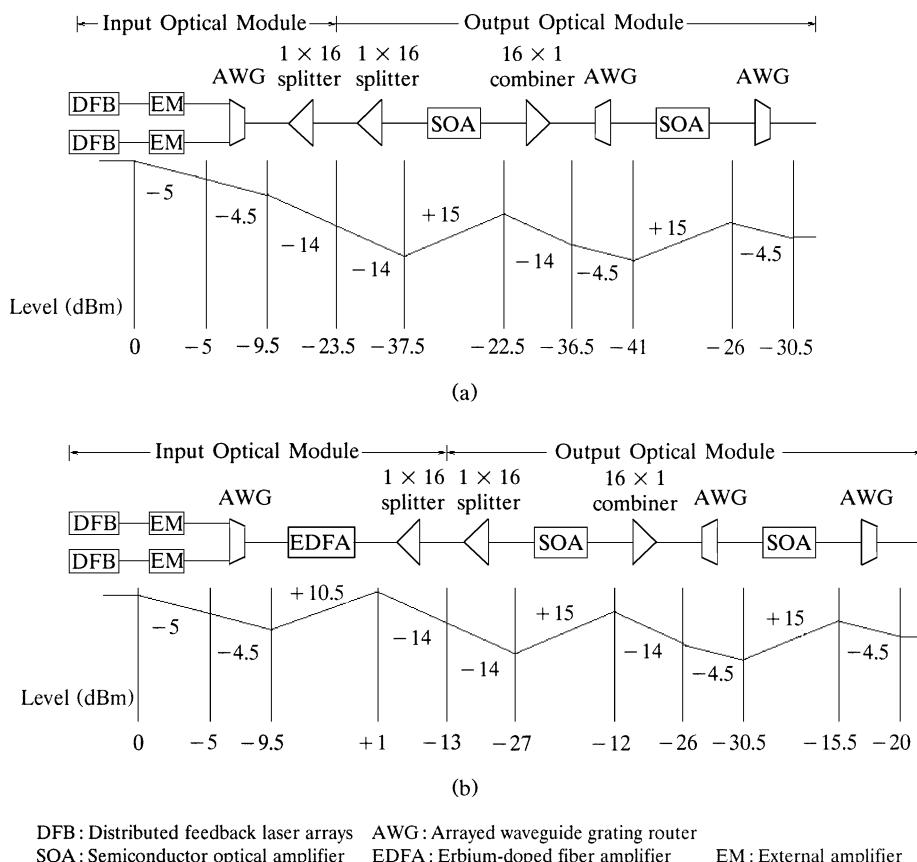


Fig. 11.41 Power budget of the 256 × 256 optical interconnection network with 16 wavelengths and type-II tunable filters: (a) before and (b) after power compensation by EDFA.

short-length fibers in the OIN. Finally, all the light reflections in the system are ignored and all the passive components are assumed to be polarization-insensitive.

Figure 11.41(a) shows that, without power compensation, the received power is only -30.5 dBm, much less than required sensitivity of about -20 dBm at the bit error rate (BER) of 10^{-12} for signals at 10 Gbit/s. To increase the received optical power, optical amplifiers are necessary to provide sufficient gain to compensate for the power loss in the OIN. In the proposed OIN, there are two kinds of optical amplifiers to perform the power compensation function. One is the SOA and the other is the EDFA. As shown in Figure 11.41(b), an EDFA with 10.5 -dB gain at each IOM is used to amplify 16 wavelengths simultaneously, so that the sensitivity at the receiver is increased to -20 dBm. Note that the gain provided by the EDFA needs to be increased to 20 dB if the AWG, which is near the output link of type-II tunable filter, is replaced by a 16×1 combiner.

11.4.8 Crosstalk Analysis

In the OIN, crosstalk is caused by the finite on/off ratio (i.e., the ratio of gain on and gain off, or G_{ON}/G_{OFF}) of an SOA gate. More specifically, $P_{x,gate} = P_{in,gate} \times G_{OFF}$, where $P_{x,gate}$ is the output power of an SOA gate in the off state, and $P_{in,gate}$ is the input power of the SOA gate. Basically, crosstalk components are categorized into two types. One, referred to as *incoherent* crosstalk, can be taken into account by power addition. The second type, *homowavelength* crosstalk, occurs when the signal channels and their beats fall within the receiver electrical bandwidth.

The on and off states of SOAs, as switching gates, are controlled by their injection currents. An SOA in the off state (without injection current) absorbs incident optical power and blocks the signal channel. However, some optical power leaks from the SOA even when it is in the off state. The power leakage becomes crosstalk when it is combined with the signal at an output port of OIN.

It is assumed that the OIN is fully loaded and packets arriving at all of its input ports are sent to their destined output ports. Only one output port of the OIN, as shown in Figure 11.42, is considered.

Figure 11.42 shows the crosstalk path in an OOM, where up to 16 WDM signals from 16 different IOMs are sent to the 16×16 switching fabric (see position A in the figure). Each WDM signal can carry up to 16 different wavelengths at each output port in the switching fabric. For each output of the OOM, only one of these 16 WDM signals will be selected and delivered to the tunable filter. That is, for each output of the 16×16 switching fabric only one SOA will be turned on and the other 15 SOA gates will be turned off (see position B in the figure). Thus, there is only one dominant WDM signal, and up to 15 WDM crosstalk channels are combined with it to be sent to the tunable filter at each output switch module (OSM) (position C in the figure).

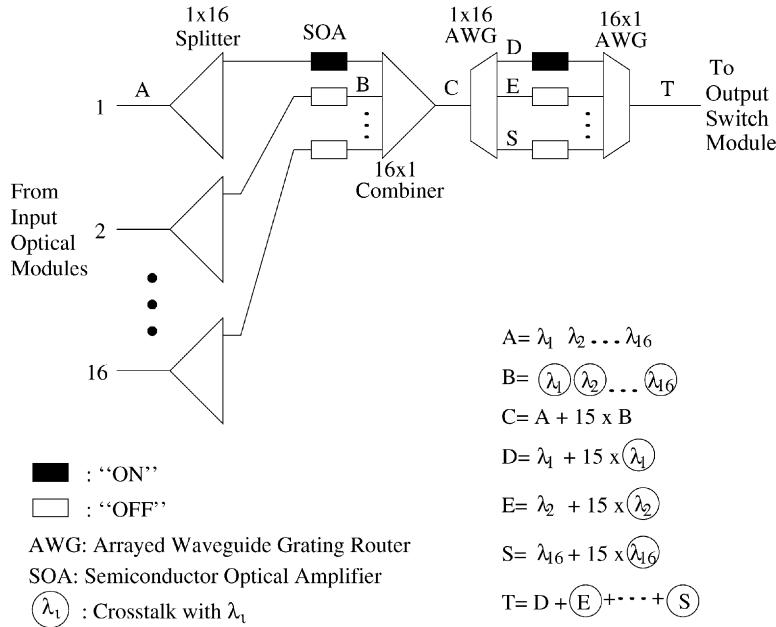


Fig. 11.42 Crosstalk path of the 256×256 optical interconnection network.

The tunable filter at each OSM selects the wavelength of the selected WDM signal, which is demultiplexed by an AWG to 16 different wavelengths, and one of them is chosen by turning on one SOA (position D in the figure) and turning off the other 15 (positions E to S in the figure). The finally selected wavelength is sent to the OSM through another AWG. Here, it is assumed that λ_1 from the first IOM is chosen for the OSM. The receiver at the OSM (position T in the figure) will receive not only the selected wavelength (λ_1), but also incoherent crosstalk and homowavelength crosstalk (those λ_i 's with circles in the figure). Homowavelength crosstalk channels come from different IOMs with the same selected wavelength, and incoherent crosstalk channels contain different wavelengths from the selected wavelength at the receiver.

There are two sources that contribute to the incoherent crosstalk channels in the proposed OIN. One is from wavelengths other than the selected one in the same IOM, and the other is from wavelengths other than the selected one in the other 15 IOMs. Of these two sources of incoherent crosstalk, the former is dominant, and the latter is negligible because it passes through two SOA gates in the off state, one in the switching fabric and the other in the tunable filter.

Figure 11.43 shows the BER as a function of the received optical power and crosstalk ratio (CR, the reciprocal of the on/off ratio) of an SOA gate. To meet a BER requirement of 10^{-12} , one has to increase the input power by about 2 dB to compensate for the crosstalk with CR = -20 dB. Further-

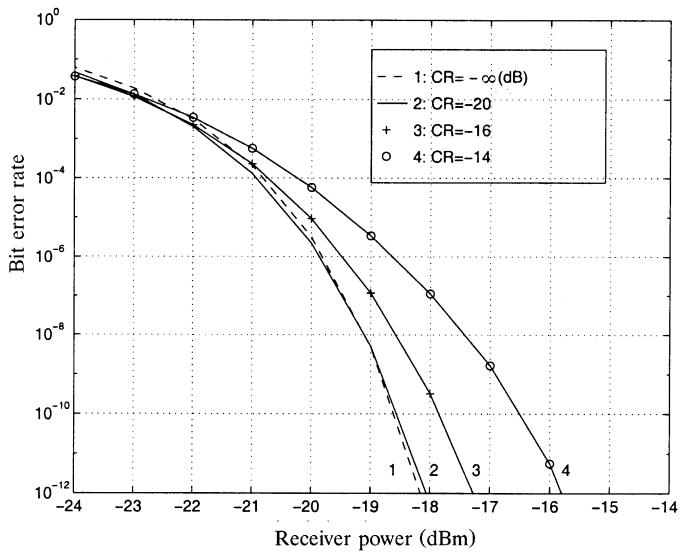


Fig. 11.43 Bit error rate versus received power with different crosstalk ratios (CR) of the SOA gate; the signal extinction ratio is 20 dB.

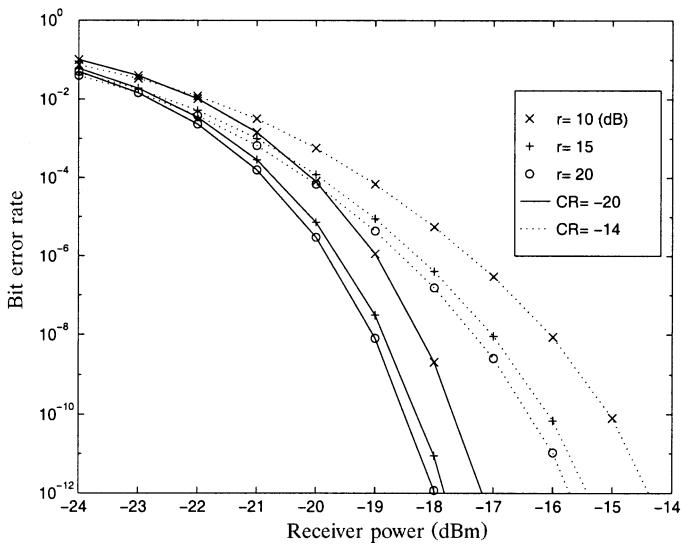


Fig. 11.44 Bit error rate vs. received power with different extinction ratios and for CR = -20 and -14 dB.

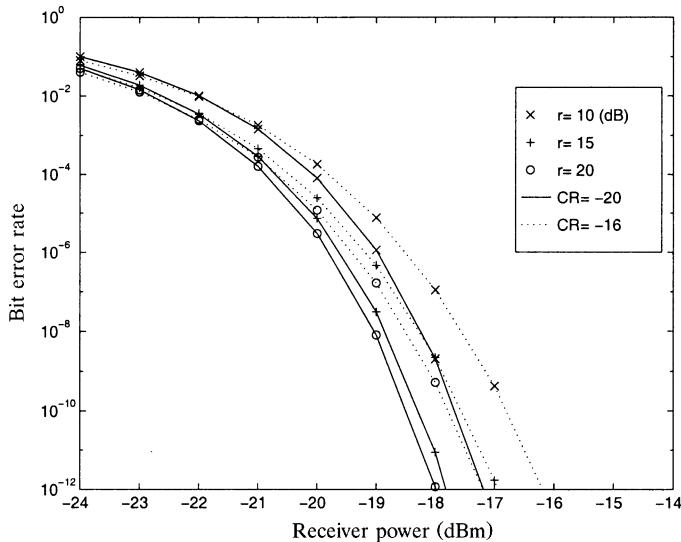


Fig. 11.45 Bit error rate vs. received power with different extinction ratios (r) and with $\text{CR} = -20$ and -16 dB.

more, to compensate for the power penalty for the CR of -16 dB, the input power needs to be increased by 1 dB. In order to reduce the gain saturation effect of the SOA gate due to high input power, gain-clamped SOA gates [57] can be used to provide a constant optical gain (≈ 22 dB) over a wider range of input power.

With the same system parameters as those used in [60, 61], Figure 11.44 shows the BER performance using different signal extinction ratios [i.e., the ratio of the power of logic one (mark) to the power of logic zero (space)], r , with crosstalk ratios of -20 and -14 dB. As shown in Figures 11.44 and 11.45, the power penalty for these three signals with $r = 20, 15$, and 10 dB strongly depends on the BER. That is, the signals with lower extinction ratios have a larger space power, so that crosstalk channels can be significant. To alleviate the performance degradation due to the incoherent or homowavelength crosstalk, one should make the signal extinction ratio sufficiently high so that the beating between the signal data spaces and the crosstalk is negligible. A signal extinction ratio of 15 dB will be appropriate with a power penalty of about 0.1 dB and a CR of -20 to -16 dB.

REFERENCES

1. P. E. Green, *Fiber Optic Communication Networks*, Prentice Hall, 1992.
2. N. V. Srinivasan, “Add-drop multiplexers and cross-connects for multiwavelength optical networking,” *Tech.Dig., OFC '98*, San Jose, CA, pp. 57–58, 1998.

3. C. K. Chan, F. Tong, L. K. Chen, and K. W. Cheung, "Demonstration of an add-drop network node with time slot access for high-speed WDMA dual bus/ring packet networks," *Tech. Dig., OFC '98*, San Jose, CA, pp. 62–64, 1998.
4. G. Chang, G. Ellinas, J. K. Gamelin, M. Z. Iqbal, and C. A. Brackett, "Multi-wavelength reconfigurable WDM/ATM/SONET network testbed," *J. Lightwave Technol.*, vol. 14, pp. 1320–1340.
5. R. E. Wanger, R. C. Alferness, A. A. M. Saleh, and M.S. Goodman, "MONET: multiwavelength optical networking," *J. Lightwave Technol.*, vol. 14, pp. 1349–1355, 1996.
6. S. Okamoto and K. Sato, "Optical path cross-connect systems for photonic transport networks," *Proc. IEEE Global Telecommun. Conf.*, pp. 474–480, 1993.
7. A. K. Srivastava et al., "1 Tb/s transmission of 100 WDM 10 Gb/s channels over 400 km of TrueWave™ fiber," *Postdeadline Papers, OFC '98*, San Jose, CA, pp. PD10-1–4, 1998.
8. A. S. Tanenbaum, *Computer Networks*, Prentice Hall, 1981.
9. Z. Hass, "The 'staggering switch': an electrically controlled optical optical packet switch," *IEEE Lightwave Technol.*, vol. 11, no. 5/6, pp. 925–936, May/Jun. 1993.
10. D. Chiaroni, C. Chauzat, D. De Bouard, M. Sotom, "Sizeability analysis of a high-speed photonic packet switching architecture," *Proc. 21st Eur. Conf. on Opt. Comm. (ECOC '95)*, Paper We.P.47, pp. 793–796, 1995.
11. G. H. Duan, J. R. Fernandez, and J. Garabal, "Analysis of ATM wavelength routing systems by exploring their similitude with space division switching," *IEEE Int. Con. on Communication*, vol. 3, pp. 1783–1787, Jun. 1996.
12. K. Sasayama, Y. Yamada, K. Habara and K. Yukimatsu, "FRONTIERNET: frequency-routing-type time-division interconnection network," *Lightwave Technol.*, vol. 15, no. 3, pp. 417–429, Mar. 1997.
13. O. Ishida, H. Takahashi, and Y. Inoue, "Digitally tunable optical filters using arrayed-waveguide grating (AWG) multiplexers and optical switches, *J. Lightwave Technol.*, vol. 15, no. 2, pp. 321–327 1997.
14. E. Arthurs, M. S. Goodman, H. Kobrinski, and M. P. Vecchi, "HYPASS: an optoelectronic hybrid packet switching system," *IEEE Selected Areas Commun.*, vol. 6, no. 9, pp. 1500–1510, Dec. 1988.
15. T. T. Lee, M. S. Goodman and E. Arthurs, "STAR-TRACK: a broadband optical multicast switch," Bellcore Technical Memorandum Abstract, TM-ARH-013510, 1989.
16. B. Bingham and H. Bussey, "Reservation-based contention resolution mechanism for batcher-banyan packet switches," *Electron. Lett.*, vol. 24, no. 13, pp. 772–773, Jun. 1988.
17. A. Cisneros and C. A. Brackett, "A large ATM switch based on memory switches and optical star couplers," *IEEE Selected Areas Commun.*, vol. 9, no. 8, pp. 1348–1360, Oct. 1991.
18. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space division packet switch," *IEEE Trans. Commun.*, vol. COM-35, no. 12, Dec. 1987.
19. H. J. Chao, X. Guo, and C. H. Lam, "A fast arbitration scheme for terabit packet switches," *IEEE GLOBECOM '99*, pp. 1236–1243, Dec. 1999.

20. E. Munter, J. Parker, and P. Kirkby, "A high-capacity ATM switch based on advanced electronic and optical technologies," *IEEE Commun. Mag.*, pp. 64–71, Nov. 1995.
21. Y. Nakahira, H. Inoue, and Y. Shiraishi, "Evaluation of photonic ATM switch architecture—proposal of a new switch architecture," *Int. Switching Symp.*, pp. 128–132, 1995.
22. F. S. Choa and H. J. Chao, "On the optically transparent WDM ATM Multicast (3M) switches," *Fiber Integrated Opt.*, vol. 15, pp. 109–123, 1996.
23. H. J. Chao, L. Wu, Z. Zhang, S. H. Yang, L. M. Wang, Y. Chai, J. Y. Fan, and F. S. Coa, "A photonic front-end processor in a WDM ATM multicast switch," *IEEE J. Lightwave Technol.*, vol. 18, no. 3, pp. 273–285, 2000.
24. H. J. Chao and T-S. Wang, "Design of an optical interconnection network for terabit IP router," *Proc. IEEE LEOS '98*, vol. 1, pp. 233–234, 1998.
25. Y. Chai, J. H. Chen, F. S. Choa, J. P. Zhang, J. Y. Fan, and W. Lin, "Scalable and modularized optical random access memories for optical packet switching networks," *Proc. CLEO '98*, CThO17, 1998.
26. Y. Chai, J. H. Chen, X. J. Zhao, J. P. Zhang, J. Y. Fan, F. S. Choa, and W. Lin, "Optical DRAMs using refreshable WDM loop memories," *Proc. ECOC '98*, Madrid, Spain, 1998.
27. R. Langenhorst, et al., "Fiber loop optical buffer," *IEEE J. Lightwave Technol.*, vol. 14, no. 3, pp. 324–335, 1996.
28. G. Bendelli, et al., "Photonic ATM switch based on a multiwavelength fiber-loop buffer," *Proc. OFC '95*, Paper WJ4, pp. 141–142.
29. Y. Yamada, et al., "Transparent optical-loop memory for optical FDM packet buffering with differential receiver," *Proc. ECOC '96*, pp. 317–320, Sep. 1996.
30. Y. Takahashi, K. Ando, M. Miyata, and E. Amada, "New retiming and synchronization scheme for optical ATM switching systems," *Electron. Lett.*, vol. 26, no. 2, pp. 99–100, Jan. 1990.
31. M. Burzio, P. Cinato, R. Finotti, P. Gambini, M. Puleo, E. Vezzoni, and L. Zucchelli, "Optical cell synchronization in an ATM optical switch," *Proc. ECOC '94*, Firenze, pp. 581–584, 1994.
32. L. Zucchelli, M. Burzio, and P. Gambini, "New solution for optical packet delineation and synchronization in optical packet switched networks," *Proc. ECOC '96*, Oslo, pp. 301–304, 1996.
33. *Research Challenges for the Next Generation Internet*, J. E. Smith and F. W. Weingarten, Eds., Computing Research Association, May 1997.
34. A. Asthana, C. Delph, H. V. Jagadish, and P. Krzyzanowski, "Towards a gigabit IP router," *J. High Speed Networks*, vol. 1, pp. 281–288, 1992.
35. Avici System, Inc., "The world of terabit switch/router technology," White Paper, 1997. http://www.avici.com/html/white_paper.html.
36. Cisco Product Document, "Cisco 12000 gigabit switch router," Sep. 1997.
37. S. Keshav and R. Sharma, "Issues and trends in router design," *IEEE Commun. Mag.*, pp. 144–151, May 1998.
38. V. P. Kumar, T. V. Lakshman, and D. Stiliadis, "Beyond best effort: router architectures for the differentiated services of tomorrow's internet," *IEEE Commun. Mag.*, pp. 152–164, May 1998.

39. NetStar, Inc., "NetStar's GigaRouter." Product Document, 1994.
<http://www.sresearch.com/search/130000.htm#prodinfo>.
40. P. Newman, G. Minshall, T. Lyon, and L. Huston, "IP switching and gigabit routers," *IEEE Commun. Mag.*, Jan. 1997.
41. C. Partridge, et al., "A fifty gigabit per second IP router," *IEEE/ACM Trans. Networking*, 1997.
42. G. Parulkar, D. C. Schmidt, and J. S. Turner, " $a^I t^P m$: a Strategy for Integrating IP with ATM," *ACM Comput. Commun. Rev.*, vol. 25, no. 4, Oct. 1995.
43. A. Tantawy, O. Koufopavlou, M. Zitterbart, and J. Abler, "On the design of a multi-gigabit IP router," *J. High Speed Networks*, vol. 3, pp. 209–232, 1994.
44. K. Kato, M. Ishii, and Y. Inoue, "Packaging of large-scale planar lightwave circuits," *Proc. IEEE Components and Technology Conf.*, pp. 37–45, 1997.
45. T. L. Koch and U. Koren, "Photonic integrated circuits," *ATT Tech. J.*, pp. 63–74, Jan./Feb. 1992.
46. K. Okamoto, "Application of planar lightwave circuits to optical communication systems," *Proc. 21st Eur. Conf. on Opt. Commun. (ECOC '95)*, Paper Mo.B.4.1, pp. 75–82, 1995.
47. S. Nilsson and G. Karlsson, "Fast address lookup for Internet routers," *Proc. IFIP 4th Int. Conf. on Broadband Communications*, pp. 11–22, Apr. 1998.
48. A. Brodnik, S. Carlsson, M. Degermark, and S. Pink, "Small forwarding tables for fast routing lookups," *Proc. ACM SIGCOMM*, pp. 3–14, Aug. 1997.
49. W. Doeringer, G. Karjoth, and M. Nassemi, "Routing on longest matching prefixes," *IEEE/ACM Trans. Networking*, vol 4, no. 1, pp. 86–97, Feb. 1996.
50. P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," *Proc. IEEE INFOCOM*, Session 10B-1, Mar. 1998.
51. M. Waldvogel, G. Varghese, J. S. Turner, and B. Plattner, "Scalable high speed IP routing lookups," *Proc. ACM SIGCOMM*, pp. 25–36, Aug. 1997.
52. D. Lesterlin, S. Artigaud, and H. Haisch, "Integrated laser/modulators for 10 Gbit/s system," *Proc. 22nd European Conf. on Optical Communication (ECOC '96)*, Paper WeD.3.1, pp. 3.183–3.190, 1996.
53. M. G. Young, et al., "A 16×1 wavelength division multiplexer with integrated distributed Bragg reflector laser and electroabsorption modulators," *IEEE Photon. Technol. Lett.*, vol. 5, no. 8, pp. 908–910, Aug. 1993.
54. K. C. Syao, K. Yang, X. Zhang, G. I. Haddad, and P. Bhattacharya, "16-channel monolithically integrated InP-based p-i-n/HBT photoreceiver array with 11-GHz channel bandwidth and low crosstalk," *Optical Fiber Commun. (OFC '97)*, Paper TuD5, 1997.
55. I. Ogawa, et al., "Lossless hybrid integrated 8-ch optical wavelength selector module using PLC platform and PLC-PLC direct attachment techniques," *Optical Fiber Commun. Conf. (OFC '98)*, Postdeadline Paper, pp. PD4.1–PD4.4, 1998.
56. O. Ishida, H. Takahashi, and Y. Inoue, "Digitally tunable optical filters using arrayed-waveguide grating (AWG) multiplexers and optical switches," *J. Lightwave Technol.*, vol. 15, no. 2, pp. 321–327, 1997.

57. S. L. Danielsen et al., "Detailed experiment and theoretical investigation and comparison of the cascadability of semiconductor optical amplifier gates and gain-clamped semiconductor optical amplifier gates," *Tech. Digest of Optical Fiber Commun. Conf. (OFC '98)*, vol. 2, pp. 41–42, 1998.
58. T. Ido, M. Koizumi, S. Tanaka, M. Suzuki, and H. Inoue, "Polarization and wavelength insensitive MQW electroabsorption optical gates for WDM switching systems," *IEEE Photon. Technol. Lett.*, vol. 8, no. 6, pp. 788–790, Jun. 1996.
59. F. Devaux et al, "High-speed tandem of MQW modulators for coded pulse generation with 14-dB fiber-to-fiber gain," *IEEE Photon. Lett.*, vol. 33, no. 2, pp. 218–220, Feb. 1996.
60. Y. Jin and M. Kavehrad, "An optical cross-connect system as a high-speed switching core and its performance analysis," *J. Lightwave Technol.*, vol. 14, pp. 1183–1197, Jun. 1996.
61. R. Ramaswami and P. A. Humblet, "Amplifier induced crosstalk in multichannel optical networks," *J. Lightwave Technol.*, vol. 8, pp. 1882–1896, Dec. 1990.

CHAPTER 12

WIRELESS ATM SWITCHES

The goal of next-generation wireless systems is to enable mobile users to access ubiquitous multimedia information anytime anywhere. New mobile and wireless services include Internet access to interactive multimedia and video conferencing as well as traditional services such as voice, email, and Web access. The extension of broadband services to the wireless environment is being driven mainly by the increasing demand for mobile multimedia services coupled with the advent of high-performance portable devices such as laptop PCs and personal digital assistants [1, 2].

In recent years, wireless ATM has drawn much attention as a solution for QoS-based mobile multimedia services. It has been an active topic of research and development in many organizations worldwide [2, 3, 4, 5, 6] and is now under standardization within applicable bodies such as the ATM Forum [7] and ETSI.

Wireless ATM is intended to provide seamless support of qualitatively similar multimedia services on both fixed and mobile terminals. The realization of wireless ATM raises a number of technical issues that need to be resolved, however. First, there is a need for the allocation and standardization of appropriate radio frequency bands for broadband communications. Second, new radio access and other wireless-channel-specific functions are required to operate at high speed. For example, a high-speed radio physical layer, a medium access control (MAC), and a data link control (DLC) layer are necessary for implementing wireless ATM. Next, mobility management is required to support personal and terminal mobility. Mobility management involves two aspects: location management and handoff management. Location management must be capable of tracking mobile users for delivery of an

incoming call as they move around the network. Handoff management must be capable of dynamically reestablishing virtual circuits to new access points without disconnecting communication between a mobile terminal and its peer. A mobility-support ATM switch must guarantee in-sequence and loss-free delivery of ATM cells when it is involved in handoff. Finally, wireless ATM should provide uniformity of end-to-end QoS guarantees. However, providing such guarantees is not easy, due to limited wireless bandwidth, time-varying channel characteristics and terminal mobility.

The remainder of this chapter is organized as follows. Section 12.1 outlines various reference configurations for a wireless ATM architecture and a wireless ATM protocol architecture. The wireless ATM protocol architecture is based on incorporation of wireless access and mobility-related functions into the existing ATM protocol stack. Section 12.2 reviews some recent proposals to build wireless ATM systems and related research work. Section 12.3 describes wireless-specific protocol layers. A radio physical layer, a MAC layer, and a DLC layer are summarized in that section. Section 12.4 discusses handoff and rerouting schemes. It also discusses cell routing and cell loss in a crossover switch during handoff. Section 12.5 introduces a mobility-support ATM switch architecture that can avoid cell loss and guarantee cell sequence during handoff. Performance of the switch is also discussed in that section.

12.1 WIRELESS ATM STRUCTURE OVERVIEWS

12.1.1 System Considerations

Within the Wireless ATM Working Group (WAG) of the ATM Forum, various reference configurations for a wireless ATM architecture are discussed [8, 9]. In a *fixed wireless network* scenario, the network components, switching elements, and end user devices (terminals) are fixed. The fixed wireless users are not mobile but connect over wireless media. This is the simplest case of wireless access provided to an ATM network, without any mobility support. Examples of this kind of service are fixed wireless LANs and network access or network interconnection via satellite or microwave links. In a mobile ATM network scenario, mobile end user devices communicate directly with the fixed network switching elements. The communication channel between the mobile end user devices and the switching elements can be wired or wireless. Mobile wired users change their points of attachment into the network over time, though connections are not maintained during times of movements. In contrast, mobile wireless users can maintain their connections while changing their points of attachment into the network.

The next scenario is *wireless ad hoc networks* where there is no access node available. For example, such a network may consist of several wireless mobile terminals gathered together in a business conferencing environment.

The last scenario is ATM networks supporting *personal communications service* (PCS) access. In this scenario, the end user devices are PCS terminals. This scenario uses the mobility-supporting capabilities of the fixed ATM network to route traffic to the appropriate PCS base station. A PCS-to-ATM interworking function (IWF) is required to translate the data stream, which is delivered to the en users via a wireless link or delivered to the ATM network via a mobility-support ATM switch. The IWF resides in the PCS base station controller (BSC), which controls several base stations and manages wireless resources.

For the wireless access architecture, two approaches are under consideration: integrated access and modular access. An integrated access model incorporates all mobility and radio functions into ATM switches. This approach places more complexity in the switches. In a modular access model, ATM switches are alienated from the radio access mechanisms. ATM switches implement call and connection control and mobility management aspects of wireless ATM, whereas a separate physical entity, known as the *access point* (AP), implements the radio access functions and deals with all the radio-specific functionality, such as radio MAC and radio resource management functions. This approach reduces the effect of radio and mobility functions on the switches, but requires a new protocol, called *access point control protocol* (APCP) [10], to convey messages between the APs and the ATM switches.

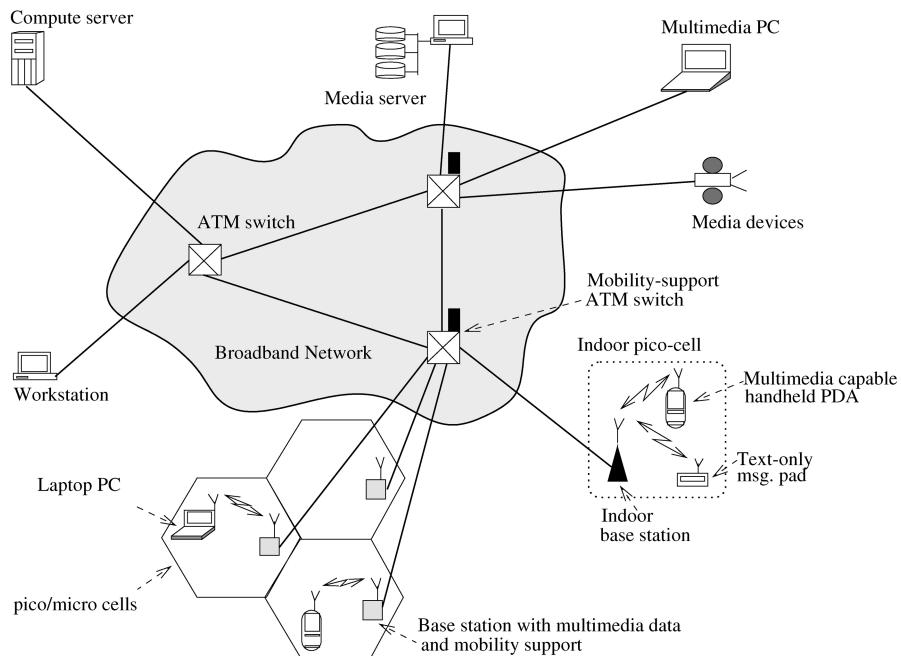


Fig. 12.1 A wireless ATM system architecture.

Figure 12.1 illustrates a typical wireless ATM architecture. Fixed wired end user devices and media servers are connected to regular ATM switches. In a mobile wireless ATM environment, the wireless ATM system has a geographical cell structure like traditional cellular systems. The whole coverage area (indoor or outdoor area) of the wireless ATM system is divided into cells (microcells or picocells). Each cell has one base station, which is an AP to the wired ATM network. Several base stations are connected to a mobility-support ATM switch. Mobile wireless terminals within a cell communicate with a base station through the wireless medium to each the ATM network.

12.1.2 Wireless ATM Protocol

A wireless ATM system needs to provide seamless support of qualitatively similar multimedia services on both fixed and mobile wireless users. The objective here is to provide mobile wireless users with the full range of services they would enjoy if connected via traditional wired media. To support this wireless ATM feature, a set of new protocols is required. This includes wireless channel-specific protocols such as a MAC protocol and a DLC protocol for error correction. In addition, the mobility feature requires extensions to existing ATM control protocols for handoff, location management, dynamic routing and QoS.

Figure 12.2 shows a wireless ATM reference configuration with related protocol stacks [8].

In the end user system protocol stack, support for mobility resides both on the user plane and on the control plane. The user plane requires a wireless

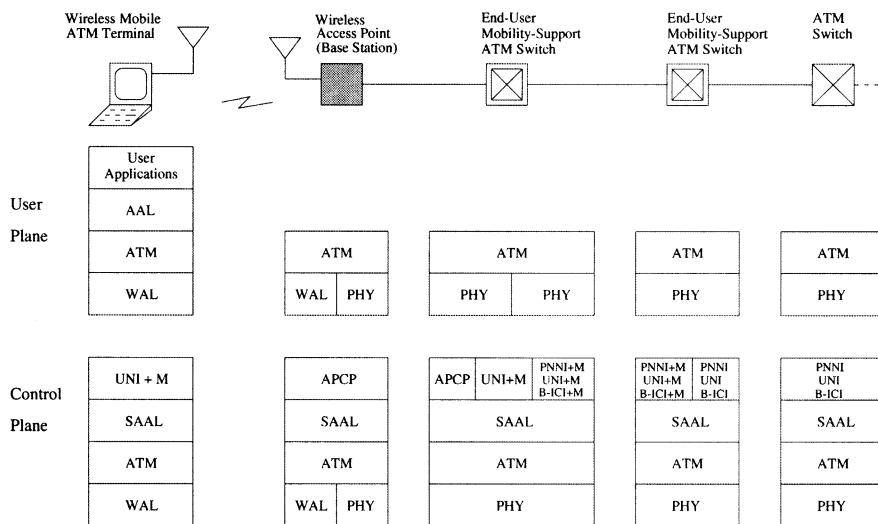


Fig. 12.2 Wireless ATM reference configuration and protocol stacks.

access layer (WAL), which includes wireless physical (PHY), MAC, and LLC layers. The control plane includes the necessary signaling enhancements as well as the WAL. An APCP is used between the AP and the mobility-support ATM switch. This protocol is needed for the communication of information related to the status of the radio resources from the access point to the switch. The mobility-support ATM switch connected to the access point includes not only APCP but also changes to the UNI, B-ICI, and PNNI (UNI + M, B-ICI + M and PNNI + M). InB-ICI + M, UNI + M, and PNNI + M, + M represents supplemental signaling information to support mobility.

12.2 WIRELESS ATM SYSTEMS

12.2.1 NEC's WATMnet Prototype System

The research reported in [1] proposes an ATM cell relay paradigm to be adopted as the basis for the next-generation wireless transport architectures. It is argued that in addition to its regular advantages, the use of ATM switching for intertransmission cell traffic also avoids the crucial problem of developing a new backbone network with sufficient throughput to support intercommunication among the base stations. A protocol layering strategy, harmonized with the standard ATM protocol stack, was proposed. The wireless-specific PHY, MAC, and DLC layers below the ATM layer are added to the standard ATM protocol stack. The baseline ATM network and signaling protocol will have to be augmented to support specific mobility-related functions such as address registration, roaming, handoff, and QoS renegotiation in response to the channel impairment. These imply that the regular ATM network layer and control services such as call setup, VCI/VPI addressing, cell prioritization, and flow control indication will continue to be used for the mobile services.

The research presented in [11] describes an experimental wireless ATM network prototype (WATMnet) system, which was developed at NEC C & C Research Laboratories in Princeton, NJ. The prototype system operates in the 2.4-GHz industrial, scientific, and medical (ISM) band at a channel rate of 8 Mbit/s. Wireless network protocols at portable terminal and base station interfaces support available bit rate (ABR), unspecified bit rate (UBR), variable bit rate (VBR), and constant bitrate (CBR) transport services compatible with ATM, using a dynamic time-division multiple access/time division duplex (TDMA/TDD) MAC protocol for channel sharing and a DLC protocol for error recovery. All network entities, including portable terminals, base stations, and switches, use a mobility-enhanced version of ATM signaling (Q.2931 +) for switched virtual circuit (SVC) connection control functions, including handoffs.

Hardware and software implementation details are given in [12] for the second version of the WATMnet system operating at 25 Mbit/s in the 5 GHz ISM band. Some experimental results in [13, 14, 15] show the validation of major protocol and software aspects, including DLC, wireless control, and mobility signaling for handoffs.

12.2.2 Olivetti's Radio ATM LAN

The work reported in [4] presents the design issues of a wireless ATM LAN that was developed at Cambridge-Olivetti Research Laboratories. All the base stations operate at 10 Mbit/s, using a quaternary phase-shift keying (QPSK) radio at 2.45 GHz (10-MHz bandwidth available for short-range data links in buildings in the United Kingdom). The network uses low-end ATM switches operating at 100 Mbit/s, and has a picocellular structure without frequency reuse. A picocell has a radius of about 10 m.

The Olivetti radio ATM system employs slotted ALOHA with exponential backoff as the MAC layer protocol, which was implemented using Xilinx reprogrammable gate arrays. Packets over the wireless link are one ATM cell long, and cell headers are appropriately altered to accommodate QoS, with the VPI/VCI field compressed. A CRC field of 16 bits is used for error detection, and retransmissions (up to 10) are used for error correction. The next version of this testbed will have a data rate of 25 Mbit/s, operate in the 5-GHz band, and have a range up to 30 m [16].

12.2.3 Virtual Connection Tree

In a paper that deals with topology issues in a cellular radio access network with ATM transport technology, some authors have proposed a structure called the virtual tree architecture [17]. The virtual connection tree totally decentralizes handoff operations, with each mobile responsible for managing its own handoff events without requiring intervention of the admission controller. They showed that this approach can support a very high rate of handoffs between cells. This capability in turn enables very small size cells, therefore very high capacity.

The virtual connection tree consists of ATM switches and cellular base stations connected via some of the ATM switches. The switches are connected via some physical network topology, on top of which is a virtual tree. The root of the tree is a fixed ATM switch node, and the leaves are base stations. When a mobile terminal establishes a connection to the ATM networks, a connection tree is formed from a root (an ATM switch) to a set of base stations to which a mobile terminal may move. Whenever the mobile terminal moves into one of those base stations, it uses one of the preestablished virtual channels from the root node to the corresponding base station.

The goal of a virtual connection tree is to reduce handoff latency by removing connection setup time for the new connection subpath. However,

this method uses preestablished multiple virtual channels, which require more bandwidth even though only one of them is used at a time.

12.2.4 BAHAMA Wireless ATM LAN

A wireless ATM LAN called BAHAMA was proposed at Bell Laboratories [6]. The basic characteristic of the network is its ad hoc nature. The network is self-organizing, that is, a predetermined topology does not exist. The BAHAMA network consists of two types of network elements: portable base stations (PBSs) and mobile endpoints. Each PBS combines ATM switching functionality with wireless access interfaces. The PBSs communicate to determine the network topology after changes due to the addition or deletion of network elements. PBSs are designed with simplicity in mind, and the ATM segmentation and reassembly are carried out in the portable units.

In order to support mobility in the simplest possible way, a new VPI/VCI concept is defined that supports routing based on the destination address. Mobility is supported by means of an adaptive homing algorithm. The advantages of this homing algorithm include preservation of FIFO cell sequence within a connection, and fast handoffs (achieved by eliminating node-by-node connection setup). The network employs a wireless data link layer that provides high reliability, based on both automatic repeat request (ARQ) and forward error correction (FEC). Multiple access is provided by distributed-queuing request update multiple access (DQRUMA) [18].

SWAN is another prototype system proposed at Bell Laboratories [5]. This system is an experimental indoor wireless ATM network based on off-the-shelf 2.4-GHz ISM band radios that operate 625 kbit/s between a base station and a mobile terminal.

12.2.5 NTT's Wireless ATM Access

A system proposed at NTT Wireless System Laboratories called ATM Wireless Access (AWA) concentrates on wireless access to an ATM network [2]. It operates in the superhigh-frequency band (3–30 GHz) for public and private access at data rate between 30 and 80 Mbit/s. Highly directive antennas help alleviate serious shadowing effects that are apparent at these frequencies. The system is designed as a dual wireless ATM LAN with access to an ATM-based public multimedia network. The AWA system is designed for high-speed access with limited terminal mobility and incorporates dynamic reservation TDMA, FEC/ARQ, and QPSK modulation with differential detection.

12.2.6 Other European Projects

A number of European projects are in the process of developing advanced wireless ATM (WATM) demonstrations. Magic WAND (Wireless ATM Network Demonstration) is a joint European project to build a demonstrator

for multimedia information access using a fast WATM network [19]. The WAND demonstrator operates at up to 25 Mbit/s in the 5-GHz frequency band, and the project is also investigating performance issues related to bit rates exceeding 50 Mbit/s in the 17-GHz frequency band. The overall goal is to design a WATM access network demonstration system that can be commercialized and standardized in the European Telecommunications Standards Institute (ETSI). Other European projects include MEDIAN [20] and SAMBA [21].

12.3 RADIO ACCESS LAYERS

A high-speed but low-complexity wireless access technique is critical for providing QoS-based multimedia services to portable terminals. This section outlines wireless-specific protocol layers. These include a radio physical layer, a medium access control (MAC) layer, and a data link control (DLC) layer.¹

12.3.1 Radio Physical Layer

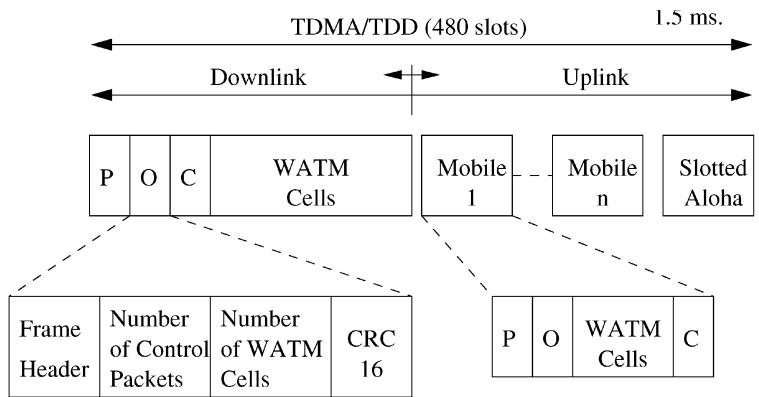
WATM requires a high-speed radio technology capable of providing reasonably reliable transmission and reception in the range of 100–500 m. WATM systems may operate in various frequency bands depending on the regulatory policies. Currently, they are usually associated with the recently allocated 5 GHz band in the US and the HIPERLAN [22] band in Europe. The expected operating frequency range is on the order of 20–25 GHz. Typical target bit rates for the radio physical layer of WATM are around 25 Mbit/s, and a modem must be able to support burst operation with relatively short preambles consistent with transmission of short control packets and ATM cells.

The radio physical layer can be divided into the radio physical medium dependent (RPMD) and the radio transmission convergence (RTC) sub-layers [25]. The RPMD sublayer specifies the lower-level transmission requirements, while the RTC sublayer specifies details of formatted data transmission.

In [12], the RTC sublayer adopts a dynamic TDMA/TDD approach where several virtual circuits are multiplexed in a single radio channel. The TDMA/TDD frame structure is shown in Figure 12.3.

The frame consists of 480 slots every 1.5 ms (25.6 Mbit/s). Each slot is formed by 10 bytes, including 8 data bytes and a 2-byte Reed–Solomon code [RS(10, 8)] for FEC. The frame is divided into an uplink (mobile to base) and a downlink (base to mobile) part. The downlink subframe consists of the

¹In [22] the radio access layers consist of a radio physical layer and a radio DLC layer that contains a MAC sublayer and a logical link control (LLC) sublayer.



P = Preamble

O = Sub-Frame Delineation Overhead

C = Control Region

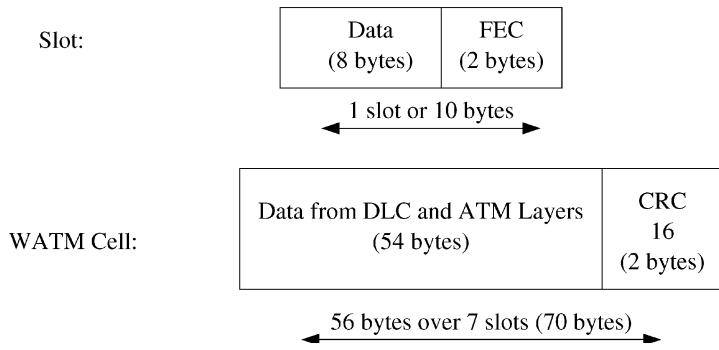


Fig. 12.3 Dynamic TDMA/TDD frame structure.

modem preamble (P), the subframe delineation overhead (O), and the control region (C), followed by WATM cells. The preamble is required for TDMA frame delineation and modem synchronization. The frame header in the downlink subframe delineation overhead (O) consists of a *frame number*, a *radio port identifier*, and *reserved bytes*. The *number of control packets* (C) delineates the control region. Control packets (8-byte) are embedded in the TDMA/TDD slots. They are used by the MAC and DLC layers. WATM cells (56-byte) transport multiplexed downlink user information.

The base station controls the uplink bandwidth allocation for WATM cells from each mobile, taking into account the number and type of active

connections and their bandwidth requirements. Mobiles assemble a subframe similar to that of the base station. The slotted ALOHA region is used by the mobile terminals to send their bandwidth requirements to the base station.

12.3.2 Medium Access Control Layer

For efficient sharing of the available wireless bandwidth among multiple mobile users, a radio MAC layer is required. One of the key issues of WATM is the choice of an appropriate MAC protocol, which has to incorporate the wired ATM capability of integrating a variety of traffic types with different QoS requirements. In Figure 12.3, the slotted ALOHA section includes control packets for bandwidth request. The control packet includes the number of requested slots and the request type that is used to identify the ATM traffic types. For CBR traffic, the fixed number of slots is allocated at connection setup time. VBR traffic is constantly adapted, while ABR traffic bandwidth demand is adjusted based on resource management (RM) cells. UBR traffic is served in a best-effort manner.

The standardization process of a MAC protocol for WATM is ongoing within the Broadband Radio Access Networks (BRAN) project of ETSI. MAC protocols, such as dynamic slot assignment (DSA++) [23] and MAS-CARA [24], are currently under investigation as candidates for the ETSI HIPERLAN Type 2 standard, a developing ETSI standard for WATM. These MAC protocols use a combination of contention-based and contention-free access on the physical channel and are based on scheduled TDMA.

12.3.3 Data Link Control Layer

Link-by-link error control is usually omitted in fixed ATM networks because cell corruption due to channel error is extremely rare for reliable media like copper wire and optical fiber. However, for wireless, bit error rates (BERs) up to 10^{-3} are quite common [26] due to shadowing and other fading effects. To cope with wireless link impairments, a proper error control mechanism is necessary in WATM networks.

A DLC layer is necessary to mitigate the effect of radio channel errors before cells are released to an ATM network layer. Depending on the type of service provided, channel quality, capacity and utilization, the DLC layer may implement a variety of means, including FEC and ARQ.

In [14, 15], a retransmission-based error control scheme is presented. The proposed error control scheme is applied over a wireless link between a mobile terminal and a base station. The wireless data link control is intended to provide a relatively error-free service to the higher layers by recovering corrupted cells. The error control protocol can be flexibly adapted to different service classes with varying QoS. The recovery mode for each individual virtual circuit can be set independently during its setup time. As an example,

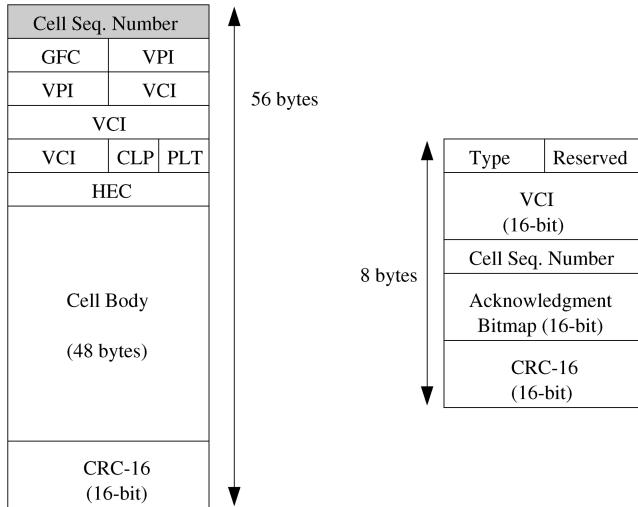


Fig. 12.4 WATM cell format and ACK packet format at DLC layer.

for TCP traffic carried in UBR mode, the error control should be programmed to perform 100% error recovery without worrying about the cell transfer delay over the wireless hop. For CBR voice traffic, on the other hand, the error control attempts to recover a cell only for a specific duration.

Figure 12.4 illustrates WATM cell format and acknowledgment (ACK) packet format at DLC layer that are used for the error recovery scheme. For error detection and selective cell retransmission, a 1-byte cell sequence number (CSN) and a 2 byte CRC are added to the regular 53-byte ATM cell. In the ACK packet format, the 16-bit VCI region specifies the wireless ATM connection for which ACK control packets are generated. The CSN shows the starting control sequence number from which a 16-bit ACK bitmap indicates the reception status of up to 16 transmitted cells.

12.4 HANDOFF IN WIRELESS ATM

In a WATM system, a network must provide seamless services to mobile users. The network has to support uninterrupted services to roaming users even though they change their access points to the ATM networks during the lifetime of a connection. This is done through a *handoff* process [17]. Handoff is the procedure that transfers an ongoing call from one base station (or access point) to another base station as a user moves through the coverage area. During handoff, connection rerouting and cell buffering are performed in the fixed ATM network. When a mobile user crosses a cell boundary, the original connection path between two end points is broken

and part of it is replaced with a new subpath for continuous support of the ongoing call. Buffering is also necessary for in-sequence and loss-free delivery of the ATM cell containing user data in order to guarantee the QoS on the connection.

12.4.1 Connection Rerouting

In a cell-structured WATM system, the radio coverage area of a base station is limited to its cell size. A mobile terminal cannot communicate with a base station if the terminal is beyond its coverage area. To provide seamless service to a mobile user crossing a cell boundary, handoff is needed. When the radio link quality between a mobile terminal and its current base station deteriorates, the mobile terminal has to terminate communication with the base station and communicate with an adjacent base station through a new radio link, which is established by a handoff procedure.

In a fixed ATM network, handoff means connection rerouting during a call. As a result of handoff, the original connection path between two end points is broken, and part of it is replaced with a new subpath for continuous support of the ongoing call.

Figure 12.5 shows a typical handoff procedure in a WATM system. Assume that the mobile terminal communicates with the other end party through BS1. When the mobile terminal moves away from BS1 and goes into the cell of BS2, handoff occurs. That is, the mobile terminal switches its access point from BS1 to BS2 and uses the newly established radio link between the mobile and BS2. In the fixed ATM network of the figure, as a result of handoff, the path between SW3 and BS1 is released, and the path between SW3 and BS2 is added to the original connection path.

Virtual channel (VC) routes need to be continually modified whenever a mobile terminal switches its network access point during the lifetime of a connection.

One possible rerouting scheme is to rebuild a total end-to-end VC whenever handoff occurs. This introduces large handoff latency. Handoff should be done locally without involving the other end party to reduce this latency.²

Many connection rerouting methods for handoff have been proposed in the literature [6, 17, 27, 28, 29, 30, 31, 32]. Most of them rely on partial connection rerouting.

In [17], the virtual connection tree concept was proposed. When a mobile terminal establishes a connection to the ATM networks, a connection tree is formed from a root (an ATM switch) to a set of base stations to which a mobile terminal may move. Whenever the mobile terminal moves into one of

²Handoff should be performed quickly, since it usually introduces temporary interruption of communication. Due to this service interruption, the QoS may degenerate below an acceptable level.

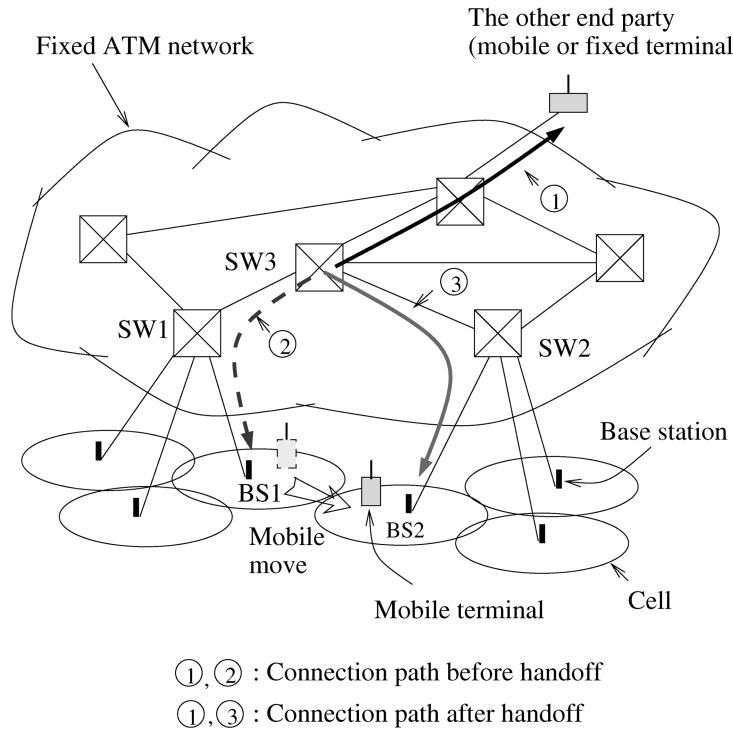


Fig. 12.5 Handoff in a wireless ATM system.

those base stations, it uses one of a set of preestablished virtual channels from the root node to the corresponding base station. The goal of a virtual connection tree is to reduce handoff latency by removing connection setup time for the new connection subpath. However, this method uses preestablished multiple virtual channels, even though only one of them is used at a time. This requires more network resources (VC space, bandwidth, and memory for routing information) than other schemes.

The path (or VC) extension scheme extends an original connection path and adds a new connection subpath from an old base station to a new base station that a mobile terminal is going to visit [28, 29]. This method is simple, and makes it easy to maintain the ATM cell sequence during handoff. However, the extended connection path will increase end-to-end latency, and if base stations and ATM switches have a hierarchical structure as in Figure 12.5, this scheme will produce poor routing efficiency. In addition, a base station needs an ATM switching function to forward ATM cells between base stations.

In the path (or VC) rerouting scheme of [28, 29], an original connection path between the mobile's peer party and an old base station is torn down

and part of it is replaced with a new connection segment between a new base station and an intermediate node on the original connection path during handoff. The intermediate node from which a new connection segment is established is known as a crossover switch (COS) [27].³ (In the handoff example of Figure 12.5, SW3 is used as a COS.) The important issue related to WATM handoff has been the discovery of a COS in the network. The location of a COS is determined by considering the tradeoff between handoff latency and routing efficiency after the handoff.

In a handoff procedure, a COS can be fixed [6, 17] or can be determined dynamically [27, 30, 31]. The algorithms for selection of a COS determine the tradeoff between end-to-end routing efficiency and handoff latency. For an optimal end-to-end route, a complicated algorithm is needed for selection of a COS. This will increase handoff latency [27]. A fast but less optimal scheme is to select a common ancestor node of two handoff-related base stations as a COS [30, 31].

Although all rerouting schemes have their pros and cons as described above, they are based on the following three basic steps:

- Select a crossover switch. (This is omitted in the path extension scheme and the fixed COS scheme.)
- Set up a new subpath between a COS and a new base station. (If the preestablished-multiple-VC scheme is used, this step is unnecessary.)
- Release an old subpath between a COS and an old base station.

12.4.2 Buffering

Unlike in a circuit-switched cellular system, handoff in wireless ATM networks introduces ATM cell loss. To avoid the cell loss, buffering is required during handoff, especially for loss-sensitive traffic.

For uplink traffic from a mobile terminal to the fixed ATM network via a base station, buffering is performed at a mobile terminal until a connection is reestablished. Since we can expect that all ATM cells which left the mobile terminal before the mobile terminal starts to buffer will be sent to a COS during the buffering time, we assume that uplink traffic does not suffer a cell out-of-sequence problem.

For downlink traffic, buffering is required at a COS until the new connection subpath between a COS and a new base station is established by a handoff procedure. Handoff is usually initiated by a mobile terminal. After new base station is chosen, the next step in a handoff procedure is to determine a COS. Once a COS is determined, the switch stops transmitting ATM cells⁴ destined for the mobile terminal and starts buffering them. After

³In the path extension scheme, the old base station can be considered a COS. In the virtual connection tree of [17] the root node becomes a COS.

⁴Since a mobile's handoff request implies degradation of wireless link quality, transmission through the current wireless link has to be avoided as much as possible to reduce cell loss.

the new connection subpath is established, these buffered ATM cells are forwarded to the mobile terminal via the newly established subpath. To guarantee in-sequence cell delivery, these buffered cells should be transmitted before newly arriving cells.

12.4.3 Cell Routing in a COS

As mentioned in Section 12.4.1, a COS is an ATM switch that acts like an anchor in a rerouting procedure, and in which buffering is performed during handoff.

Figure 12.6 illustrates ATM cell flow within a COS. We assume that the original connection path between the mobile terminal and its peer party is routed through input port 1 and output port 1 of the COS. Before handoff occurs, all downlink cells (shaded cells in the figure) departed from the peer party are multiplexed with other ATM cells and arrive at the input port 1. These downlink cells are routed to output port 1 and transmitted to the mobile terminal via a current base station.

When the mobile terminal moves away from the coverage area of the old base station, handoff is initiated. Here we assume that the new base station is routed from output port j of the COS. Once a signaling message for handoff request is sent from a mobile terminal and a new base station is determined, the ATM network will select a COS, If it is identified as a COS, the switch

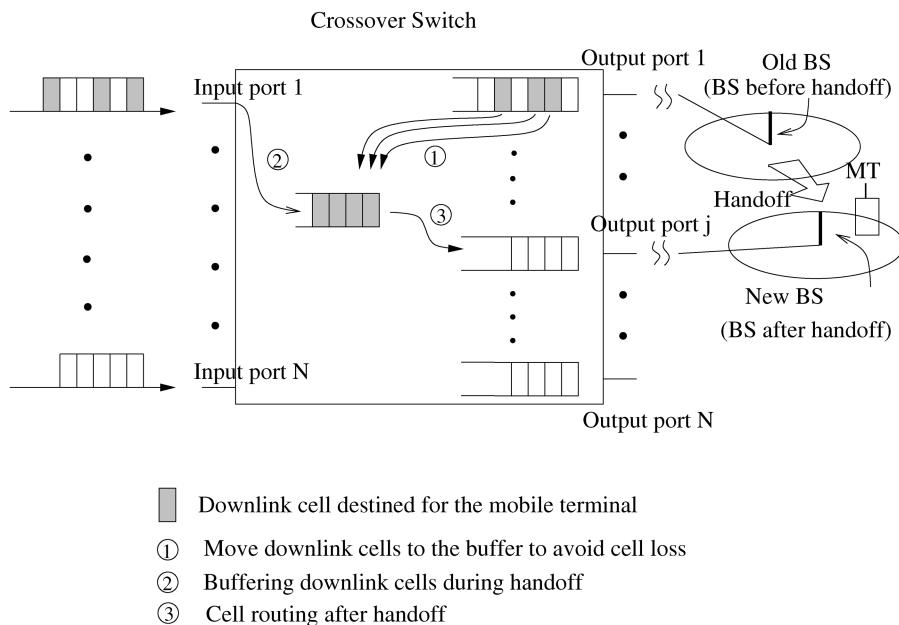


Fig. 12.6 Cell routing within a crossover switch.

will start buffering⁵ the downlink cells until a new connection subpath is established between the COS and the new base station. After the handoff, all downlink cells, including the buffered cells, will be routed to output port j and transmitted to the mobile terminal through the new base station.

In an ATM switch, all cells arriving at the switch are stored in a switch memory. Their output ports are determined based on their input VCI values. Even though a COS starts buffering newly arriving cells destined for a mobile terminal, the cells for the mobile terminal that are already stored in the switch will be transmitted through the old path.

For example, in Figure 12.6, when the COS starts buffering the newly arriving downlink cells, we can notice that some downlink cells are already stored in the output buffer of port 1. These cells will be lost unless the switch prevents them from being transmitted through output port 1, since they will be transmitted to the previous base station after the mobile terminal leaves the base station.

To avoid this cell loss, a COS has to stop transmitting not only newly arriving downlink cells but also the cells that are already stored in the switch memory (the shaded cells in the output buffer of port 1 in Fig. 12.6). If the COS is an output-buffered ATM switch, to avoid cell loss these cells have to be physically moved to other memory, which will be used for buffering cells during handoff. Moving cells from one buffer to the other also increases the end-to-end delay and wastes memory bandwidth. These buffered cells and physically moved cells from the output buffer will be routed to a different output port (output port j in the example of Fig. 12.6) after a new connection setup is completed between the switch and the new base station. In addition, these cells should be transmitted in sequence.⁶ To realize these functions with a regular ATM switch is not straightforward, since normal (nonmobile) ATM traffic does not change its route during the lifetime of a connection.

12.5 MOBILITY-SUPPORT ATM SWITCH

While many researchers have proposed handoff schemes to support mobility in wireless ATM networks, there has been little concern about designing a mobility-support ATM switch architecture. Connection path reformation due to handoff requires ATM cell buffering, and this can cause a cell out-of-sequence problem. To solve this problem, a mobility-support ATM switch needs to perform buffering and cell rerouting efficiently, achieve in-sequence cell transmission, and avoid cell loss during handoff [33].

A COS needs an additional buffer that temporarily stores cells during handoff. It has to stop transmitting all downlink cells (newly arriving cells and

⁵A COS needs an additional buffer for this.

⁶ATM cell order should be guaranteed in ATM networks, even in a handoff situation, to reduce the resequencing burden in higher layers.

cells already stored in the switch memory) destined for a mobile terminal immediately after it is identified as a COS. This can avoid cell loss during handoff. In addition, a COS is required to guarantee in-sequence cell delivery during handoff.

12.5.1 Design of a Mobility-Support Switch

In [33] the authors proposed a mobility-support ATM switch architecture that satisfies all the above requirements. The switch architecture uses a shared memory, which is fully shared by all output ports. Using a shared memory eliminates the need for additional memory for handoff and makes it easy to route cells to the corresponding output port, which can be dynamically changed during the lifetime of a connection due to handoff.

Handoff is based on a VC, and by it the route of the VC is changed. That is, the output port address is changed in the middle of a connection. To efficiently manage each VC, whose output port address can be dynamically changed due to handoff, each connection has its own logical queue (VC queue in Fig. 12.7) in the shared memory. A logical queue is used to link all of the backlogged cells of a connection in the order of their arrival times. Cells are stamped with their arrival times when joining their VC queues, and the timestamps are stored together with the cells.

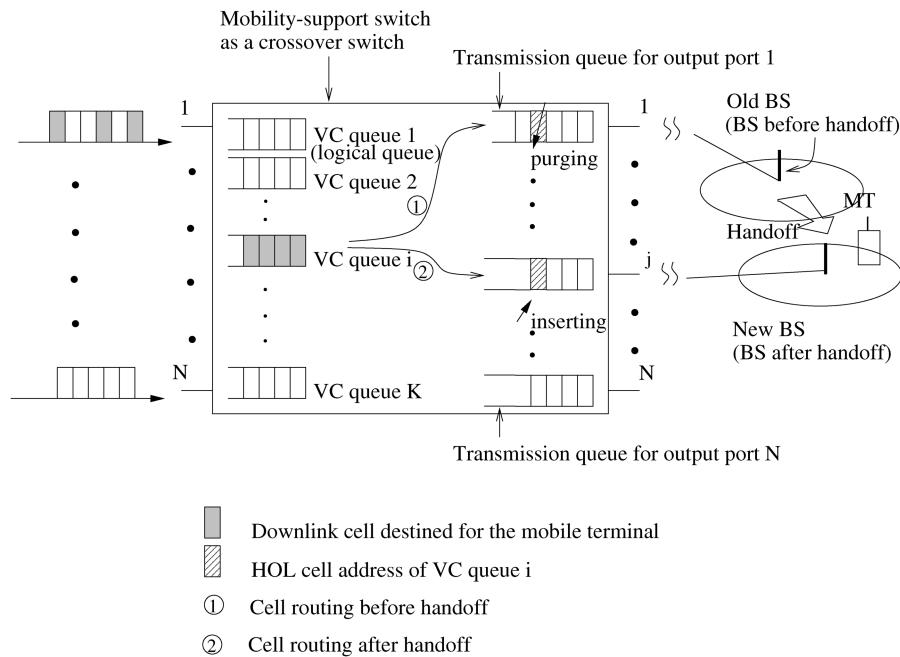


Fig. 12.7 Cell routing in a mobility-support ATM switch.

There will be one transmission queue per output port. Each transmission queue keeps only the addresses of HOL cells⁷ of VCs that are destined for a corresponding output port. When a cell is transmitted, the address of the next cell in the same VC queue is attached to a transmission queue. All cells' addresses are sorted according to their arrival times in a transmission queue.

Figure 12.7 illustrates how to manage VC queues in the shared memory and how to route cells before and after handoff. In this example, it is assumed that a virtual connection between a mobile terminal and its peer party was established through input port 1 and output port 1 of the switch. All downlink cells (shaded cells in the figure) of a virtual connection for the mobile terminal joined the same VC queue. The HOL cell's address of the VC queue is stored in the transmission queue of output port 1 according to its arrival time. When the HOL cell is transmitted, the address of the next cell in the same VC queue ($VC\ queue\ i$ in the figure) is extracted and joins the transmission queue in the order of arrival time.

Now we assume that the mobile terminal moves to a new base station and requests handoff to the ATM network. When a signaling message for handoff arrives at the switch, the HOL cell's address of the corresponding virtual channel is purged from a transmission queue. A newly arriving cell destined for a mobile terminal will still be attached to the corresponding VC queue i , and all cells of the same VC will not be transmitted, since the HOL cell's address has been deleted from the transmission queue. By doing so, all cells destined for the mobile terminal are buffered immediately after the switch received a handoff signaling message, even though some of the cells had been queued in the switch memory before the message arrived. After a new subpath is established, that is, after the new output port address (output port j in the figure) for the VC is determined, the HOL cell's address of the same VC is inserted in a corresponding transmission queue. This will restart cell transmission to the mobile terminal through output port j , which is connected to the new base station.

Storing HOL cells' addresses in transmission queues can prevent cell loss during handoff by purging only a corresponding HOL cell's address from the transmission queues. Otherwise, the cells for the mobile terminal that were queued in the switch memory before a handoff signaling message will be transmitted to the old base station and be lost (since the mobile has already left the base station).

Each transmission queue requires a sorting and purging function for handoff. This function can be realized by an application specific integrated circuits (ASIC), improved from the existing sequencer chip [34].

Figure 12.8 depicts the mobility-support switch architecture with a shared memory. It consists of a cell multiplexer, a demultiplexer, sequencer chips (one for each output port), a mobility controller, an idle-address FIFO, a cell

⁷The HOL cell of a backlogged connection is defined as the oldest backlogged cell of the connection.

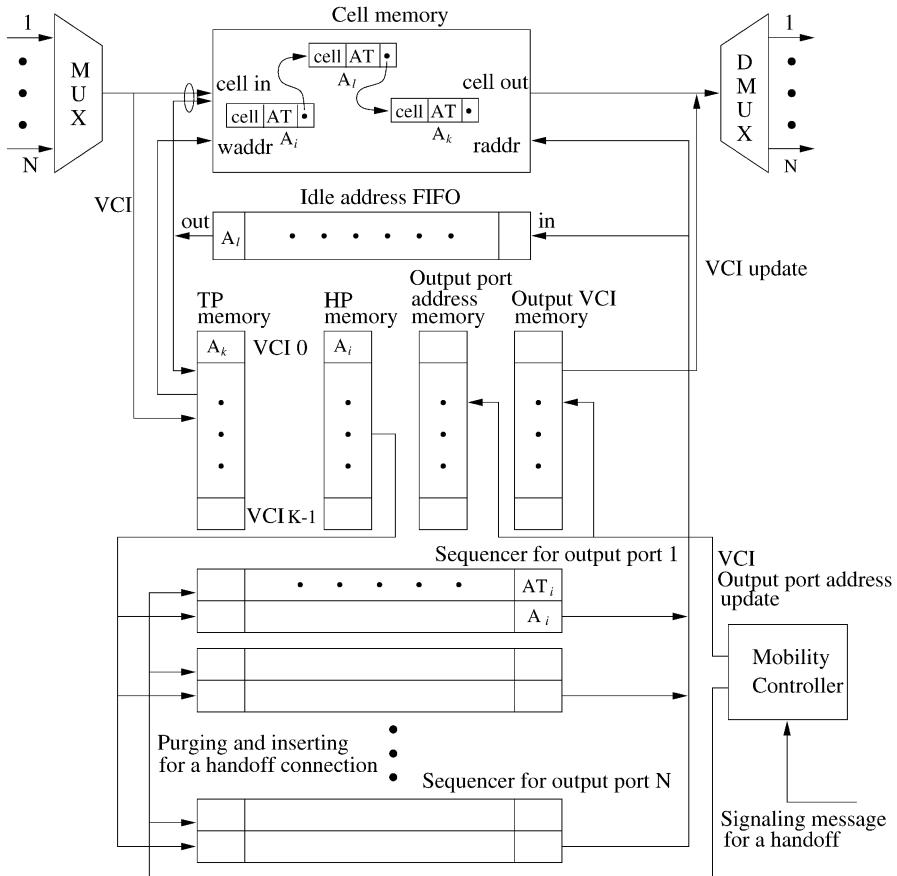


Fig. 12.8 Mobility-support switch architecture with a shared memory.

memory, and four other small memories⁸ (HP memory, TP memory, output VCI memory, and output port address memory). The VLSI chip called the sequencer chip replaces a transmission queue in Figure 12.7 and is used for transmission.

Each module of the sequencer contains two types of information associated with each cell. One is the arrival time, AT, of a cell, with which each cell arriving at the switch is time-stamped. The other is the cell's physical address, A . Each sequencer sorts the (AT, A) pairs associated with HOL cells of VC queues destined for the corresponding output port. Each sequencer arranges the (AT, A) pair, based on the cell's arrival time AT, in such a way that smaller AT appears closer to the head of the sequencer. In other words, cells with smaller ATs are scheduled to depart earlier.

⁸The number of entries of these memories is the same as the total number of VCIs.

Cells can also be timestamped using a weighted RR or weighted fair queueing algorithm according to the allocation bandwidth to the virtual connections. Thus, it may guarantee the delay bound of each connection (using weighted fair queueing), and it only needs to calculate the timestamps when the HOL cells join the transmission queues, instead of assigning time-stamps when they arrive [35].

All arriving cells from N inputs are first time-division multiplexed and written one by one into the cell memory, each with its timestamp at empty cell locations with the addresses obtained from the idle-address FIFO. The FIFO contains the addresses of current vacant cell locations in the cell memory. Every cell that belongs to the same virtual connection is attached to a logical queue with a forward pointer (FP) to point to the next linked cell in the same logical queue. Thus, all of the backlogged cells of a connection are linked in a logical queue in the order of the cells' arrival times.

Each logical queue is confined by two pointers, the head pointer (HP) and tail pointer (TP), which are stored in the HP memory and the TP memory, respectively. The former, which is also stored in a sequencer according to its output port, points to the first cell of a logical queue. The latter, which is stored in the TP memory according to its VCI, points to the last cell. This constructs a linked list and acts as a logical queue for a connection. Note that the HOL cell's address is stored in both the transmission queue and the HP memory. Therefore, when the HOL cell's address is purged from the transmission queue due to handoff, it is still kept in the HP memory.

Figure 12.9 shows an example of a logical queue where a cell is leaving or arriving. When a cell arrives, it is stored in the cell memory, and the TP of the associated connection is checked. The TP value stored in the TP memory can be accessed through the cell's VCI. If the TP is null, the newly arriving cell is a HOL cell, and its (AT, A) pair is inserted in a sequencer according to its output port address, which can be obtained from the output port address memory through the cell's VCI. If the TP is not null, the cell pointed to by the TP [A_k in Fig. 12.9(a)] is accessed, and its FP is changed from null to the new cell's address [A_l in Fig. 12.9(b)]. After that, the TP is replaced by the new cell's address (A_l). When a cell is transmitted, the cell's (AT, A) pair is evacuated from its sequencer. If it has its successor, that is, if the cell's FP is not null, a cell to which the FP points [A_j in Fig. 12.9(b)] becomes a HOL cell. The new (AT_j, A_j) pair of the HOL cell is inserted in the sequencer. If the departing cell's FP is null, that is, if the cell is the last cell in a logical queue, the TP of the corresponding entry in the TP memory is updated with a null address. When cells destined for each output port depart from the switch, their addresses are obtained from the rightmost modules of each sequencer, and they are read out from a cell memory one by one for demultiplexing. As soon as each cell is transmitted, its (AT, A) pair is purged from the corresponding sequencer and all other (AT, A) pairs in the sequencer are shifted to the right by one position. The freed-up locations in the cell memory can be used by future cells, and their addresses are returned to the idle-address FIFO.

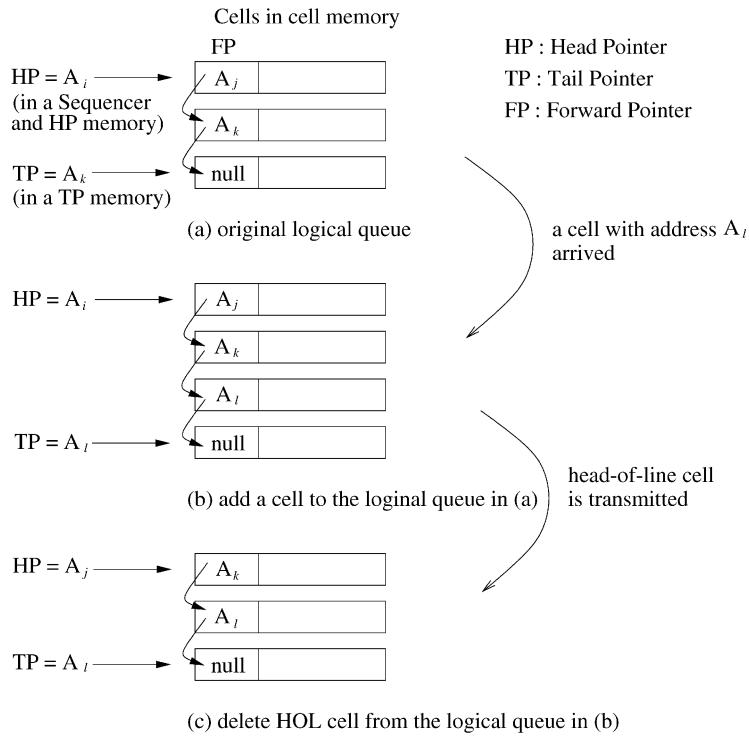


Fig. 12.9 Adding and deleting a cell in a logical queue.

Unlike in a normal ATM switch architecture, in the mobility-support switch architecture a cell's VPI/VCI update is performed when the cell departs from the switch, since the VPI/VCI and output port address can be changed due to handoff while the cell is in the switch memory.

When a signaling message for handoff arrives, the switch stops transmitting ATM cells to an associated output port. This is done through the following operation. When the mobility controller receives the signaling message, it checks the HP memory with the associated VCI. If the HP is null, that means the logical queue of the virtual connection is empty. Then, the corresponding output port address is updated with a null address. When a new cell of the same connection arrives, its (AT, A) pair is not inserted in a sequencer, since the associated output port address is updated with a null address and this results in stopping transmission of ATM cells for the handoff connection. If the HP is not null, the address exists in the corresponding sequencer for transmission. This information should be removed from the sequencer to stop transmission. The way to purge the associated (AT, A) pair in a sequencer is to broadcast the cell's address to all modules. If the local address matches the broadcast address, its (AT, A) pair will be purged from the Sequencer.

When a signaling message for the completion of the connection setup between the switch and the new base station which a mobile terminal is currently visiting arrives, the handoff controller updates the output port address in the output port address memory and the output VCI in the output VCI memory for the handoff connection. Since the signaling message has information about a new connection subpath for a mobile's handoff, a COS can resume transmitting ATM cells of a handoff connection with a new allocated output port address and VCI. To resume transmission, the mobility controller accesses the HP memory through the old VCI to find the HOL cell of the handoff connection. If the HP is not null, its (AT, A) pair is inserted in a sequencer that is associated with the new output port address. This can resume transmitting cells of the handoff connection through the new output port.

12.5.2 Performance

In this subsection we present the performance of the mobility-support switch architecture described in the previous section.

For performance study, it is assumed that cells from different VCs are multiplexed in bursts, and that the bursts are interleaved as they arrive at each input port of the switch. To quantify the traffic characteristics, the following traffic model is considered in which an arrival process to an input port alternates between on (active) and off (idle) periods.

Figure 12.10 shows the traffic model used for simulations. During the on period, cells that originate from the same VC arrive at an input port continuously in consecutive cell time slots, and during the off period no cells are generated. Both the on and off periods are assumed to be geometrically distributed with average lengths $E[B]$ and $E[I]$, respectively.

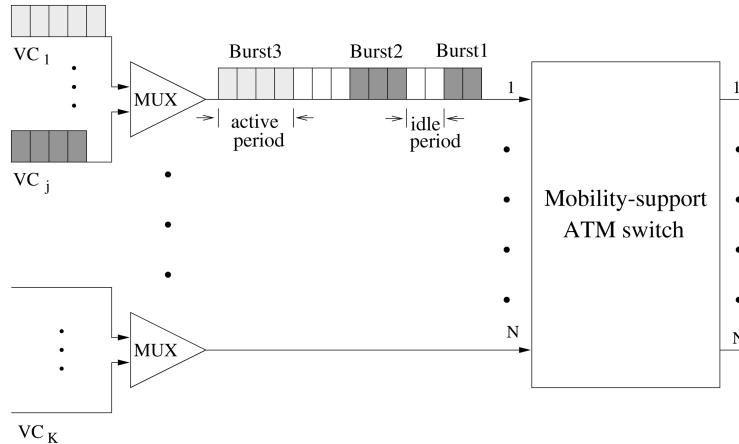


Fig. 12.10 The simulated traffic model.

Let us define p as the probability that an arriving cell is the last cell in a burst, and q as the probability that a new burst starts per time slot. Then the probability that the burst has i cells is

$$P[B = i] = (1 - p)^{i-1} p, \quad i \geq 1, \quad (12.1)$$

and the probability that an idle period will last for j time slots is

$$P[I = j] = (1 - q)^j q, \quad j \geq 0, \quad (12.2)$$

where we assume that there is at least one cell in each burst and the duration of an idle period can be zero. The average lengths $E[B]$ and $E[I]$ are given by

$$E[B] = \frac{1}{p} \quad \text{and} \quad E[I] = \frac{1 - q}{q}. \quad (12.3)$$

Given p and q , the offered load ρ is equal to

$$\frac{E[B]}{E[B] + E[I]}. \quad (12.4)$$

In this simulation study, it is assumed that the switch size N is 16 and the total number of VCs is 512 (32 VCs per input port). The initial route of each VC is shown in Table 12.1. Then the offered load (ρ_i) of each input port i

TABLE 12.1 Initial Routing Table

VC	Input Port Address	Output Port Address
1	1	1
2	1	2
⋮	⋮	⋮
16	1	16
17	1	1
18	1	2
⋮	⋮	⋮
32	1	16
33	2	1
34	2	2
⋮	⋮	⋮
512	16	16

and the offered load (ρ_i) of each output port j are given by

$$\rho_i = \sum_{k=1}^{32} \rho_{VC_{(i-1) \times 32+k}}, \quad i = 1, 2, \dots, 16, \quad (12.5)$$

and

$$\rho_j = \sum_{k=0}^{16} \rho_{VC_{j+16 \times k}}, \quad j = 1, 2, \dots, 16, \quad (12.6)$$

where $\rho_{VC_{(i-1) \times 32+k}}$ and $\rho_{VC_{j+16 \times k}}$ are the average load of $VC_{(i-1) \times 32+k}$ and the average load of $VC_{j+16 \times k}$, respectively.

For simplicity, a uniform source distribution is considered, in which any burst to each input port has an equal probability of being originated from any VC, and successive choices of VC are independent. Then the average load of each VC is the same and can be expressed as

$$\rho_{VC_k} = \rho_{VC}, \quad k = 1, 2, \dots, 512 \quad (12.7)$$

and ρ_i in (12.5) and ρ_j in (12.6) become

$$\rho = \rho_i = \rho_j = 32\rho_{VC}. \quad (12.8)$$

Although it was assumed that the traffic loads to each output port are uniformly distributed, the uniform distribution is no longer valid when handoff is considered. Since handoff changes the route of the corresponding VC (e.g., the output port address of the VC), the traffic load distribution to each output port is dynamically changed depending on the handoff rate. It is noted that this will affect the delay and cell loss performance of a mobility-support switch.

Figure 12.11 shows the average cell delay as a function of input offered load (ρ_i) in different situations. Here N and M are the switch size and the total number of VCs, respectively. BL means the average burst length, which is given in (12.3), and HR is the handoff rate, which is defined as the average number of handoffs that occur in a switch during a second. It is assumed that all VCs are mobile connections, and each VC has an equal probability of handoff. It is also assumed that each handoff is independent. As mentioned in Section 12.4, handoff requires buffering until a new connection subpath is established between a COS and the new base station that a mobile terminal is roaming to. It is assumed that this buffering time is 10 ms, allowing for the connection setup for several hops [36]. Thus, when handoff for a VC occurs, the cells of the VC are queued in the switch memory and transmitted through a different output port after 10 ms.

In Figure 12.11, the case of 100/s handoff rate is compared with the case of no handoff. The 100/s handoff rate means that the handoff of each VC

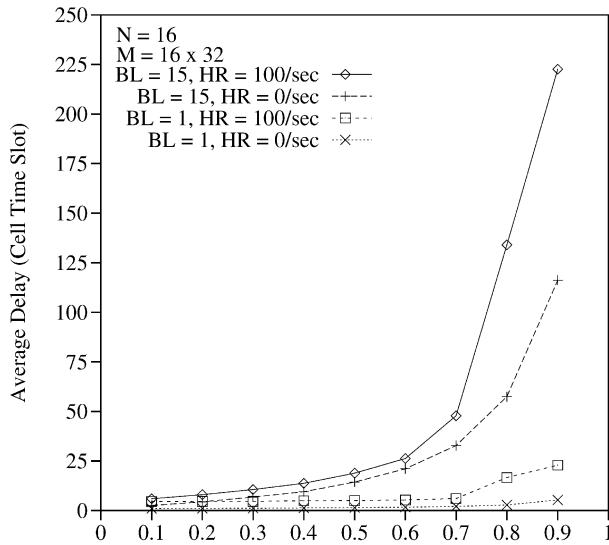


Fig. 12.11 Average delay as a function of offered load.

occurs in every 5.12 s on the average, and the rate is sufficiently high to see the handoff effect of the worst case. As shown in Figure 12.11, the average cell delay of the handoff case is larger than that of the no-handoff case. This is so with both bursty ($BL = 15$) and nonbursty ($BL = 1$) traffic and is prominent with large input offered load (ρ_i). For example, for an input offered load ρ_i of 0.9 and burst length BL of 15, the handoff case's average delay (222.7 cell times) is almost twice as large as that in the no-handoff case (116.2 cell times).

There are two factors that increase the average delay in the handoff case. One is that due to handoffs of each VC, the offered loads to each output port are unbalanced, while the load distribution of each output port is assumed to be uniform for the no-handoff case. The other is that when handoff occurs, no incoming cells of the corresponding VC are served during the connection setup time of a new subpath.

Figure 12.12 shows simulation results on cell loss probability vs. buffer size. In this experiment, the switch size, the number total of VCs, and the offered load to each input port are 16, 512, and 0.9, respectively. For uniform random traffic ($BL = 1$), the required buffer size is much smaller than that for bursty traffic ($BL = 15$), and the required buffer size in the handoff case is larger than that in the no-handoff case. This shows that a COS needs a large memory for buffering cells of each handoff VC, although it uses a shared memory. For example, for bursty traffic the required buffer size in the handoff case is about 7000 cells to maintain the cell loss probability at less than 10^{-6} . This is at least twice as large as in the no-handoff case (about 3300).

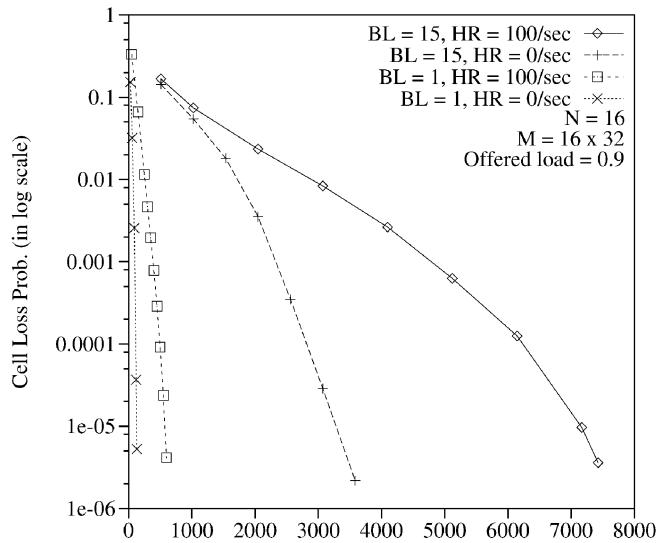


Fig. 12.12 Cell loss probability as a function of buffer size.

Simulation results show the impact of handoffs on the performance of the mobility-support switch architecture. It can be noticed that the average cell delay in the handoff case is almost two times larger than that in the no-handoff case when the traffic load is heavy and bursty and the handoff rate is high. It can be also noticed that the required memory size in the handoff case is double that in the no-handoff case for heavily loaded bursty traffic and high handoff rate.

REFERENCES

1. D. Raychaudhuri and N. D. Wilson, "ATM-based transport architecture for multiservices wireless personal communication networks," *IEEE J. Select. Areas Commun.*, vol. 12, no. 8, pp. 1401–1414, Oct. 1994.
2. M. Umehira, M. Nakura, H. Sato, and A. Hashimoto, "ATM wireless access for mobile multimedia: concept and architecture," *IEEE Personal Commun.*, pp. 39–48, Oct. 1996.
3. D. Raychaudhuri, "Wireless ATM networks: architecture, system design and prototyping," *IEEE Personal Commun.*, pp. 42–49, Aug. 1996.
4. J. Porter and A. Hopper, "An overview of the ORL wireless ATM system," *Proc. IEEE ATM Workshop*, Sept. 1995.
5. E. Hyden, J. Trotter, P. Krzyzanowski, M. Srivastava, and P. Agarwal, "SWAN: an indoor wireless ATM network," *Proc. ICUPC '95*, Tokyo, Japan, pp. 853–857, Nov. 1995.

6. M. J. Karol, K. Y. Eng, and M. Veeraraghavan, "BAHAMA: a broadband ad-hoc wireless ATM local area network," *Proc. IEEE ICC '95*, pp. 1216–1223, June 1995.
7. D. Raychaudhuri, L. Dellaverson, M. Umehira, J. Mikkonen, T. Phipps, C. Lind, and H. Suzuki, "Charter, scope and work plan for proposed wireless ATM working group," *ATM Forum/96-0530/PLEN*, Apr. 1996.
8. R. R. Bhat and K. Rauhala, "Draft baseline text for wireless ATM capability set 1 specification," *ATM Forum/BTD-WATM-01.10*, Dec. 1998.
9. R. R. Bhat and R. Gupta, "Requirements document for wireless ATM," *ATM Forum/98-0395/WATM*, Jul. 1998.
10. G. Bautz, H. Mitts, J. Mikkonen, M. Niemi, and K. Rauhala, "A proposal for access point control protocol," *ATM Forum/97-0507/WATM*, July 1997.
11. D. Raychaudhuri, L. J. French, R. J. Siracusa, S. K. Biswas, R. Yuan, P. Narasimhan, and C. A. Johnston, "WATMnet: a prototype wireless ATM system for multimedia personal communication," *IEEE J. Select. Areas Commun.*, vol. 15, no. 1, pp. 83–95, Jan. 1997.
12. C. A. Johnston, P. Narasimhan, J. Kokudo, and M. Ohki, "Architecture and implementation of radio access protocols in wireless ATM networks," *IEEE ICC '98*, Atlanta, GA, Jun. 1998.
13. R. Yuan, S. Biswas, and D. Raychaudhuri, "A signaling and control architecture for mobility support in wireless ATM networks," *Proc. IEEE ICC '96*, pp. 478–484, Dallas, 1996.
14. P. Narasimhan, S. K. Biswas, C. A. Johnston, R. J. Siracusa, and H. Kim, "Design and performance of radio access protocols in WATMnet, a prototype wireless ATM network," *Proc. ICUPC '97*, San Diego, CA, Oct. 1997.
15. S. K. Biswas, H. Kim, P. Narasimhan, R. J. Siracusa, and C. A. Johnston, "Design and implementation of data link control protocol for CBR traffic in wireless ATM networks," *Proc. ICUPC '98*, Florence, Italy, Oct. 1998.
16. ORL Radio ATM project. <http://www.orl.co.uk/radio/>
17. A. Acampora and M. Naghshineh, "An architecture and methodology for mobile-executed handoff in cellular ATM networks," *IEEE J. Select. Areas Commun.*, vol. 12, no. 8, pp. 1365–1375, Oct. 1994.
18. M. J. Karol, Z. Liu, and K. Y. Eng, "Distributed-queuing request update multiple access (DQRUMA) for wireless packet (ATM) networks," *Proc. IEEE ICC '95*, pp. 1224–1231, Jun. 1995.
19. The Magic WAND wireless ATM demonstrator. <http://www.tik.ee.ethz.ch/~wand>
20. The MEDIAN project. <http://www.imst.de/median/median.html>
21. The SAMBA project. <http://hostria.cet.pt/samba/index.html>
22. European Telecommunications Standards Institute (ETSI) project on Broadband Radio Access Network (BRAN). <http://www.etsi.org/bran>
23. D. Petras et al., "Wireless ATM: performance evaluation of a DSA++MAC protocol with fast collision resolution by a probing algorithm," *Int. J. Wireless Inf. Networks*, vol. 4, no. 4, 1997.
24. F. Bauchot et al., "MASCARA: a MAC protocol for wireless ATM," *ACTS Mobile Summit*, Granada, Spain, Nov. 1996.

25. P. Narasimhan et al., "Proposal for radio transport convergence, medium access control, and logical link control layers for wireless ATM," *ETSI BRAN #4*, London, Sep. 1997.
26. H. Hashemi, "The indoor radio propagation channel," *Proc. IEEE*, vol. 81, no. 7, pp. 943–968, Jul. 1993.
27. C. -K. Toh, "Crossover switch discovery for wireless ATM LANs," *ACM J. Mobile Networks Appl.*, vol. 1, no. 2, pp. 145–165, Oct. 1996.
28. B. Rajagopalan, "Mobility management in integrated wireless-ATM networks," *Proc. 1st Int. Conf. on Mobile Computing and Networking*, Berkeley, CA, Nov. 1995.
29. A. Acharya, J. Li, B. Rajagopalan, and D. Raychaudhuri, "Mobility management in wireless ATM networks," *IEEE Commun. Mag.*, vol. 35, no. 11, pp. 100–109, Nov. 1997.
30. J. H. Condon et al., "Rednet: a wireless ATM local area network using infrared links," *Proc. of 1st Int. Conf. on Mobile Computing and Networking*, pp. 151–159, Berkeley, CA, Nov. 1995.
31. B. A. Akyol and D. C. Cox, "Rerouting for handoff in a wireless ATM network," *IEEE Personal Commun.*, pp. 26–33, Oct. 1996.
32. C.-K. Toh and B. Akyol, "A survey of handover techniques in wireless ATM networks," *Int. Wireless Inf. Networks J.*, vol. 5, no. 1, 1998.
33. H. Kim and H. J. Chao, "Design of a mobility-support ATM switch," *Proc. Globecom'98*, Sydney, Australia, Nov. 1998.
34. H. J. Chao and N. Uzun, "A VLSI sequencer chip for ATM traffic shaper and queue manager," *IEEE J. Solid-State Circuits*, vol. 9, no. 7, pp. 1634–1643, Nov. 1992.
35. H. J. Chao, H. Cheng, Y. R. Jeng, and D. Jeong, "Design of a generalized priority queue manager for ATM switches," *IEEE J. Select. Areas Commun.*, vol. 15, no. 5, pp. 867–880, Jun. 1997.
36. S. K. Biswas, "Handling realtime traffic in mobile networks," Ph.D. Thesis, University of Cambridge Computer Laboratory, Sep. 1994.

CHAPTER 13

IP ROUTE LOOKUPS

The emergence of new multimedia networking applications and the growth of the Internet due to the exponentially increasing number of users, hosts, and domains have led to a situation where network capacity is becoming a scarce resource. In order to maintain the good service, three key factors have been considered in designing IP networks: large-bandwidth links, high router data throughput, and high packet forwarding rates [16]. With the advent of fiber optics, which provides fast links, and the current switching technology, which is applied for moving packets from the input interface of a router to the corresponding output interface at gigabit speeds, the first two factors can be readily handled. Therefore, the key to the success of the next generation IP networks is the deployment of high-performance routers to forward the packets at the high speeds.

There are many tasks to be performed in packet forwarding; packet header encapsulation and decapsulation, updating a time-to-live (TTL) field in each packet header, classifying the packets into queues for specific service classes, etc. The major task, which seems to dominate the processing time of the incoming packet, is searching for the next-hop information of the appropriate prefix matching the destination address from a routing table. This is also called an *IP route lookup*.

As the Internet has evolved and grown over in recent years, it has been proved that the IP address space, which is divided into classes A, B, and C, is inflexible and wasteful. Class C, with a maximum of 254 host addresses, is too small, while class B, which allows up to 65,534 addresses, is too large for most organizations. The lack of a network class of a size that is appropriate for mid-sized organization results in exhaustion of the class B network address

space. In order to use the address space efficiently, bundles of class C addresses were given out instead of class B addresses. This also causes massive growth of routing table entries.

To reduce the number of routing table entries, *classless interdomain routing* (CIDR) [4] was introduced to allow for arbitrary aggregation of networks. A network that has identical routing information for all subnets except a single one requires only two entries in the routing table: one for the specific subnet (which has preference if it is matched) and the other for the remaining subnets. This decreases the size of the routing table and results in better usage of the available address space. On the other hand, an efficient mechanism to do IP route lookups is required.

In the CIDR approach, a routing table consists of a set of IP routes. Each IP route is represented by a \langle route prefix/prefix length \rangle pair. The prefix length indicates the number of significant bits in the route prefix. Searching is done in a longest-matching manner. For example, a routing table may have the prefix routes $\langle 12.0.54.8/32 \rangle$, $\langle 12.0.54.0/24 \rangle$, and $\langle 12.0.0.0/16 \rangle$. If a packet has the destination address $\langle 12.0.54.2 \rangle$, the second prefix route is matched, and its next hop is retrieved and used for forwarding the packet.

This chapter is organized as follows: Starting from Section 13.1, the architectures of generic routers are described and the design criteria that should be considered for IP lookups is discussed. Recently, there have been a number of techniques proposed to provide fast IP lookups [2, 3, 5, 6, 9, 12, 14, 15, 16]. From Section 13.2 to Section 13.9 we will look at several IP route-lookup schemes proposed over the past few years.

13.1 IP ROUTER DESIGN

13.1.1 Architectures of Generic Routers

The architectures of generic routers can be broadly classified into two categories. One is schematically shown in Figure 13.1. A number of *network interfaces*, *forwarding engines*, and a *network processor* are interconnected with a *switching fabric*. Inbound interfaces send packet headers to the forwarding engines through the switching fabric. The forwarding engines in turn determine which outgoing interface the packet should be sent to. This information is sent back to the inbound interface, which forwards the packet to the outbound interface. The only task of a forwarding engine is to process packet headers. All other tasks—such as participating in routing protocols, resource reservation, handling packets that need extra attention, and other administrative duties—are handled by the network processor. The BBN multigigabit router [13] is an example of this design.

Another router architecture is shown in Figure 13.2. Here, processing elements in the inbound interface decide to which outbound interface

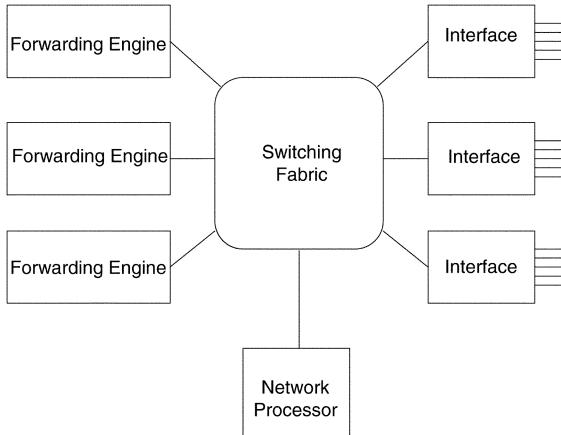


Fig. 13.1 Router design with forwarding engines.

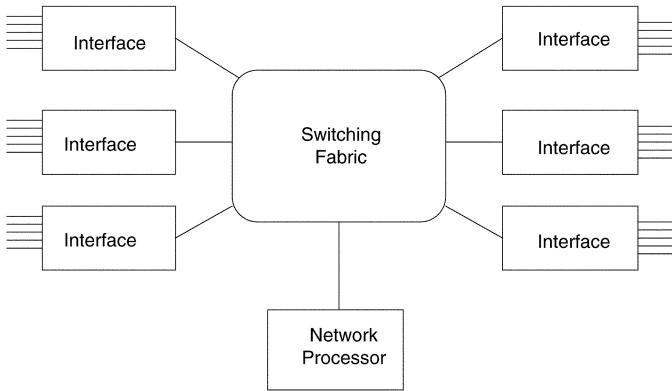


Fig. 13.2 Router design with processing power on interfaces.

packets should be sent. The GRF routers from Ascend Communications, for instance, use this design.

The forwarding engines in Figure 13.1 and the processing elements in Figure 13.2 use a local version of the routing table—a *forwarding table*, downloaded from the network processor—to make their routing decisions. It is not necessary to download a new forwarding table for each routing update. Routing updates can be frequent, but since routing protocols need time on the order of minutes to converge, forwarding tables can be allowed to grow a little stale and need not change more than once per second.

The network processor needs a dynamic routing table designed for fast updates and fast generation of forwarding tables. The forwarding tables, on the other hand, can be optimized for lookup speed and need not be dynamic.

13.1.2 IP Route Lookup Design

When designing the data structure used in the forwarding table, the primary goal is to minimize lookup time. To reach that goal, two parameters should be simultaneously minimized:

- the number of memory accesses required during lookups, and
- the size of the data structure.

Reducing the number of memory accesses required during a lookup is important because memory accesses are relatively slow and usually the bottleneck of lookup procedures. If the data structure can be made small enough, it can fit entirely in the cache of a conventional microprocessor. This means that memory accesses will be orders of magnitude faster than if the data structure needs to reside in memory consisting of the relatively slow DRAM.

If the forwarding table does not fit entirely in the cache, it is still beneficial if a large fraction of the table can reside in cache. Locality in traffic patterns will keep the most frequently used pieces of the data structure in cache, so that most lookups will be fast. Moreover, it becomes feasible to use fast SRAM for the small amount of needed external memory. SRAM is expensive, and the faster it is, the more expensive it is. For a given cost, the SRAM can be faster if less is needed.

As secondary design goals, the data structure should

- need few instructions during lookup, and
- keep the entities naturally aligned as much as possible to avoid expensive instructions and cumbersome bit-extraction operations.

These goals have a second-order effect on the performance of the data structure.

For the purpose of understanding the data structure, imagine a binary tree that spans the entire IP address space (Fig. 13.3). Its height is 32, and the number of its leaves is 2^{32} , one for each possible IP address. The prefix of a routing table entry defines a path in the tree ending in some node. All IP

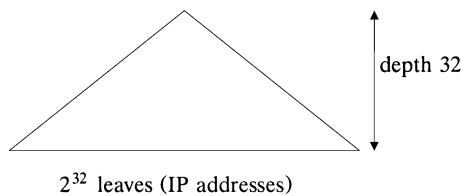


Fig. 13.3 Binary tree spanning the entire IP address space.

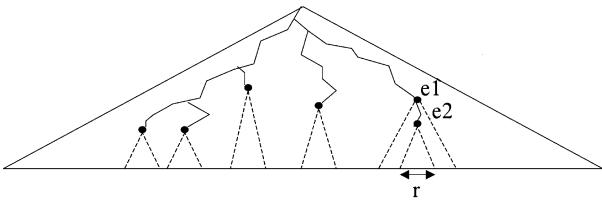


Fig. 13.4 Routing entries defining ranges of IP addresses.

addresses (leaves) in the subtree rooted at that node should be routed according to that routing entry. In this manner each routing table entry defines a range of IP addresses with identical routing information (next-hop IP address).

If several routing entries cover the same IP address, the rule of the longest match is applied; it states that for a given IP address, the routing entry with the longest matching prefix should be used. This situation is illustrated in Figure 13.4; the routing entry e1 is hidden by e2 for addresses in the range r .

13.2 IP ROUTE LOOKUP BASED ON CACHING TECHNIQUE

One possible IP route-lookup approach uses a caching technique where the routing entries of the most recently used destination addresses are kept in a cache. The technique relies on there being enough locality in the traffic so that the cache hit rate is sufficiently high and the cost of a routing lookup is amortized over several packets. These caching methods have worked well in the past. However, as the current rapid growth of the Internet increases, the required size of address caches and hardware caches might become uneconomical.

13.3 IP ROUTE LOOKUP BASED ON STANDARD TRIE STRUCTURE

A *trie structure* is a multiway tree in which each node contains zero or more pointers to its child nodes. In the 1-bit trie structure [8], each node contains two pointers, the 0-pointer and the 1-pointer. A node X at level h represents the set of all route prefixes that have the same h bits as their first bits. Depending on the value of the $(h + 1)$ th bit, 0 or 1, each pointer of the node X points to the corresponding subtree (if it exists), which represents the set of all route prefixes that have the same $(h + 1)$ th bits as their first bits. Each IP lookup starts at the root node of the trie. Based on the value of each bit of the destination address of the packet, the lookup algorithm determines the next node to be visited. The next hop of the longer matching prefix found along the path is maintained while the trie is traversed.

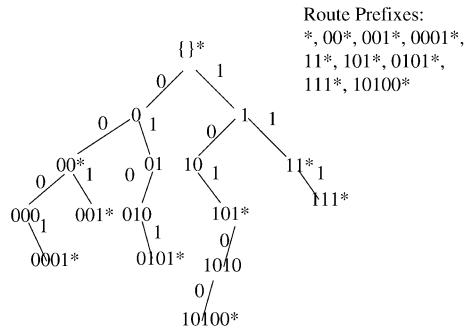


Fig. 13.5 A 1-bit trie structure for IP route lookups.

An example is shown in Figure 13.5. Each node has a flag associated with it, which indicates whether the particular route prefix terminated there is in the routing table. The flag is indicated with an asterisk. Assume that the destination address 10101101 is given. The IP lookup starts at the top and traverses the path indicated by the destination address, remembering the last time a flag (an asterisk) was seen. Starting at the top, the zero-length prefix has a flag. The first bit of 10101101 is 1, so we go to the right and get to the node with the name 1. There is no asterisk, so {} is still the longest prefix matched so far. The second and third bits of 10101101 are 0 and 1, so we go to the left and right to the node 101*. Here an asterisk is found, so the longest matching prefix is updated from {} to 101. The next bit is 0. We go to the node 10101. The next bit is 1, but there is no pointer marked 1 out of the node 1010. Thus, the lookup is stopped here, and the longest matching prefix is 101.

Maintaining the trie structure is not difficult. To add a route prefix, say 1011, simply follow the pointers to where 1011 would be in the tree. If no pointers exist for that prefix, they should be added. If the node for the prefix already exists, it needs to be marked as being in the forwarding table, that is, an asterisk must be added. To delete a route prefix that has no children, the node and the pointer pointing to it are deleted and the parent node is examined. If it has another child or is marked with an asterisk, it is left alone. Otherwise, that node is also deleted and its parent node is examined. The deletion process is repeated up the tree until a node that has another child or is marked is found.

Using the 1-bit trie structure for the IP route lookups has the drawback in that the number of memory accesses in the worst case could be 32 for IPv4.

The performance of lookups can be substantially improved by using a multibit trie structure. This more general idea was suggested by Srinivasan and Varghese [14]. In the multibit, say K -bit, trie structure, each node contains 2^K pointers. Each route prefix in the forwarding table is expanded to include new prefixes whose lengths are the smallest numbers that are greater than or equal to the original length and also multiples of K bits. For

TABLE 13.1 An Example of Expanding the Route Prefixes by Using 2-Bit Trie Structure

Forwarding Table	
Original	Expanded
P1 = 0*	01* (P1)
P2 = 1*	00* (P1)
P3 = 10*	10* (P3)
P4 = 111*	11* (P2)
P5 = 1000*	1110* (P4)
P6 = 11001*	1111* (P4) 1000* (P5) 110010* (P6) 110011* (P6)

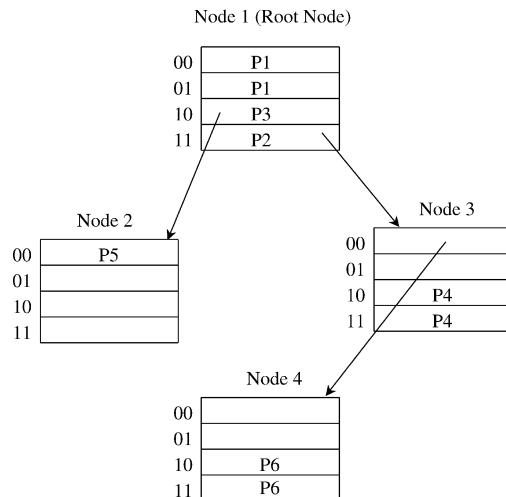


Fig. 13.6 The 2-bit trie structure corresponding to the forwarding table in Table 13.1.

example, a routing table, shown in the left column of Table 13.1, contains route prefixes P1–P6. The asterisk in each route prefix indicates that the remaining bits are insignificant. By using a 2-bit trie structure, the expanded version of the forwarding table can be obtained as shown in the right column of Table 13.1. Note that the prefix P2 = 1* can be expanded into 10* and 11*, but 10* is not used, since it overlaps P3, which is the longer matching prefix. The corresponding 2-bit trie structure is also shown in Figure 13.6.

The traversal to the node at each level of the K-bit trie is based on each K-bit part of the destination address of the packet. The advantage of the

K -bit trie structure is that it provides fast lookups; its disadvantage is the large memory space that is required for nodes of the structure.

13.4 PATRICIA TREE

Traditional implementations of routing tables use a version of Patricia trees [10]. A *Patricia tree* is an improvement on the 1-bit trie structure. It is based on the observation that each internal node of the trie that does not contain a route prefix and has only one child can be removed in order to shorten the path from the root node. By removing some internal nodes, the technique requires a mechanism to record which nodes are missing. A simple mechanism is to store a number, the *skip value*, in each node that indicates how many bits have been skipped on the path. The path-compressed version of the 1-bit trie in Figure 13.5 is shown in Figure 13.7.

Suppose that the destination address 10001101 is given. The IP lookup starts at the top and traverses the path indicated by the destination address, remembering the last time a flag (an asterisk) was seen. Starting at the top, the zero-length prefix has a flag. The first bit of 10001101 is 1, so we go to the right and get to the node with the name 1. There is no asterisk at this node, so {} is still the longest prefix matched so far. Since the second bit of 10101101 is 0, we go to the left. At the left branch, the skip value 2 indicates that in order to take the 0 branch out of the node 1, we must match the two subsequent bits 01. Since the second and the third bits of 10001101 are 00, there is no match. Thus, {} is returned as the longest matching prefix.

With around 40,000 prefix entries in a routing table, a straightforward implementation of Patricia tree structure is almost 2 Mbyte, and 15 or 16 nodes must be traversed to find a routing entry. There are optimizations that can reduce the size of a Patricia tree and improve lookup speeds. Nevertheless, the data structure is large, and too many expensive memory accesses are needed to search it.

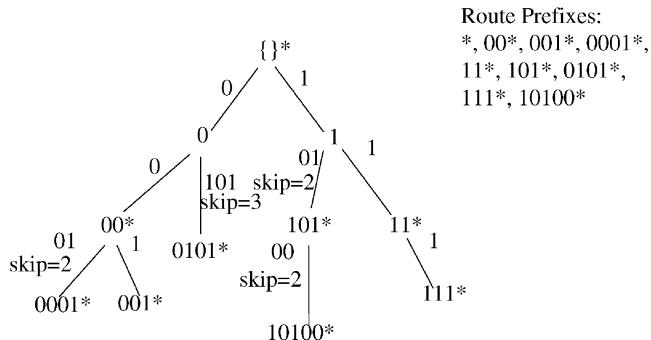


Fig. 13.7 The Patricia tree corresponding to the 1-bit trie in Figure 13.5.

13.5 SMALL FORWARDING TABLES FOR FAST ROUTE LOOKUPS

Degermark et al. [2] proposed a data structure that can represent large routing tables in a very compact form, small enough to fit entirely in a cache. This provides an advantage in that the fast IP route-lookup algorithm can be implemented in software running on general-purpose microprocessors.

The forwarding table is a representation of the binary tree spanned by all routing entries. This is called the *prefix tree*. The prefix tree is required to be full, that is, such that each node in the tree has either two or no children. Nodes with a single child must be expanded to have two children; the children added in this way are always leaves, and their next-hop information will be the same as that of the closest ancestor with next-hop information.

This procedure, illustrated in Figure 13.8, increases the number of nodes in the prefix tree, but allows building a small forwarding table. Note that it is not needed to actually build the prefix tree to build the forwarding table. The prefix tree is used to simplify the explanation. The forwarding table can be built during a single pass over all routing entries.

The forwarding table is essentially a tree with three levels. As shown in Figure 13.9, level 1 of the data structure covers the prefix tree down to depth 16, level 2 covers depths 17 to 24, and level 3 depths 25 to 32. Wherever a

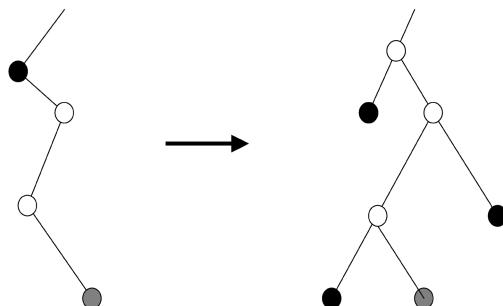


Fig. 13.8 Expanding the prefix tree to be complete.

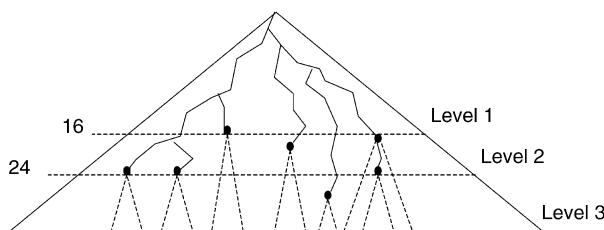


Fig. 13.9 The three levels of the data structure.

part of the prefix tree extends below level 16, a level-2 chunk describes that part of the tree. Similarly, chunks at level 3 describe parts of the prefix tree that are deeper than 24. The result of searching a level of the data structure is either an index into the next-hop table or an index into an array of chunks for the next level.

13.5.1 Level 1 of Data Structure

Imagine a cut through the prefix tree at depth 16. The cut is stored in a bit vector, with one bit per possible node at depth 16. For this, 2^{16} bits = 64 Kbit = 8 Kbyte are required. To find the bit corresponding to the initial part of an IP address, the upper 16 bits of the address is used as an index into the bit vector.

When there is a node in the prefix tree at depth 16, the corresponding bit in the vector is set. Also, when the tree has a leaf at a depth less than 16, the lowest bit in the interval covered by that leaf is set. All other bits are zero. A bit in the bit vector can thus be

- a one representing that the prefix tree continues below the cut: a root head (bits 6, 12, and 13 in Fig. 13.10), or
- a one representing a leaf at depth 16 or less: a genuine head (bits 0, 4, 7, 8, 14, and 15 in Fig. 13.10), or
- zero, which means that this value is a member of a range covered by a leaf at a depth less than 16 (bits 1, 2, 3, 5, 9, 10, and 11 in Fig. 13.10).

For genuine heads an index to the next-hop table is stored. Members will use the same index as the largest head smaller than the member. For root heads, an index to the level-2 chunk that represents the corresponding subtree is stored.

The head information is encoded in 16-bit pointers stored in an array. Two bits of the pointer encodes what kind of pointer it is, and the 14 remaining bits are indices either into the next-hop table or into an array containing level-2 chunks.

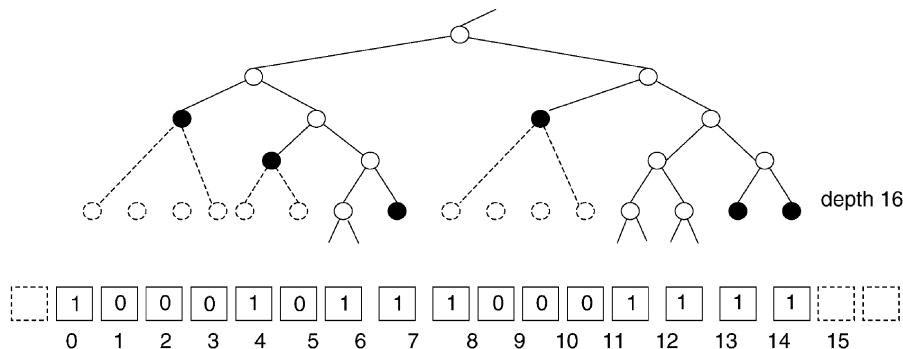


Fig. 13.10 Part of cut with corresponding bit vector.

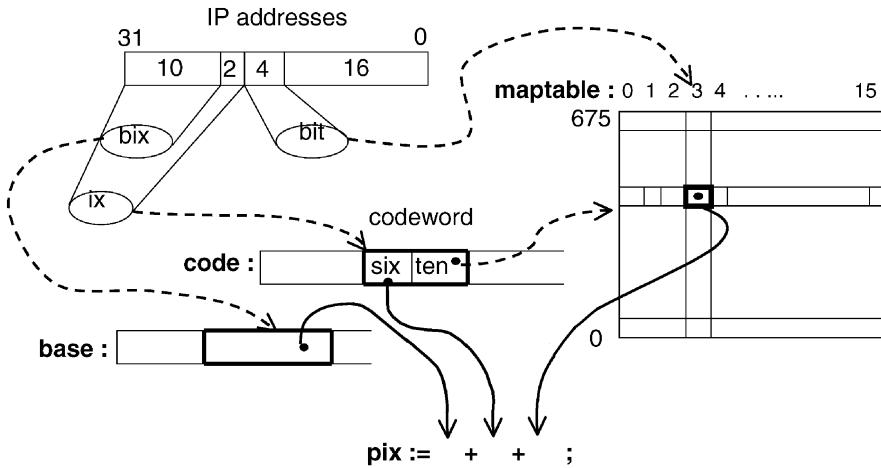


Fig. 13.11 Finding the pointer index.

So how to find the proper pointer? The bit vector is divided into bit masks of length 16, and there are $2^{12} = 4096$ of those. The position of a pointer in the array is obtained by adding three entities: a base index, a 6-bit offset, and a 4-bit offset. The base index plus 6-bit offset determines where the pointers corresponding to a particular bit mask are stored. The 4-bit offset specifies which pointer among those to retrieve. Figure 13.11 shows how these entities are found. The following paragraphs elaborate on the procedure.

Since bit masks are generated from a full prefix tree; not all combinations of the 16 bits are possible. A nonzero bit mask of length $2n$ can be any combination of two bit masks of length n or the bit mask with value 1. Let $a(n)$ be the number of possible nonzero bit masks of length $2n$. Then $a(n)$ is given by the recurrence

$$a(0) = 1, \quad a(n+1) = 1 + a(n)^2.$$

The number of possible bit masks with length 16 is thus $a(4) + 1 = 678$; the additional one is because the bit mask can be zero. An index into a table with an entry for each bit mask thus only needs 10 bits.

A table, `maptable`, is kept to map bit numbers within a bit mask to 4-bit offsets. The offset specifies how many pointers to skip over to find the wanted one, so it is equal to the number of set bits smaller than the bit index. These offsets are the same for all forwarding tables, regardless of what values the pointers happen to have. `maptable` is constant; it is generated once and for all.

The actual bit masks are not needed, and instead of the bit vector an array of 16-bit codewords consisting of a 10-bit index into `maptable` plus a 6-bit offset is kept. A 6-bit offset covers up to 64 pointers, so one base index per four codewords is needed. There can be at most 64K pointers, so the base indices need to be at most 16 bits ($2^{16} = 64K$).

The following steps are required to search the first level of the data structure; the array of codewords is called `code`, and the array of base addresses is called `base`. Figure 13.11 illustrates the procedure.

```

ix := high 12 bits of IP address
bit := low 4 of high 16 bits of IP address
codeword := code[ix]
ten := ten bits from codeword
six := six bits from codeword
bix := high ten bits of IP address
pix := base[bix] + six + maptable[ten] [bit]
pointer := level1_pointers[pix]
```

The code is extremely simple. A few bit extractions, array references, and additions are all that is needed. No multiplication or division instructions are required except for the implicit multiplications when indexing an array.

A total of seven bytes needs to be accessed to search the first level: a two-byte codeword, a two-byte base address, one byte (4 bits, really) in `maptable`, and finally a two-byte pointer. The size of the first level is 8 Kbyte for the code word array, plus 2 Kbyte for the array of base indices, plus a number of pointers. The 5.3 Kbytes required by `maptable` is shared among all three levels.

When the bit mask is zero or has a single bit set, the pointer must be an index into the next-hop table. Such pointers can be encoded directly into the codeword, and thus `maptable` need not contain entries for bit masks 1 and 0. The number of `maptable` entries is thus reduced to 676 (indices 0 through 675). When the ten bits in the codeword (`ten` above) are larger than 675, the codeword represents a direct index into the next-hop table. The six bits from the code word are used as the lowest six bits in the index, and `ten - 676` gives the upper bits of the index. This encoding allows at most $(1024 - 676) \times 2^6 = 22,272$ next-hop indices, which is more than the 16 K we are designing for. This optimization eliminates three memory references when a routing entry is located at depth 12 or more, and reduces the number of pointers in the pointer array considerably. The cost is a comparison and a conditional branch.

13.5.2 Levels 2 and 3 of Data Structure

Levels 2 and 3 of the data structure consist of chunks. A chunk covers a subtree of height 8 and can contain at most $2^8 = 256$ heads. A root head is level n . There are three varieties of chunks, depending on how many heads the imaginary bit vector contains. When there are

- 1–8 heads, the chunk is *sparse* and is represented by an array of the 8-bit indices of the heads, plus eight 16-bit pointers: a total of 24 bytes.

- 9–64 heads, the chunk is *dense*. It is represented analogously to level 1, except for the number of base indices. The difference is that only one base index is needed for all 16 codewords, because 6-bit offsets can cover all 64 pointers. A total of 34 bytes are needed, plus 18 to 128 bytes for pointers.
- 65–256 heads, the chunk is *very dense*. It is represented analogously to level 1. The 16 codewords and 4 base indices give a total of 40 bytes. In addition, the 65 to 256 pointers require 130 to 512 bytes.

Dense and very dense chunks are searched analogously to level 1. For sparse chunks, the first to eighth values are placed in decreasing order. To avoid a bad worst case when searching, the fourth value is examined to determine if the desired element is among the first four or last four elements. After that, a linear scan determines the index of the desired element, and the pointer with that index can be extracted. The first element less than or equal to the search key is the desired element. At most 7 bytes need to be accessed to search a sparse chunk.

Dense and very dense chunks are optimized analogously to level 1 as described in Section 13.5.1. In sparse chunks, two consecutive heads can be merged and represented by the smaller if their next hops are identical. When deciding whether a chunk is sparse or dense, this merging is taken into account, so that the chunk is deemed sparse when the number of merged heads is 8 or less. Many of the leaves that were added to make the tree full will occur in order and have identical next hops. Heads corresponding to such leaves will be merged in sparse chunks. This optimization shifts the chunk distribution from the larger dense chunks towards the smaller sparse chunks. For large tables, the size of the forwarding table is typically decreased by 5% to 15%.

13.5.3 Performance

Forwarding tables are constructed from routing tables of various sizes. For the largest routing tables with 40,000 routing entries, the data structure is 150–160 Kbyte, which is small enough to fit in the cache of a conventional general-purpose processor. With the table in the cache, a 200-MHz Pentium Pro or a 333-MHz Alpha 21164 can perform a few million IP lookups per second without special hardware, and no traffic locality is assumed.

13.6 ROUTE LOOKUPS IN HARDWARE AT MEMORY ACCESS SPEEDS

Gupta et al. [5] proposed a route lookup mechanism that, when implemented in a pipelined fashion in hardware, can achieve one route lookup every memory access. It is called the DIR-24-8-BASIC scheme. With current 50-ns

DRAM, this corresponds to approximately 20×10^6 packets per second, much faster than current commercially available routing lookup schemes.

The technique presented here is based on the following assumptions:

1. Memory is cheap.
2. The route lookup mechanism will be used in routers where speed is at a premium, for example, those routers that need to process at least 10 million packets per second.
3. On backbone routers there are very few routes with prefixes longer than 24 bits. By examining the MAE-EAST backbone routing table [11], we see that 99.93% of the prefixes are 24 bits or less.
4. IPv4 is here to stay for the time being. Thus, a hardware scheme optimized for IPv4 routing lookups is still useful today.
5. There is a single general-purpose processor participating in routing table exchange protocols and constructing a full routing table (including protocol-specific information, such as the route lifetime, for each route entry). The next-hop entries from this routing table are downloaded by the general-purpose processor into each forwarding table, and are used to make per-packet forwarding decisions.

13.6.1 The DIR-24-8-BASIC Scheme

The DIR-24-8-BASIC scheme makes use of the two tables shown in Figure 13.12, both stored in DRAM. The first table (called TBL24) stores all possible route prefixes that are not more than 24 bits long. This table has 2^{24} entries, addressed from 0.0.0 to 255.255.255. Each entry in TBL24 has the format shown in Figure 13.13. The second table (TBLlong) stores all route prefixes in the routing table that are longer than 24 bits.

Assume for example that we wish to store a prefix, X , in an otherwise empty routing table. If X is no more than 24 bits long, it need only be stored

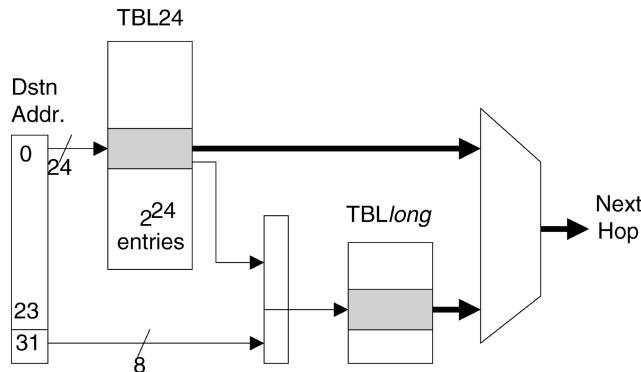


Fig. 13.12 Proposed DIR-24-8 BASIC architecture.

If longest route with this 24-bit prefix is < 25 bits long :

0	Next Hop
1 bit	15 bits

If longest route with this 24-bit prefix is > 24 bits long :

1	Index into 2nd table
1 bit	15 bits

Fig. 13.13 TBL24 entry format.

in TBL24: the first bit of the entry is set to zero to indicate that the remaining 15 bits designate the next hop. If, on the other hand, the prefix X is longer than 24 bits, then we use the entry in TBL24 addressed by the first 24 bits of X . We set the first bit of the entry to one to indicate that the remaining 15 bits contain a pointer to a set of entries in TBLLong.

In effect, route prefixes shorter than 24 bits are expanded; for example, the route prefix 128.23/16 will have entries associated with it in TBL24, ranging from the memory addresses 128.23.0 through 128.23.255. All 256 entries will have exactly the same contents (the next hop corresponding to the routing prefix 128.23/16). By using memory inefficiently, we can find the next-hop information within one memory access.

TBLLong contains all route prefixes that are longer than 24 bits. Each 24-bit prefix that has at least one route longer than 24 bits is allocated $2^8 = 256$ entries in TBLLong. Each entry in TBLLong corresponds to one of the 256 possible longer prefixes that share the single 24-bit prefix in TBL24. Note that because we are simply storing the next hop in each entry of the second table, it need be only 1 byte wide (if we assume that there are fewer than 255 next-hop routers—this assumption could be relaxed if the memory were wider than 1 byte).

When a destination address is presented to the route lookup mechanism, the following steps are taken:

1. Using the first 24 bits of the address as an index into the first table TBL24, we perform a single memory read, yielding 2 bytes.
2. If the first bit equals zero, then the remaining 15 bits describe the next hop.
3. Otherwise (if the first bit equals one), we multiply the remaining 15 bits by 256, add the product to the last 8 bits of the original destination address (achieved by shifting and concatenation), and use this value as a direct index into TBLLong, which contains the next hop.

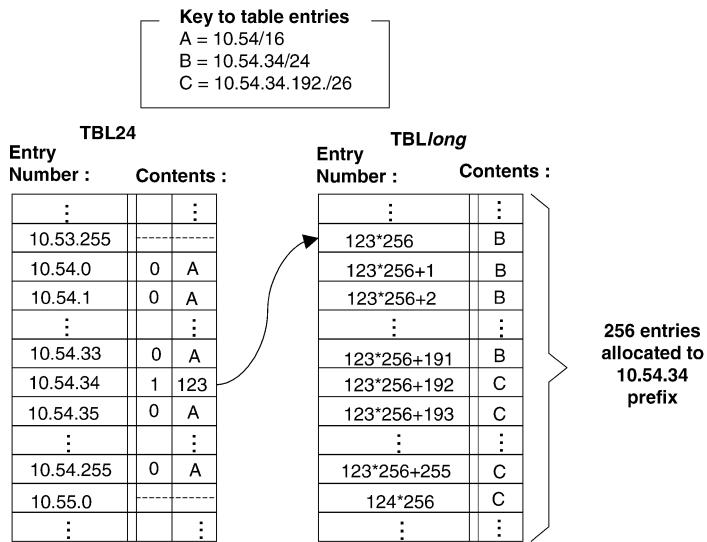


Fig. 13.14 Example of two tables containing three routes.

Consider the following examples of how route lookups are performed on the table in Figure 13.14. Assume that the following routes are already in the table: 10.54/16, 10.54.34/24, 10.54.34.192/26. The first route requires entries in TBL24 that correspond to the 24-bit prefixes 10.54.0 through 10.54.255 (except for 10.54.34). The second and third routes require that the second table be used (because both of them have the same first 24 bits and one of them is more than 24 bits long). So, in TBL24, we insert a one followed by an index (in the example, the index equals 123) into the entry corresponding to the prefix 10.54.34. In the second table, we allocate 256 entries starting with memory location 123×256 . Most of these locations are filled in with the next hop corresponding to the 10.54.34 route, but 64 of them [those from $(123 \times 256) + 192$ to $(123 \times 256) + 255$] are filled in with the next hop corresponding to the 10.54.34.192 route.

Now assume that a packet arrives with the destination address 10.54.22.147. The first 24 bits are used as an index into TBL24, and will return an entry with the correct next hop (*A*). If a second packet arrives with the destination address 10.54.34.23, the first 24 bits are used as an index into the first table, which indicates that the second table must be consulted. The lower 15 bits of the entry (123 in this example) are combined with the lower 8 bits of the destination address, and used as an index into the second table. After two memory accesses, the table returns the next hop (*B*). Finally, let's assume that a packet arrives with the destination address 10.54.34.194. Again, TBL24 indicates that TBL_{long} must be consulted, and the lower 15 bits of the entry

are combined with the lower 8 bits of the address to form an index into the second table. This time the index an entry associated with the 10.54.34.192/26 prefix (C).

13.6.2 Performance

As a summary, the advantages and disadvantages associated with the basic DIR-24-8-BASIC scheme are listed:

Advantages:

1. Although (in general) two memory accesses are required, these accesses are in separate memories, allowing the scheme to be pipelined.
2. Except for the limit on the number of distinct 24-bit-prefixed routes with length greater than 24 bits, this infrastructure will support an unlimited number of routes.
3. The total cost of memory in this scheme is the cost of 33 Mbyte of DRAM. No exotic memory architectures are required.
4. The design is well suited to hardware implementation.
5. When pipelined, 20×10^6 packets per second can be processed with currently available 50-ns DRAM. The lookup time is equal to one memory access time.

Disadvantages:

1. Memory is used inefficiently.
2. Insertion and deletion of routes from the table may require many memory accesses.

13.7 IP LOOKUPS USING MULTIWAY SEARCH

Lampson et al. [9] showed how to apply a binary search to find best-matching prefixes using two routing entries per prefix and doing some precomputations. They also proposed the way to use an initial array indexed by the first X bits of the address together with taking advantage of cache line size to do a multiway search with six-way branching. Measurements using a practical routing table of 30,000 entries yield a worst case lookup time of 490 ns, five times faster than the Patricia trie scheme.

13.7.1 Adapting Binary Search for Best-Matching Prefix

Binary search can be used to solve the best-matching-prefix problem, but only after several subtle modifications. Assume for simplicity, in the examples, that we have 6-bit addresses and three prefixes 1^* , 101^* , and 10101^* .

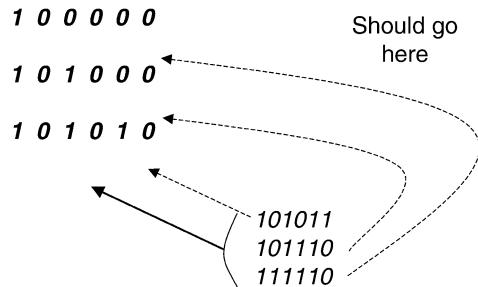


Fig. 13.15 Placing the three prefixes 1^* , 101^* , and 10101^* in a binary search table by padding each prefix with 0's to make 6-bit strings and sorting the resulting strings. Note that the addresses 101011, 101110, and 111110 all end up in the same region in the table.

First, binary search does not work with variable-length strings. Thus the simplest approach is to pad each prefix to be a 6-bit string by adding zeros. This is shown in Figure 13.15.

Now consider a search for the three 6-bit addresses 101011, 101110, and 111110. Since none of these addresses are in the table, binary search will fail. Unfortunately, on failure all three of these addresses will end up at the end of the table, because all of them are greater than 101010, which is the last element in the table. Notice however that each of these three addresses (see Fig. 13.15) has a different best-matching prefix.

Thus we have two problems with naive binary search: first, when we search for an address, we end up far away from the matching prefix (potentially requiring a linear search); second, multiple addresses that match to different prefixes end up in the same region in the binary table (Fig. 13.15).

To solve the second problem, we recognize that a prefix like 1^* is really a range of addresses from 100000 to 111111. Thus instead of encoding 1^* by just 100000 (the start of the range), we encode it using both start and end of range. Thus, each prefix is encoded by two full-length bit strings. These bit strings are then sorted. The result for the same three prefixes is shown in Figure 13.16. The start and end of a range (corresponding to a prefix) are connected by a line in Figure 13.16. Notice how the ranges are nested. If we now try to search for the same set of addresses, they each end in a different region in the table. To be more precise, the search for address 101011 ends in an exact match. The search for address 101110 ends in a failure in the region between 101011 and 101111 (Fig. 13.16), and the search for address 111110 ends in a failure in the region between 101111 and 111111. Thus it appears that the second problem (multiple addresses that match different prefixes ending in the same region of the table) has disappeared.

To see that this is a general phenomenon, consider Figure 13.17. The figure shows an arbitrary binary search table after every prefix has been

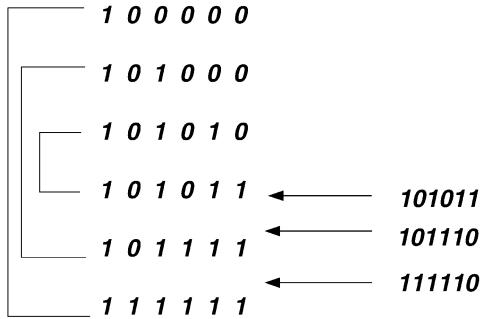


Fig. 13.16 Each prefix is encoded in the table as a range using two values: the start and end of range. This time the addresses that match different prefixes end up in different ranges.

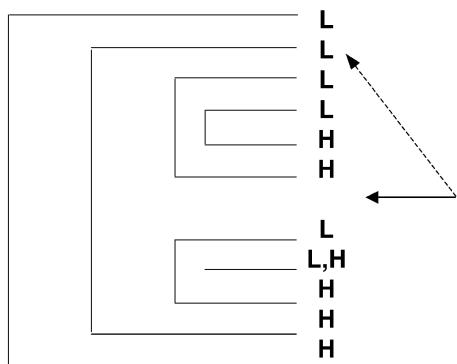


Fig. 13.17 Why each range in the modified binary search table maps to a unique prefix.

encoded by the low points (marked *L* in Fig. 13.17) and the high points (marked *H*) of the corresponding range. Consider an arbitrary position indicated by the solid arrow. If binary search for address *A* ends up at this point, which prefix should we map *A* to? It is easy to see the answer visually from Figure 13.17. If we start from the point shown by the solid arrow and we go back up the table, the prefix corresponding to *A* is the first *L* that is not followed by a corresponding *H* (see dashed arrow in Fig. 13.17).

Why does this work? Since we did not encounter an *H* corresponding to this *L*, it clearly means that *A* is contained in the range corresponding to this prefix. Since this is the first such *L*, that the smallest such range. Essentially, this works because the best matching prefix has been translated to the problem of finding the narrowest enclosing range.

Unfortunately, the solution depicted in Figure 13.16 and Figure 13.17 does not solve the first problem: notice that binary search ends in a position that is

		>	=
P1)	1 0 0 0 0 0	P1	P1
P2)	1 0 1 0 0 0	P2	P2
P3)	1 0 1 0 1 0	P3	P3
	1 0 1 0 1 1	P2	P3
	1 0 1 1 1 1	P1	P2
	1 1 1 1 1 1	-	P1

Figure 13.18 The final modified binary search table with precomputed prefixes.

far away (potentially) from the actual prefix. If we were to search for the prefix (as described earlier), we could have a linear-time search. However, the modified binary search table shown in Figure 13.17 has a nice property we can exploit: *Any region in the binary search between two consecutive numbers corresponds to a unique prefix*. As described earlier, the prefix corresponds to the first L before this region that is not matched by a corresponding H that also occurs before this region. Similarly, every exact match corresponds to a unique prefix.

But if this is the case, we can precompute the prefix corresponding to each region and to each exact match. This can slow down insertion. However, the insertion or deletion of a new prefix should be a rare event (the next hop to reach a prefix may change rapidly, but the addition of a new prefix should be rare) compared to packet forwarding times. Thus slowing down insertion costs for the sake of faster forwarding is a good idea. Essentially, the idea is to add the dashed line pointer shown in Figure 13.17 to every region.

The final table corresponding to Figure 3.17 is shown in Figure 13.18. Notice that with each table entry E , there are two precomputed prefix values. If binary search for address A ends in a failure at E , it is because $A > E$. In that case, we use the $>$ pointer corresponding to E . On the other hand, if binary search for address A ends in a match at E , we use the $=$ pointer. Notice that for an entry like 101011, the two entries are different. If address A ends up at this point and is greater than 101011, clearly the right prefix is $P2 = 101^*$. On the other hand, if binary search for address A ends up at this point with equality, the right prefix is $P3 = 10101^*$. Intuitively, if an address A ends up equal to the high point of a range R , then A falls within the range R ; if A ends up greater than the high point of R , then A falls within the smallest range that encloses R .

13.7.2 Precomputed 16-Bit Prefix Table

The worst case number of memory accesses of the basic binary search scheme can be improved with a precomputed table of best-matching prefixes for the first Y bits. The main idea is to effectively partition the single binary search

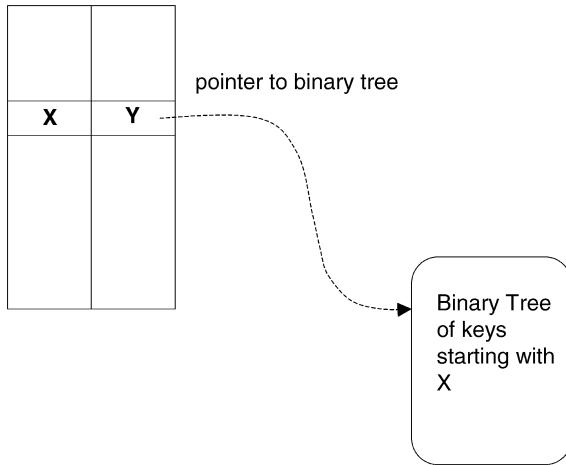


Fig. 13.19 The array element with index X will have the best-matching prefix of X (say Y) and a pointer to a binary tree of all prefixes that have X as a prefix.

table into multiple binary search tables for each value of the first Y bits. This is illustrated in Figure 13.19. $Y = 16$ is chosen for what follows, as the resulting table size is about as large as we can afford, while providing maximum partitioning.

The best-matching prefixes for the first 16-bit prefixes can be precomputed and stored in a table. This table would then have $\text{Max} = 65536$ elements. For each index X of the array, the corresponding array element stores the best matching prefix of X . Additionally, if there are prefixes of longer length with that prefix X , the array element stores a pointer to a binary search table or tree that contains all such prefixes.

Without the initial table, the worst case possible number of memory accesses is $\log_2 N + 1$, where N is the number of prefixes in the table. For large databases this could be 16 or more memory accesses. For a publicly available IP backbone router database [11], this simple trick of using an array as a front end reduces the maximum number of prefixes in each partitioned table to 336 from the maximum value of over 30,000. This leads to a worst case of 10 memory accesses.

13.7.3 Multiway Binary Search: Exploiting the Cache Line

Today processors have wide cache lines. The Intel Pentium Pro has a cache line size of 32 bytes. Main memory is usually arranged in a matrix form, with rows and columns. Accessing data given a random row address and column address has an access time of 50 to 60 ns. However, when using SDRAM, filling a cache line of 32 bytes, which is a burst access to four contiguous

64-bit DRAM locations, is much faster than accessing four random DRAM locations. When accessing a burst of contiguous columns in the same row, while the first datum would be available only after 60 ns, further columns would be available much faster.

By taking the advantage of this observation, data structures can be reorganized to improve locality of access. To make use of the cache line fill and the burst mode, keys and pointers in search tables can be laid out to allow multiway search instead of binary search. This effectively allows us to reduce the search time of binary search from $\log_2 N$ to $\log_{k+1} N$, where k is the number of keys in a search node. The main idea is to make k as large as possible so that a single search node (containing k keys and $2k + 1$ pointers) fits into a single cache line. If this can be arranged, an access to the first word in the search node will result in the entire node being prefetched into cache. Thus the accesses to the remaining keys in the search node are much cheaper than a memory access.

If we use k keys per node, then we need $2k + 1$ pointers, each of which is a 16-bit quantity. Based on a Pentium Pro processor (with the cache line size of 32 bytes), we can choose $k = 5$ keys and do a six-way search. For example, if there are keys k_1, \dots, k_8 , a three-way tree is given in Figure 13.20. The initial full array of 16 bits followed by the six-way search is depicted in Figure 13.21.

From a practical routing table obtained from Merit [11] containing 30,000 routing entries, it is found that after using a 16-bit initial array, the worst case has 336 routing entries left for doing a six-way search. This leads to a worst case of four memory accesses, since $6^4 = 1296$ takes only four memory accesses.

Each node in the six-way search table has five keys k_1 to k_5 , each of which is 16 bits. There are equal to pointers p_1 to p_5 corresponding to each of these keys. Pointers p_{01} to p_{56} correspond to ranges demarcated by the keys. This is shown in Figure 13.22. Among the keys we have the relation

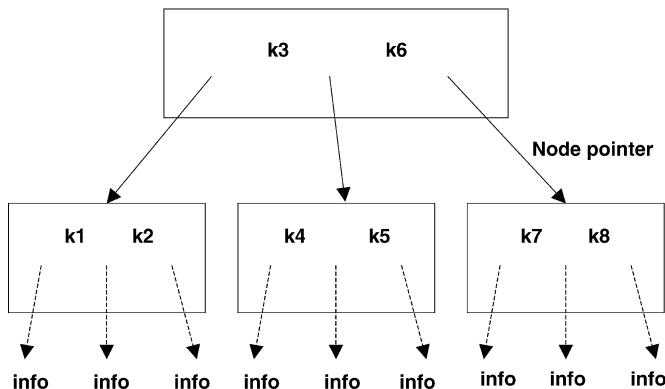


Fig. 13.20 A three-way tree for eight keys.

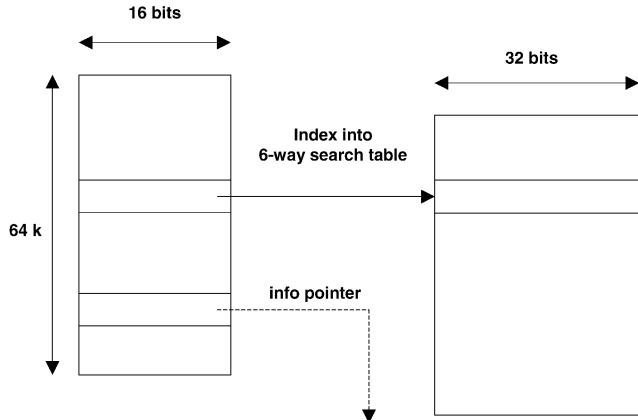


Fig. 13.21 The initial 16-bit array, with pointer to the corresponding six-way search nodes.

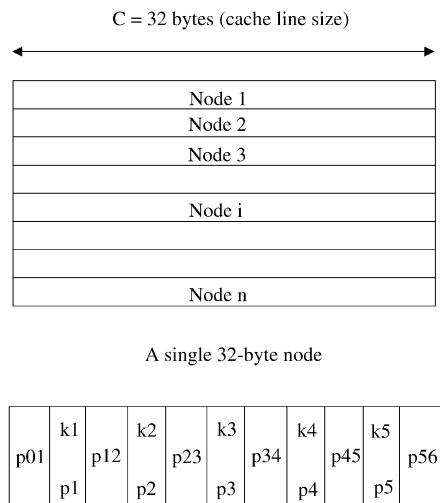


Fig. 13.22 The structure of the six-way search node.

$k_i \leq k_{i+1}$. Each pointer has a bit that says it is an information pointer or a next-node pointer.

IP Route Lookup Algorithm:

1. Index into the first 16-bit array, using the first 16 bits of the address.
2. If the pointer at the location is an information pointer, return it. Otherwise enter the six-way search with the initial node given by the pointer, the key being the next 16 bits of the address.

3. In the current six-way node, locate the key. We use binary search among the keys within a node. If the key equals any of the keys key_i in the node, use the corresponding pointer ptr_i . If the key falls in any range formed by the keys, use the pointer $ptr_{i,i+1}$. If this pointer is an information pointer, return it; otherwise repeat this step with the new six-way node given by the pointer.

13.7.4 Performance

For a practical routing table, a basic binary search scheme for the best-matching prefix requires 16 memory accesses for an IP route lookup. By using an initial precomputed 16-bit array, the number of required memory accesses can be reduced to 9.

By applying a six-way branching search technique (after using a 16-bit initial array) that exploits the fact that a Pentium Pro processor prefetches a 32-byte cache line when doing a memory access, we obtain a worst case of five memory accesses. By using a Pentium Pro machine with a 200-MHz clock and a sample of a practical routing table with over 32,000 route entries, a worst case time of 490 ns and an average time of 100 ns for an IP route lookup are obtained. The structure uses 0.7 Mbyte of memory.

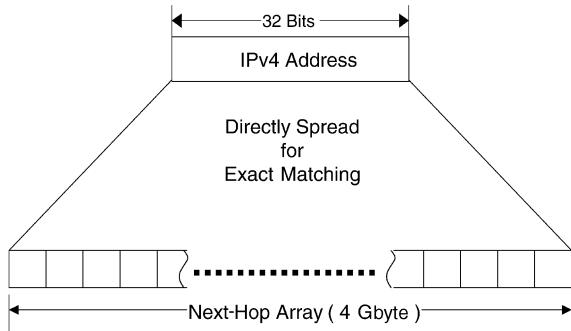
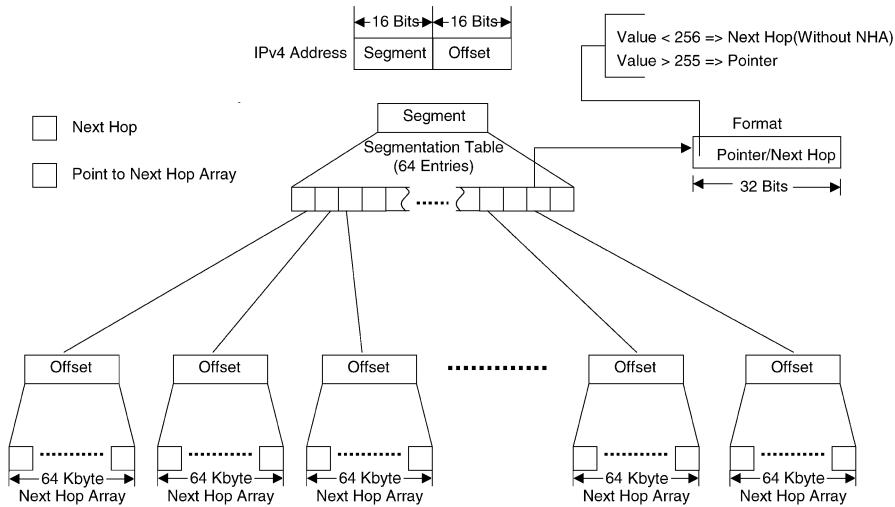
13.8 IP ROUTE LOOKUPS FOR GIGABIT SWITCH ROUTERS

Huang et al. [6] proposed a route-lookup mechanism that needs only tiny SRAM and can be implemented using a hardware pipeline. The forwarding table, based on their scheme, is small enough to fit into a faster SRAM with low cost. For example, a large routing table with 40,000 routing entries can be compacted into a forwarding table of 450–470 kbytes costing less than US\$30. Most route lookups need only one memory access; no lookup needs more than three memory accesses.

13.8.1 Lookup Algorithms and Data Structure Construction

The most straightforward way to implement a lookup scheme is to have a forwarding table in which an entry is designed for each 32-bit IP address, as depicted in Figure 13.23. This design needs only one memory access for each IP-route lookup, but the size of the forwarding table [next-hop array (NHA)] is huge (2^{32} bytes = 4 Gbyte) [5].

To reduce the size of the forwarding table, an indirect lookup can be employed (see Fig. 13.24) [5]. Each IP address is split into two parts: (a) segment (16-bit) and (b) offset (16-bit). The segmentation table has 64K entries (2^{16}), and each entry (32-bit) records either the next hop (port number, value < 256) of the routing or a pointer (value > 255) that points

**Fig. 13.23** Direct lookup mechanism.**Fig. 13.24** Indirect lookup mechanism.

to the associated NHA. Each NHA consists of 64K entries (2^{16}) and each entry (8-bit) records the next hop (port number) of the destination IP address. Thus, for a destination IP address $a.b.x.y$, the $a.b$ is used as the index of the segmentation table and the $x.y$ is employed as the index of the associated NHA, if necessary. Thus, for a segment $a.b$, if the length of the longest prefix belonging to this segment is less than or equal to 16, then the corresponding entries of the segmentation table store the output port directly, and the associated NHA is not necessary. On the other hand, if the length of the longest prefix belonging to this segment is greater than 16, then an associated 64-Kbyte NHA is required. In this design, a maximum of two

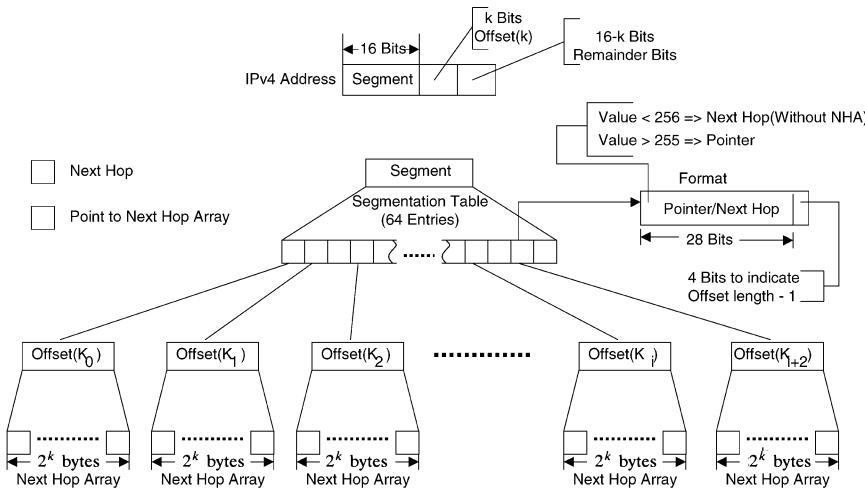


Fig. 13.25 Indirect lookup mechanism with variable offset length.

memory accesses is needed for an IP-route lookup. The indirect lookup method [5] employs a 24-bit segment length, and therefore a 16-Mbyte segmentation table is required.

Although the indirect lookup scheme furnishes a fast lookup (up to two memory accesses), it does not consider the distribution of the prefixes belonging to a segment. A 64-Kbyte NHA is required as long as the length of the longest prefix belonging to this segment is greater than 16. The size of the associated NHA can be reduced further by considering the distribution of the prefixes within a segment, as demonstrated in Figure 13.25. The IP address is still partitioned into segment (16-bit) and offset (≤ 16 -bit). The segmentation table has 64K entries, each of which (32-bit) is divided into two fields: pointer–next-hop (28-bit) and offset length (4-bit). The first field records either the next hop (port number, value < 256) of the routing, or a pointer (value > 255) to the associated NHA. The second field indicates the length of the offset (k bits, $0 < k \leq 16$). For an offset of k -bit length, the associated NHA has 2^k entries. Now, for each segment, the offset length indicates how many entries are needed in the associated NHA. This depends on the prefixes of the segment. Thus, for a segment $a.b$, assume there are m prefixes and the longest one is of length $16 < l \leq 32$; then the offset length k for this segment is $l - 16$. This also means that for a destination IP address $a.b.x.y$, the $a.b$ is used as the index of the segmentation table and the leftmost k bits of $x.y$ [from 16th bit to $(16 + k - 1)$ th bit] is employed as that of the associated NHA, if necessary. Actually, the NHA depends on the set of prefixes of the segment as well as the length of each prefix.

The mechanism to construct the NHA of a segment is now described. Let l_i and h_i denote the length and the next hop (output port) of a route prefix

p_i , respectively. Let $o_i = l_i - 16$ and $k = \max\{o_i | p_i \in P\}$ (NHA is of size 2^k). Let $P = \{p_0, p_1, \dots, p_{m-1}\}$ be the set of sorted prefixes of a segment. Thus, for any pair of prefixes p_i and p_j , $i < j$ if and only if $l_i \leq l_j$. For each prefix p_i in P , let S_i^0 and E_i^0 denote the data structure of the start point and end point of p_i , respectively, that should be forwarded to h_i . Without loss of generality, let $\text{ma}(S_i^0)$ and $\text{ma}(E_i^0)$ stand for the memory addresses of S_i^0 and E_i^0 in the NHA, respectively. Also let $\text{op}(S_i^0)$ and $\text{op}(E_i^0)$ stand for the output port destination addresses of the start point and end point, respectively. Assume $p_i = a.b.x.y$. Let $x_0, x_1, x_2, \dots, x_{15}$ denote the binary digits of $x.y$; let $s_0, s_1, s_2, \dots, s_{k-1}$ denote the start point mask, where $s_j = 1$, $j < o_i$, and $s_j = 0$, $j \geq o_i$; and let $e_0, e_1, e_2, \dots, e_{k-1}$ denote the end point mask, where $e_j = 0$, $j < o_i$, and $e_j = 1$, $j \geq o_i$. Then we have $\text{ma}(S_i^0) = (x_0, x_1, x_2, \dots, x_{k-1} \text{ AND } s_0, s_1, \dots, s_{k-1})$ and $\text{ma}(E_i^0) = (x_0, x_1, x_2, \dots, x_{k-1} \text{ OR } e_0, e_1, e_2, \dots, e_{k-1})$. For example, assume $p_i = a.b.58.0$, $l_i = 26$, and $k = 12$ (the longest prefix in this segment is of length 28). Then the binary form of 58.0 (k -bit) is 001110100000, and we have $s_0, s_1, s_2, \dots, s_{k-1} = 11111111100$ and $e_0, e_1, e_2, \dots, e_{k-1} = 000000000011$. We have $\text{ma}(S_i^0) = 001110100000 = 928$ and $\text{ma}(E_i^0) = 001110100011 = 931$. This also means that $\text{NHA}_j = h_i$, $\text{ma}(S_i^0) \leq j \leq \text{ma}(E_i^0)$.

Note that for each prefix p_i in P , we can find a pair S_i^0 and E_i^0 . The memory addresses between $\text{ma}(S_i^0)$ and $\text{ma}(E_i^0)$ can be depicted as an interval $[\text{ma}(S_i^0), \text{ma}(E_i^0)]$, and the set P of prefixes as a set of intervals. If all the intervals do not overlap, then the NHA can be directly constructed by setting $\text{NHA}_j = h_i$, $\text{ma}(S_i^0) \leq j \leq \text{ma}(E_i^0)$. However, in most cases, this may not be true. An overlap represents that a destination IP address has more than one matching address. However, the proposed design looks for the longest matching. Thus, if a memory address j belongs to a set P' of intervals simultaneously, then we should set $\text{NHA}_j = h_i$, where p_i is the longest prefix of P' . For example, assume each route prefix is presented as prefix/prefix length/next hop (output port). Then the set P of seven sorted prefixes given by 168.188/16/1, 168.188.36/18/2, 168.188.68/24/1, 168.188.192.3/26/3, 168.188.86.8/28/6, 168.188.78.9/30/8, 168.188.68.10/32/10 can be presented as the six intervals shown in Figure 13.26(a). The corresponding constructed NHA is given in Figure 13.26(b).

Now, let us consider another example where we have three prefixes on a segment: $p_0 = 11.7/16/1$, $p_1 = 11.7.12.10/32/2$. Since the longest prefix is of length 32, the offset length k equals to $32 - 16 = 16$ (recorded as 1111 in the offset length field). Then the NHA for prefix 11.7 has $2^{16} = 64K$ entries. In this case, we have $\text{ma}(S_0^0) = 0000000000000000 = 0$, $\text{ma}(E_0^0) = 1111111111111111 = 65535$, $\text{ma}(S_1^0) = \text{ma}(E_1^0) = 0000110000001010 = 3082$. Thus, we have $\text{NHA}_0 - \text{NHA}_{3081} = 1$, $\text{NHA}_{3082} = 2$, and $\text{NHA}_{3083} - \text{NHA}_{65535} = 1$. For destination IP address 11.7.12.10, the offset is 3082, and the output port can be found in $\text{NHA}_{3082} = 2$. For the destination IP address 11.7.x.y, $0 \leq x, y \leq 255$, except 11.7.12.10, the offset q is $256x + y$, and the output port can be found in $\text{NHA}_q = 1$.

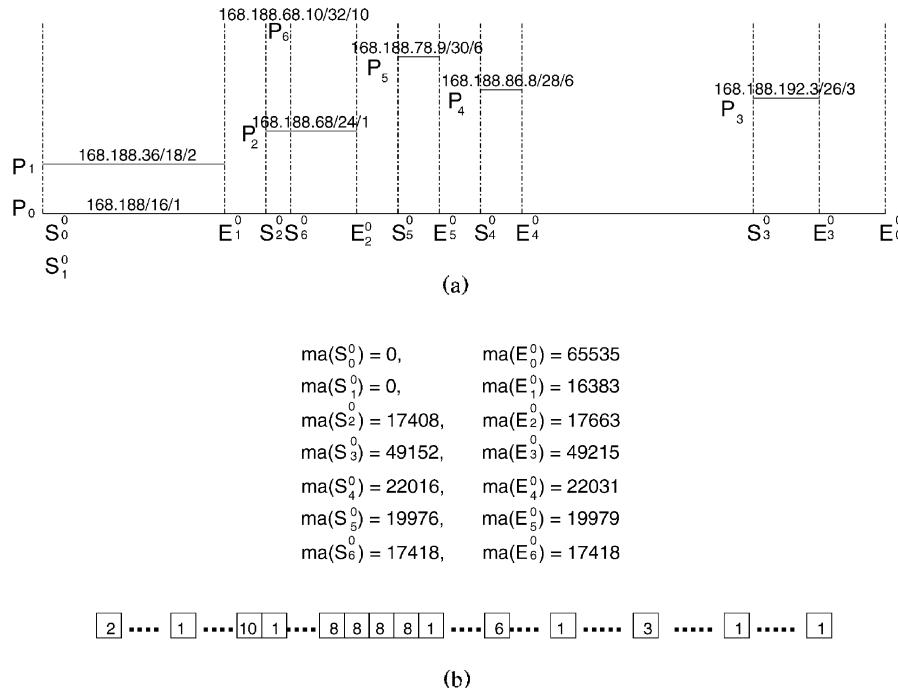


Fig. 13.26 NHA construction example: (a) interval presentation of P , (b) constructed NHA.

The formal algorithm for constructing the NHA of a segment is given as follows.

NHA Construction Algorithm

Input: The set of route prefixes of a segment.

Output: The corresponding NHA of this segment.

Step 1. Let l_i and h_i be the length and output port of a route prefix p_i , respectively.

Step 2. Let $P = \{p_0, p_1, \dots, p_{m-1}\}$ be the set of sorted prefixes of a segment. Thus, for any pair of prefixes p_i and p_j , $i < j$ if and only if $l_i \leq l_j$.

Step 3. Let $k = l_{m-1} - 16$ /* The size of NHA is 2^k */

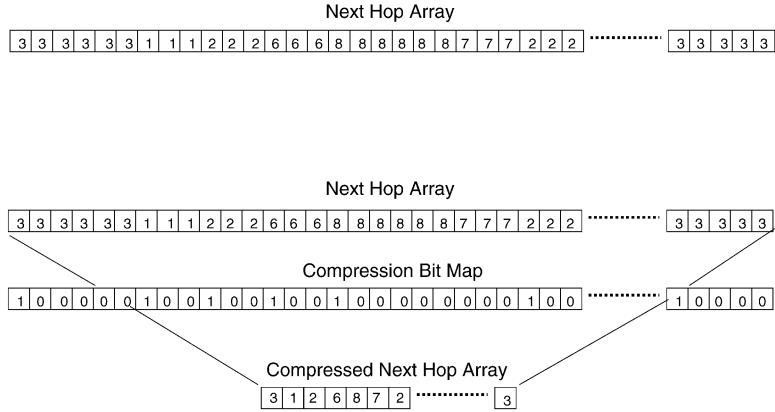
Step 4. For each prefix p_i in P , calculate S_i^0 and E_i^0 .

Step 5.

For $i = 0$ to $m - 1$ do

$$\text{NHA}_j = h_i, \text{ ma}(S_i^0) \leq j \leq \text{ma}(E_i^0);$$

Step 6. Stop.

**Fig. 13.27** NHA compression example.

In the worst case, the number of memory accesses for an IP address lookup is still two for this design, but the total amount of memory required is reduced significantly.

The forwarding database structure (NHAs) shown in Figure 13.25 can be further improved by employing the concept of compression. Thus, for each segment with offset length $k > 3$, the associated NHA can be replaced by a codeword array (CWA) and a compressed NHA (CNHA). To construct the CWA, we employ a compression bit map (CBM), one bit for each entry in the original NHA. The compression rule is as follows. Let a_i denote the value (port number) of the i th entry of the NHA, b_i the corresponding bit in the CBM, and c_j the value (port number) of the j th entry of the CNHA. Initially, $c_0 = a_0$, $b_0 = 1$, and $j = 1$. Then scan the NHA from left to right. If $a_{i+1} = a_i$, then $b_{i+1} = 0$, else $b_{i+1} = 1$, $c_j = a_{i+1}$, and $j = j + 1$. For example, the NHA shown in Figure 13.27(a) can be compressed into the corresponding CBM and CNHA as shown in Figure 13.27(b).

The CBM and CNHA of a segment can be constructed directly without constructing the NHA first. The algorithm to construct the CBM and CNHA directly from the given prefixes of a segment is depicted as follows. Before starting, we need to note that for any two distinct prefixes in a segment, either one is completely contained in the other, or the two distinct prefixes have no entries overlapping with each other [5].

Algorithm for CBM and CNHA Construction

Input: The set $P = \{p_0, p_1, \dots, p_{m-1}\}$ of sorted route prefixes of a segment and $L = S_0^0, E_0^0, S_1^0, E_1^0, \dots, S_{m-1}^0, E_{m-1}^0$.

Output: CBM and CNHA.

Step 1. Sort L by memory address in the segment. For identical memory addresses, keep their order the same in L .

Step 2 Let $A = \emptyset$ and stack $C = \emptyset$.

Step 3.

Process the elements in L from left to right, and for each element do

Begin

If the selected element is a start point S_i^0 then

Push S_i^0 in C . Append S_i^0 to A .

Else /* It is an end point E_i^0 */

Begin

Remove top element from stack C .

If the top of stack C is S_j^k then

Begin

Append S_j^{k+1} in A ,

where $\text{op}(S_j^{k+1}) = \text{op}(S_j^k)$ and $\text{ma}(S_j^{k+1}) = \text{ma}(E_i^0) + 1$.

Replace the top of stack C by S_j^{k+1} .

End

Else /* Stack C is \emptyset */

Do nothing.

End

End

Step 4. Compact A : for consecutive elements S_j^k and S_p^q , remove S_j^k from A if $\text{ma}(S_j^k) = \text{ma}(S_p^q)$, and remove S_p^q from A if $\text{op}(S_j^k) = \text{op}(S_p^q)$.

Step 5. Remove each element S_j^k from A whose $\text{ma}(S_j^k) > \text{ma}(E_0^0)$.

Step 6.

For $i = 0$ to $|A| - 1$ do

/* $|A|$ is the number of elements in A */

Begin

Let S_j^k be the i th element in A ;

$\text{CBM}_p = 1$, where $p = \text{ma}(S_j^k)$;

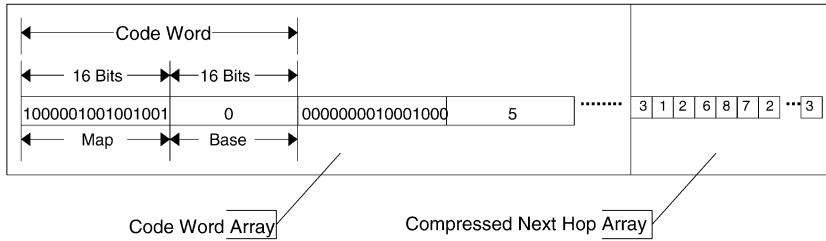
$\text{CNHA}_i = \text{op}(S_j^k)$

End

Step 7. Stop.

The time complexity of the proposed algorithm is $O(n \log n)$, where n is the number of prefixes in a segment. Since this algorithm constructs the CBM_s and CNHA_s directly from the given prefixes, the forwarding table can be built in a very short time.

The CBM should be encoded as a sequence of codewords (CWA) as follows. Each codeword consists of a map (16-bit) and a base (16-bit). The

**Fig. 13.28** Codeword array example.

CBM is treated as a bit stream and partitioned into a sequence of 16-bit maps. Then these maps are put into the codewords, one for each codeword, sequentially. The base of each codeword equals the number of 1's accumulated in the maps of previous codewords. For example, the CBM shown in Figure 13.27(b) is encoded as the codewords depicted in Figure 13.28. The maps of the first two codewords are 1000000010000000 and 0000000010001000, respectively. For the first codeword, the base has a value of zero. For the second codeword, since the number of 1's accumulated in the maps of previous codewords is two, we have a base value of two. The base is used to indicate the start entry of the associated CNHA. Thus, for an offset value q , the output port can be computed as follows. Let $cw_s = map_s + base_s$ be the code word contains this offset, where $s = q \text{ div } 16$, with div referring to integer division. Let $w = q \bmod 16$ denote the corresponding bit of q in map_s , and $|w|$ stand for the number of accumulated 1's from the 0th bit to the w th bit of map_s . Then the output port of an offset value q is calculated as

$$\text{op}_q = \text{CHNA}_t, \quad \text{where } t = base_s + |w| - 1.$$

For example, consider the case shown in Figures 13.27 and 13.28 again. For offset $q = 8$, we have $s = 0$, $w = 8$, and $|w| = 2$; then $t = base_0 + |w| - 1 = 0 + 2 - 1 = 1$, and the corresponding output port is $\text{CNHA}_1 = \text{port } 8$. For offset value $q = 25$, we have $s = 1$, $w = 9$, and $|w| = 1$; then $t = (base_1 + |w| - 1) = 2 + 1 - 1 = 2$, and the corresponding output port is $\text{CNHA}_2 = \text{port } 7$.

To update the forwarding table, we can either rebuild a new one in a short time or use a special hardware design, such as dual-port memory or dual-memory banks.

13.8.2 Performance

The proposed route lookup mechanism needs only tiny SRAM and can be easily implemented in hardware. Based on the data obtained from [11], a large routing table with 40,000 routing entries can be compacted to a

forwarding table of 450–470 Kbyte. Most of the address lookups can be done by a memory access. In the worst case, the number of memory accesses for a lookup is three. When implemented in a pipeline in hardware, the proposed mechanism can achieve one routing lookup every memory access. With current 10-ns SRAM, this mechanism furnishes approximately 100×10^6 routing lookups per second.

13.9 IP ROUTE LOOKUPS USING TWO-TRIE STRUCTURE

IP route lookups based on a so-called *two-trie structure* are described here. In the two-trie structure, the nodes representing the front and rear parts of the prefixes are shared so that the resulting number of nodes in the structure can be reduced. Moreover, it still provides fast lookups. The original two-trie structure was proposed by Aoe et al. [1]. Their algorithms can be adapted to various applications as long as searching is done in the exact match manner. A new version of the two-trie structure and its related algorithms that was specifically designed for doing IP route lookups was described in [7]. Specifically, the lookup algorithm does the longest-prefix match, while the updating algorithms have the property in maintaining the next hop information of the longest prefixes in the structure when the prefix addition or deletion is performed.

A K -bit two-trie structure consists of two K -bit tries, the *front trie* and the *rear trie*, joining leaf nodes together in the middle. A node is called a front (rear) node if it is on the front (rear) trie. Both tries are allowed to traverse in both directions. The direction outgoing from the root node of each trie to its own children nodes is called the *forward direction*, while the opposite one is called the *backward direction*.

We assume that any route prefix (or IP address) X of length Y bits is represented in the dotted decimal form $\langle x(0).x(1)\dots.x(N) \rangle$, where $x(i)$ is a K -bit-wide integer of the prefix X , $0 \leq i \leq N - 1$; $x(N)$ is a special symbol $\#$; and $N = \lceil Y/K \rceil$. For example, given $K = 8$, a prefix $X = 128.238.0.0$ of length $Y = 16$ bits would be represented as $\langle 128.238.\# \rangle$. The special symbol $\#$ indicates the end of the prefix and is used to distinguish, for example, between $\langle 128.238.\# \rangle$ and $\langle 128.238.3.\# \rangle$, so that a prefix is not allowed within another prefix. If Y is not a multiple of K bits (for example, if $K = 8$, $X = 128.238.32.0$, and $Y = 20$), then X will be transformed to a set of prefixes $\langle 128.238.j. \neq \rangle$, where $32 \leq j \leq 47$.

Figure 13.29 shows an example of the two-trie structure when $K = 8$. The nodes of both tries are represented by the node numbers, where each root node is represented by the node number 0. Since we use the same node numbers on both tries, the node numbers of the rear trie are distinguished by an underline. The forward direction is represented by a solid line, and the backward one by a dashed line. Each leaf node on the front trie (the last node on the front trie in each path when traversing the front trie in the forward direction) is called a *separate node*. Note that the separate nodes are

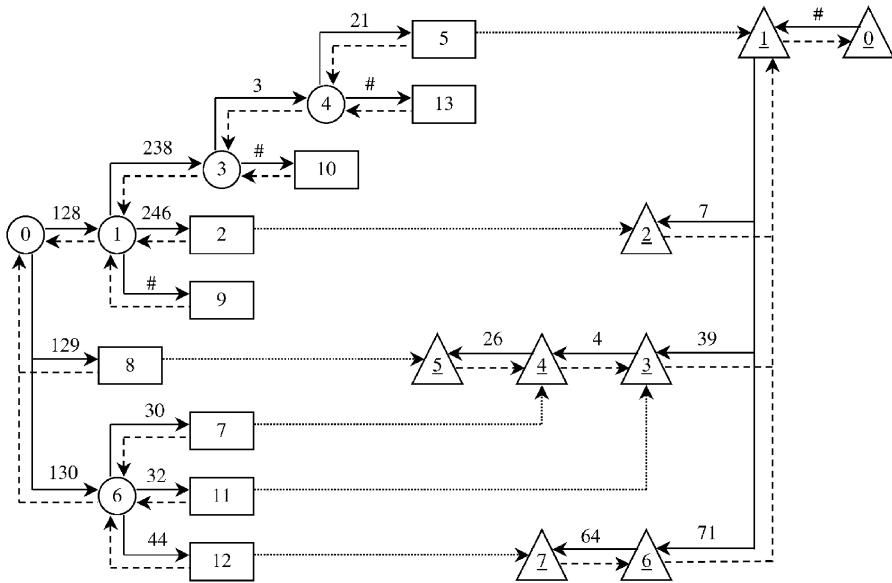


Fig. 13.29 An example of the two-trie structure when $K = 8$. (©1999 IEEE.)

the nodes that differentiate a prefix from other prefixes. The next hop of each prefix is stored at each separate node. In Figure 13.29, separate nodes (nodes 2, 5, 7, 8, 9, 10, 11, 12, 13) are represented by using rectangles, while other front nodes (nodes 0, 1, 3, 4, 6) are represented by using circles. Rear nodes (nodes 0, 1, 2, 3, 4, 5, 6, 7) are represented by using triangles.

13.9.1 IP Route Lookup Algorithm

In this subsection, we will illustrate how to do IP lookups. The $\text{IPLookup}(X)$ algorithm is shown below. The input of the function is the IP destination address X of an incoming packet, which is in the dotted decimal form. The function returns the next hop of the longest matching prefix found in the two-trie structure.

Algorithm $\text{IPLookup}(X)$

1. Let Z be the variable that stores the next hop of the longest matching prefix. Initially Z is the default next hop.
2. Start to do an IP lookup from the root node of the front trie by matching each K -bit part of the destination address X of the packet with prefixes in the two-trie structure.
3. If there is a match, the traversal is moved to the child node at the next level of the front trie.

4. Whenever a new front node is arrived at, the algorithm first looks for its child node corresponding to the symbol # (which must be the separate node). If the node is found, it means that the two-trie structure contains a longer matching prefix, so the variable Z is updated with the next hop value of this prefix retrieved from the separate node.
5. When the separate node is reached, matching continues to the rear trie by using a pointer at the separate node (shown as a dashed line in Fig. 13.29). Matching on the rear trie is done in the backward direction.
6. The algorithm stops whenever
 - (a) a mismatch is detected somewhere in the structure (in such a case, the current value of Z is returned as the next hop), or
 - (b) the traversal reaches the root node of the rear trie (no mismatch is detected). This means that the destination address X of the packet is actually stored as a prefix in the structure. The variable Z is updated with the next hop value of the prefix stored at the separate node we previously visited and returned as the output of the function.

Suppose that we would like to look up the destination address $X = \langle 130.44.64.71.\# \rangle$ from the two-trie structure shown in Figure 13.29. The lookup starts from the root node of the front trie and proceeds along the front trie by moving to the next nodes 6 and 12. Then, the traversal is transferred to the rear trie and continues through the rear nodes 7, 6, and 1 until the root node of the rear trie is reached. The algorithm stops, and the next hop stored at the separate node 12 is assigned to the variable Z and returned as the output. Note that the algorithm can be improved by keeping track of the number of parts of the address X it has already matched. Instead of traversing until reaching the root node of the rear trie, in this example the algorithm can stop when 71, which is the last part of X , is matched at the rear node 6.

Now let us change the destination address X to $\langle 130.44.64.72.\# \rangle$ and do the lookup again. When the traversal is at the rear node 6, there is no match with 72. The algorithm stops and returns the current value Z , which is the default next hop, as the output.

13.9.2 Prefix Update Algorithms

In this subsection, we will introduce the $\text{AddPrefix}(X, Y, Z)$ and $\text{DelPrefix}(X, Y)$ algorithms, which perform the addition and deletion of a prefix X of length Y bits with the next hop Z . Note that when Y is not a multiple of K bits, the prefix X is first transformed to a set of the new

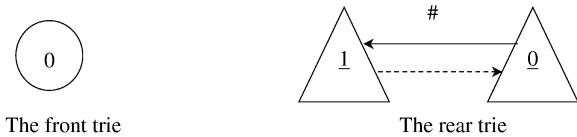


Fig. 13.30 Initialization of the front and rear tries before adding route prefix.

prefixes and then the addition and deletion are performed for each prefix that has just been generated.

Before we start to build up a routing table by adding a prefix, initially the front trie has only the root node, the node 0, while the rear trie has two nodes, 0 and 1, and pointers corresponding to the symbol #, as shown in Figure 13.30.

Algorithm AddPrefix(X, Y, Z)

1. Start by traversing nodes from the root node of the front trie to the root node of the rear trie to check whether the prefix X is in the structure.
2. If the prefix X is in the structure, the traversal will finally reach the root node of the rear trie. The algorithm retrieves the current length of X from the separate node we just visited along the path and compares it with the length Y . If Y is greater than or equal to the current length of X , the algorithm will update the length and the next hop of X , which are stored at the separate node along the path we just visited, with the values Y and Z , respectively. Then the algorithm stops.
3. If the prefix X is a new one, a mismatch is detected somewhere in the structure. There are two possible cases:
 - (a) The mismatch is detected on the front trie.
 - i. Create a new separate node for matching the K -bit part of the prefix X that starts to be different from any other prefixes in the two-trie structure. Store the length Y and the next hop Z at this node.
 - ii. Adding the remaining K -bit parts of the prefix X to the rear trie. Since the remaining parts of X can be shared with those of other prefixes in the structure, the algorithm traverses the rear trie from the root node in the forward direction and matches the remaining parts of X from the end. Some new rear nodes may be created for the parts of X that have no matches.
 - iii. Set a linkage to connect the separate node in step 3(a)i with the rear trie in step 3(a)ii.

- (b) The mismatch is detected on the rear trie.
- i. At the rear node where the mismatch is detected, let the variable M be the K -bit part of the prefix in the structure that differs from that of the prefix X . Note that M is the value that causes the occurrence of the mismatch.
 - ii. Retrieve the length and the next hop at the separate node we just visited, and copy it into the variables L and N . Change this separate node into a front node.
 - iii. Create new front nodes on the front trie for matching the parts of the prefix X the algorithm already visited on the rear trie from the first rear node visited to the rear node where the mismatch was detected.
 - iv. Create a new separate node next to the front states in step 3(b)iii for matching the value M . Set a linkage of this node with the rear trie. Store the values of L and N into this separate node.
 - v. Add the necessary parts of the new prefix X and its length and next hop into the two-trie structure by doing steps 3(a)i to 3(a)iii.
 - vi. Remove all unused nodes from the rear trie.

Suppose that we would like to add a prefix $X = \langle 130.46.2.39.\# \rangle$ of length $Y = 32$ bits and the next hop Z into the structure in Figure 13.29. The traversal starts from the root node of the front trie, matches 130, and moves to the node 6. At the node 6, a mismatch is detected on the front trie. The function creates a new separate node 14 for matching 46 and stores the length Y and the next hop Z at this node. The remaining part of X , which is $\langle 2.39.\# \rangle$, is checked with the rear trie. From the root node of the rear trie, the traversal passes the nodes 1 and 3 for matching $\#$ and 39. At the node 3, a new rear node 8 is created for matching 2. Finally, a linkage is set to connect the separate node 14 with the rear node 8. The final result of the algorithm is illustrated in Figure 13.31.

Now let us consider another example. Suppose that we would like to add a prefix $X = \langle 130.46.2.40.\# \rangle$ of length $Y = 32$ bits and the next hop Z into the structure in Figure 13.32(a). The traversal starts from the root node of the front trie, and finally a mismatch is detected at the rear node 2. Copy the value 39 into M . The length and the next hop of the prefix $\langle 130.46.2.39.\# \rangle$ stored at the separate node 3 is copied into L and N . Change the status of the node 3 from the separate node into a front node. Now create the front node 4 and the separate node 5 for matching 2 and M (which is 39). Store the values L and N into the separate node 5, and set a linkage to connect the node 5 with the rear node 1. The result is shown in Figure 13.32(b). Add the necessary parts of X by creating a new separate node 6 for matching 40, traversing the rear trie from the root node for matching $\#$, stopping at the

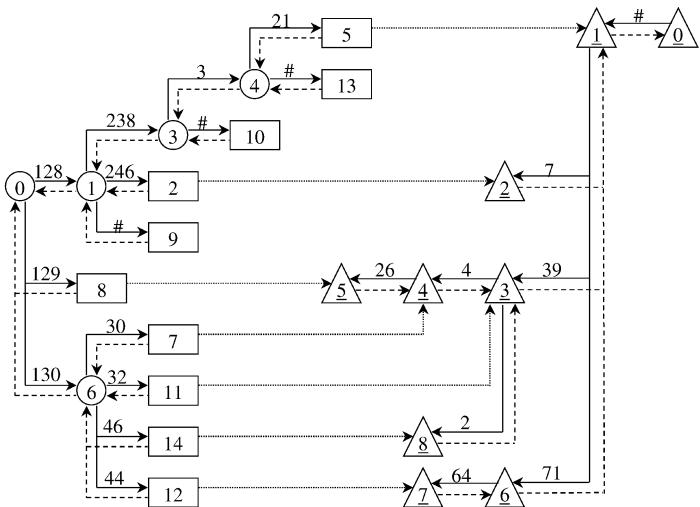


Fig. 13.31 The result of adding the prefix $X = \langle 130.46.2.39.\# \rangle$ into the two-trie structure in Figure 13.29. (©1999 IEEE.)

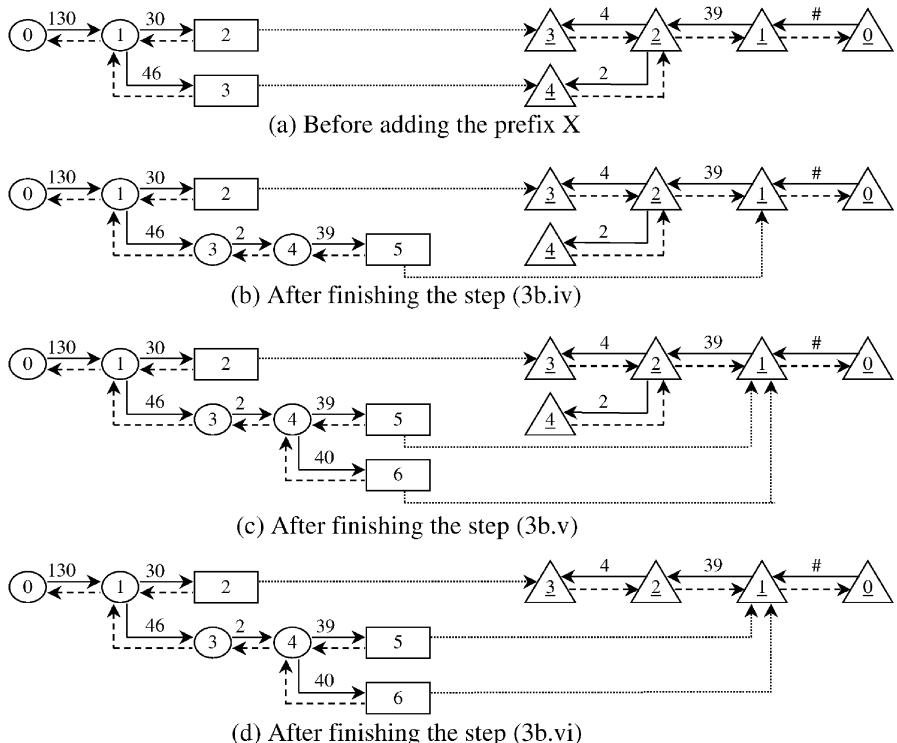


Fig. 13.32 Illustration of adding of the prefix $X = \langle 130.46.2.40.\# \rangle$. (©1999 IEEE.)

node 1, setting a linkage to connect the separate node 6 and the rear node 1. The result is shown in Figure 13.32(c). Remove the unused node 4 from the rear trie. The final result of the function is shown in Figure 13.32(d).

Algorithm DelPrefix(X, Y)

1. Start by traversing nodes from the root node of the front trie to the root node of the rear trie to check whether the prefix X is in the structure.
2. If X is not in the structure, the algorithm stops.
3. If X is in the structure, the traversal finally reaches the root node of the rear trie. The current length of X retrieved from the separate node we just visited along the path is compared with the length Y . If they are equal, the algorithm searches for the second longest prefix from the structure and replaces the length and the next hop of X with the ones of this prefix. If we cannot find the second longest prefix, we do the prefix deletion by performing the following steps:
 - (a) Remove the separate node we just visited during the traversal, and destroy the linkage that connects it with the rear trie.
 - (b) Remove unused nodes from the rear trie.
 - (c) Rearrange the two-trie structure to maintain the property that the separate nodes are the nodes that differentiate a prefix from other prefixes.

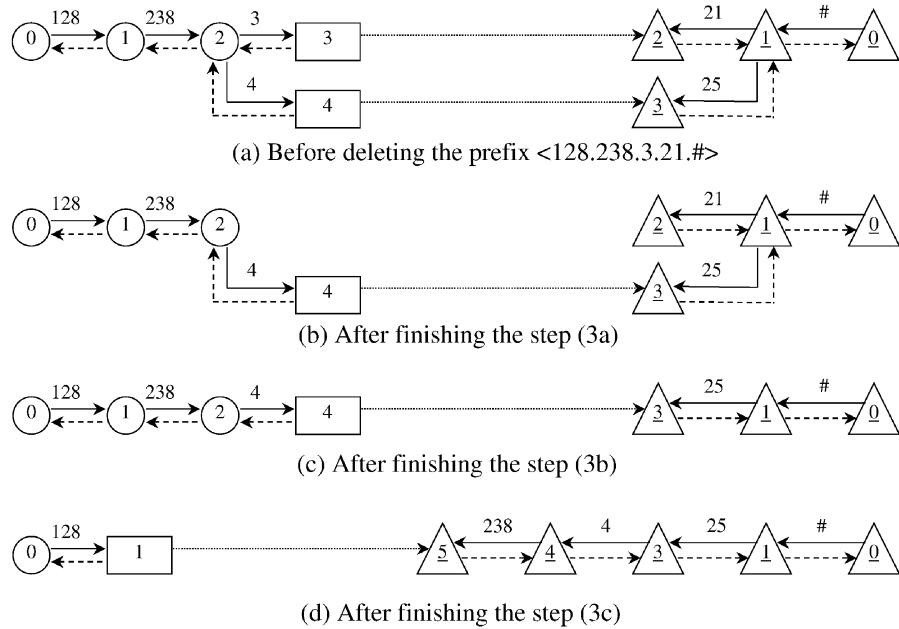


Fig. 13.33 Illustration of deleting the prefix <128.238.3.21.#>. (©1999 IEEE.)

Suppose that we would like to delete the prefix $\langle 128.238.3.21.\# \rangle$ from the structure shown in Figure 13.33(a). We do the deletion operation by removing the separate node 3 and the linkage connecting the node 2, as shown in Figure 13.33(b). The unused node 2 is removed from the rear trie, as in Figure 13.33(c). Then the two-trie structure is rearranged in that the prefix $\langle 128.238.4.25.\# \rangle$ should be differentiated by using the node 1 instead of the node 4. The final result of the operation is shown in Figure 13.33(d).

13.9.3 Performance

In this section, we will discuss the performance of our scheme based on program simulation. For performance measurement, we used a practical routing table with over 38,000 prefixes obtained from [11]. Two performance issues are considered: (1) the number of memory accesses taken for the lookup operation, and (2) the memory space required for the two-trie structure.

The worst case of the lookup operation for the 8-bit two-trie structure occurs when, for example, the IP destination address $\langle 128.238.3.21.\# \rangle$ is searched for the next hop in Figure 13.29. The lookup operation takes one memory access at the root node to transfer to the node 1. At the nodes 1, 3, and 4, it takes three memory accesses for each level: two for updating the next hop value to be that of the longer prefix, and one for transferring to the next level. At the separate node 5, the algorithm reaches the last part of the address which is 21. It takes one more memory access to retrieve the next hop value. Thus, the total number of memory accesses in the worst case is 11. The performance of the lookup operation can, however, be improved by modifying the value of K at the first level of the two-trie structure so that the root node of the front trie covers 16 bits of the prefixes. In such a case, the number of levels at the front trie is reduced by one and the total number of memory accesses in the worst case is reduced to 8.

The number of memory accesses for a lookup operation in the worst case can be reduced to 4 if we eliminate all separate nodes and put the next-hop information into the front nodes. By doing this, each entry in the front node has two more fields: one stores the next-hop information, and the other stores the pointer to the corresponding rear node. This increases the size of the front nodes and, in turn, the memory requirements of the structure; that is the price of the faster lookup operation.

To find the IP lookup performance on the average, assuming that the IP addresses are equally probable, we generated 100 million addresses and performed the lookup operations for them. For the sake of comparison, we used the 8-bit trie structure. The result is shown in Table 13.2.

A typical implementation can be used to implement the two-trie structure. The front or the rear node contains one pointer to its parent node and the entries providing the associated pointers to the nodes at the next level. The separate node stores the length and the next-hop information of each prefix. It also stores two pointers for each direction. Additionally, each node in the

TABLE 13.2 The Number of Memory Accesses for the Lookup Operation

Structure	Bits for Each Level	Average Case
Two-trie	8, 8, 8, and 8	3.6
Two-trie	1, 6, 8, and 8	1.6
Standard trie	8, 8, and 8	2.1

TABLE 13.3 Memory Requirements for the Structures after Creating the Routing Table with 38,000 Routing Entries

Structure	Bits for Each Level	Memory Requirements (Mbyte)
Two-trie	8, 8, 8, and 8	11.6
Two-trie	16, 8, and 8	11.6
Standard trie	8, 8, and 8	16.0

structure contains other necessary node information for use while performing the prefix addition and deletion.

To build up the two-trie structure, we read each IP route from the practical routing table in random order and performed the prefix addition. The result is shown in Table 13.3.

By using the two-trie structure to implement the routing table, we can reduce the memory by around 27% in comparison with the standard trie.

REFERENCES

1. J. Aoe, K. Morimoto, M. Shishibori, and K. Park, “A trie compaction algorithm for a large set of keys,” *IEEE Trans. Knowledge Data Eng.*, vol. 8, no. 3, pp. 476–490, 1996.
2. M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, “Small forwarding tables for fast routing lookups,” *Proc. ACM SIGCOMM ’97*, Cannes, France, pp. 3–14, Sep. 1997.
3. W. Doeringer, G. Karjoh, and M. Nassemi, “Routing on longest-matching prefixes,” *IEEE/ACM Trans. Networking*, vol. 4, no. 1, Feb. 1996.
4. V. Fuller, T. Li, J. Yu, and K. Varadhan, “Classless inter-domain routing (CIDR): an address assignment and aggregation strategy,” RFC 1519, Internet Engineering Task Force, Sep. 1993.
5. P. Gupta, S. Lin, and N. McKeown, “Routing lookups in hardware at memory access speeds,” *Proc. IEEEINFOCOM ’98*, San Francisco, pp. 1240–1247, Apr. 1998.
6. N. Huang, S. Zhao, J. Pan, and C. Su, “A fast IP routing lookup scheme for gigabit switching routers,” *Proc. IEEE INFOCOM ’99*, New York, Mar. 1999.
7. T. Kijkanjanarat and H. J. Chao, “Fast IP lookups using a two-trie data structure,” *Proc. IEEE Globecom ’99*, 1570–1575, 1999.

8. D. Knuth, *Fundamental Algorithms. Vol. 3: Sorting and Searching*, Addison-Wesley, 1973.
9. B. Lampson, V. Srinivasan, and G. Varghese, "IP lookups using multiway and multicolumn search," *Proc. IEEE INFOCOM '98*, San Francisco, pp. 1248–1256, Apr. 1998.
10. D. R. Morrison, "PATRICIA—practical algorithm to retrieve information coded in alphanumeric," *J. ACM*, vol.17, no. 1, pp. 1093–1102, Oct. 1968.
11. Merit Network, Inc. <http://www.merit.edu/ipma>.
12. S. Nilsson and G. Karlsson, "Fast address lookup for internet routers," *Proc. IFIP 4th Int. Conf. on Broadband Communication*, Stuttgart, Germany, pp. 11–22, Apr. 1998.
13. C. Partridge, P. Carvey, E. Burgess, I. Castineyra, T. Clarke, L. Graham, M. Hathaway, P. Herman, A. King, S. Kohlami, T. Ma, T. Mendez, W. Milliken, R. Osterlind, R. Pettyjohn, J. Rokosz, J. Seeger, M. Sollins, S. Storch, B. Tober, G. Troxel, D. Waitzman, and S. Winterble, "A fifty gigabit per second IP router," *IEEE/ACM Trans. Networking*, vol. 6, no. 3, Jun. 1998.
14. V. Srinivasan, and G. Varghese, "Faster IP lookups using controlled prefix expansion," *Proc. ACMSIGMATICS '98*, Madison, WI, pp. 1–10, Jun. 1998.
15. H.-Y. Tzeng, "Longest prefix search using compressed trees," *Proc. of IEEE GLOBECOM '98*, Sydney, Australia, Nov. 1998.
16. M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, "Scalable high-speed IP routing lookups," *Proc. ACM SIGCOMM '97*, Cannes, France, pp. 25–36, Sep. 1997.

APPENDIX

SONET AND ATM PROTOCOLS

The asynchronous transfer mode (ATM) was standardized by the International Consultative Committee for Telephone and Telegraphy (CCITT), currently called International Telecommunications Union—Telecommunication (ITU-T), as the multiplexing and switching principle for the Broadband Integrated Services Digital Network (B-ISDN). The ATM is a connection-oriented transfer mode based on statistical multiplexing techniques. It is asynchronous in the sense that the recurrence of cells containing information from an individual user is not necessarily periodic. It is capable of providing high-speed transmission with low cell loss, low delay, and low delay variation. It also provides flexibility in bandwidth allocation for heterogeneous services ranging from narrowband to wideband services (e.g., video on demand, video conferencing, video-phone, and video library).

ATM connections either are preestablished using management functions or are set up dynamically on demand using signaling, such as user-network interface (UNI) signaling and private network–network interface (PNNI) routing signaling. The former are referred to as permanent virtual connections (PVCs), while the latter are referred to as switched virtual connections (SVCs).

As shown in Figure A.1, two ATM end systems communicate through two ATM switches. The left end system sends messages generated at the application layer to the right one. The messages are first carried in IP packets,¹ which are then passed down to different layers below the IP layer. Control bytes are added at different layers to facilitate the communications through

¹In native ATM applications, messages can be directly carried by ATM cells without first being carried by IP packets.

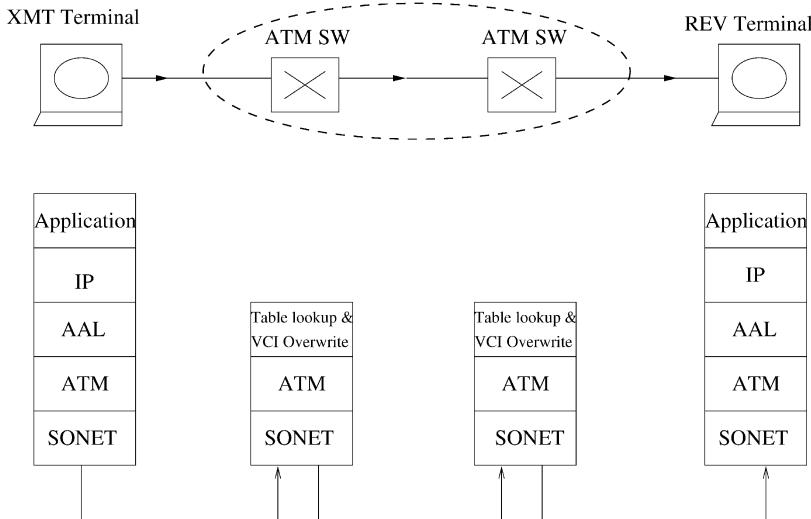


Fig. A.1 Protocol layers in the synchronous optical network (SONET) ATM network.

the ATM network. At the receiver, the control bytes are stripped off before the final messages are recovered at the application layer (if no error occurs during the transmission). Inside the ATM network, there are only two layers, the physical layer [e.g., the Synchronous Optical Network (SONET)] and the ATM layer, while at the end systems an additional layer [ATM adaptation layer, (AAL)] is added. ATM cells are routed through the ATM switches based on the routing information inside the cell header (5 bytes). The routing tables in every switch node on the path are updated either statically for PVCs or dynamically for SVCs. Once the routing tables are all updated, the connection between the two end systems is established and the left one can start to send traffic.

The detailed protocol conversion between the layers is shown in Figure A.2. A message generated at the application layer is passed down to the IP layer, where a 20-byte IP header is added. At the AAL, some AAL trailer overhead bytes are added and segmented into fixed-length data units (e.g., 48 bytes per unit). They are then passed down to the ATM layer, where a 5-byte cell header is added to every cell. A series of cells is then put into the payload of SONET frames, which repeat every 125 μ s. Here, SONET frames can be likened to trains, and their overhead bytes to train engines.² Cells are like cars in the trains. Cars in the trains may go to different destinations. As the trains arrive at a train station (i.e., an ATM switch), train engines are removed and cars are routed to different tracks (i.e., output links

²The overhead bytes are actually evenly distributed in the SONET frame. For easy explanation, they are lumped together here.

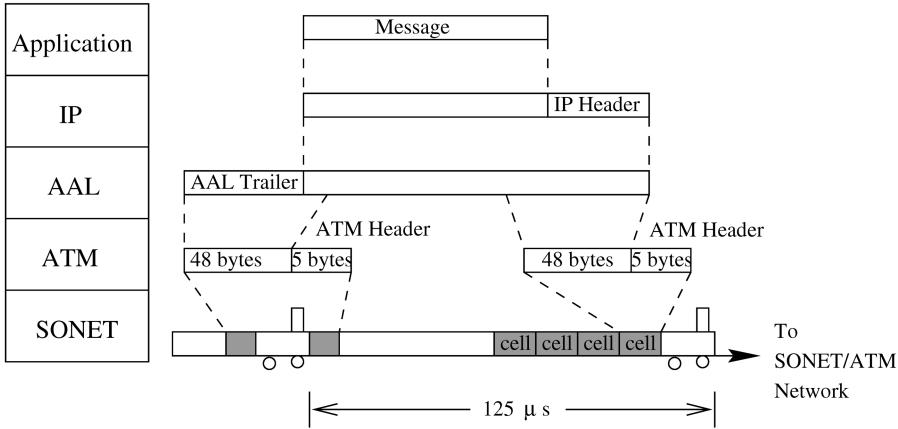


Fig. A.2 Conversion between protocol layers.

of the switch) for which the cars are destined. These cars are packed together with others into trains that go to the same train station.

Section A.1 describes the ATM protocol reference model. Section A.2 describes SONET transmission frames and the functions related to the overhead bytes. Section A.3 describes the functions performed in different sublayers of the reference model. Section A.4 describes the ATM cell format and the related functions performed in the ATM layer. Section A.5 describes different AAL types (1, 2, 3/4, and 5).

A.1 ATM PROTOCOL REFERENCE MODEL

The ATM protocol reference model shown in Figure A.3 specifies the mapping of higher-layer protocols onto AAL, ATM, and its underlying physical layer. It is composed of four layers and three planes (user, control, and management planes).

- *U-plane:* The *user* plane provides for the transfer of user application information. It contains the physical layer, the ATM layer, and multiple ATM adaptation layers required for different service users, e.g., constant-bit-rate (CBR) and variable-bit-rate (VBR) services.
- *C-plane:* The *control* plane protocols deal with call establishment and release and other connection control functions necessary for providing switched services. The C-plane structure shares the physical and ATM layers with the U-plane. It also includes AAL procedures and higher-layer signaling protocols, such as integrated local management interface (ILMI), UNI signaling, and PNNI routing protocols.

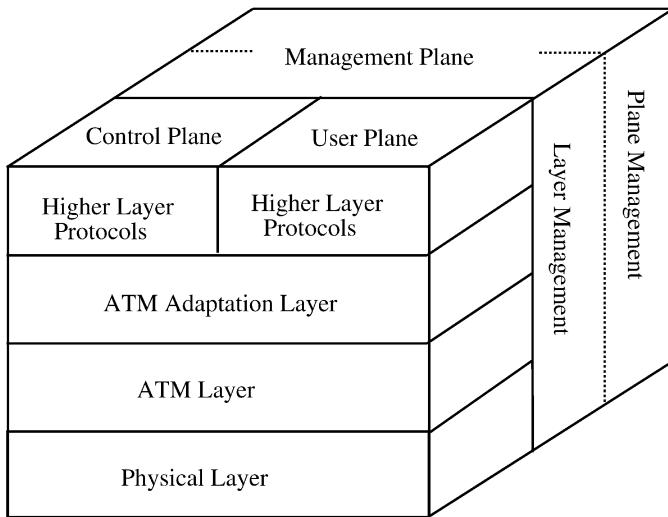


Fig. A.3 ATM protocol reference model.

- *M-plane*: The *management* plane provides management functions and the capability to exchange information between the U-plane and C-plane.

The ILMI protocol uses the *simple network management protocol* (SNMP) to provide ATM network devices with status and configuration information concerning virtual path connections (VPCs), virtual channel connections (VCCs), registered ATM addresses, and the capabilities of ATM interfaces. UNI signaling specifies the procedures for dynamically establishing, maintaining, and clearing ATM connections at the UNI. The procedures are defined in terms of messages and the information elements used to characterize the ATM connection and ensure interoperability. The PNNI protocol provides the functions to establish and clear such connections, efficiently manage the resources of the network, and allow networks to be easily configured.

A.2 SYNCHRONOUS OPTICAL NETWORK (SONET)

ATM cells can be carried on different physical layers, such as digital signal level 1 (DS1), DS3, SONET, and others. Here, we only discuss how cells are carried in SONET transmission frames. SONET is a digital transmission standard based on the synchronous rather than the plesiochronous multiplexing scheme.

A.2.1 SONET Sublayers

Figure A.4 shows a SONET end-to-end connection with some typical SONET equipment, such as SONET multiplexer (MUX) terminal, regenerator,

add/drop multiplexer (ADM), and digital cross-connect system (DCS). In practical applications, the communication is bidirectional. Let us consider an example where non-SONET transmission signals from the left are multiplexed by a SONET MUX terminal into SONET frames that are converted to optical signals. Due to attenuation and dispersion in optical fibers, the SONET bit stream needs to be *regenerated*: the optical signal is first converted to the electrical signal, which is amplified, resampled, and then

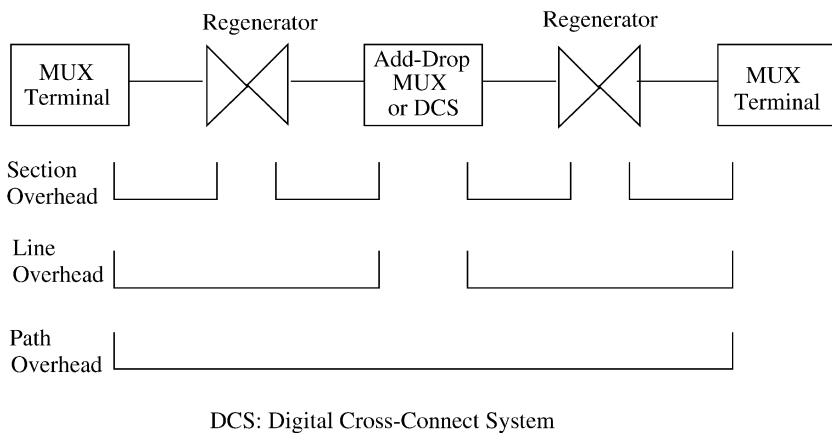
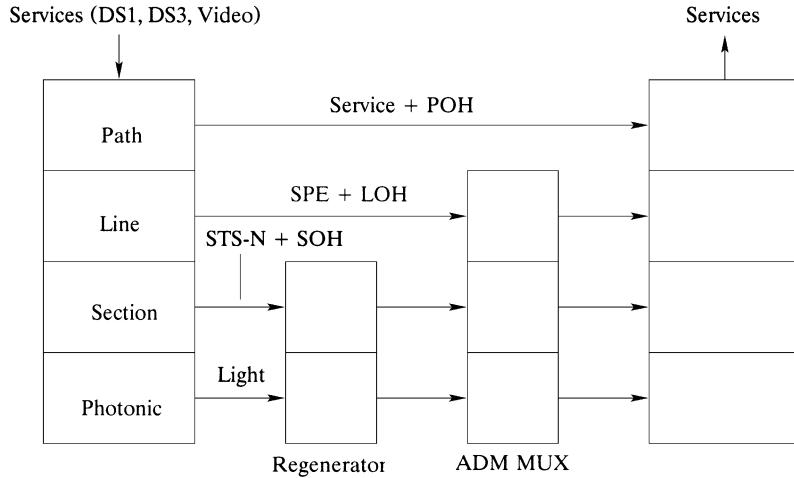


Fig. A.4 SONET end-to-end connection.



POH : Path OverHead	STS : Synchronous transport signal
LOH : Line OverHead	ADM MUX : Add and drop multiplexer
SOH : Section OverHead	DS1 : Digital signal 1
SPE : Synchronous payload envelopes	DS3 : Digital signal 3

Fig. A.5 SONET layers and associated overheads.

converted back to the optical signal. Because of the resampling, a phase-lock loop circuit is required to recover the clock from the incoming bit stream. The ADM is used to add and drop one or several low-rate tributaries, while the DCS is used for rerouting or grooming low-rate tributaries.

Section is the portion of the SONET connection between a terminal and a regenerator, or between two regenerators. *Line* is the portion of the connection between a MUX terminal and an ADM/DCS, or between ADMs/DCSs. *Path*, related to services, is a logical connection between two end terminals, where SONET frames are generated by one terminal and removed by the other. As shown in Figure A.5, each portion of the SONET connection has overhead bytes to facilitate operation, administration, and maintenance (OAM) functions in the corresponding portion of the SONET connection. For instance, messages from the service layer (e.g., DS1, DS3, or video streams) accumulate path overhead (POH), line overhead (LOH), and section overhead (SOH) as they are passed down to different layers in SONET. At the receiver end, these overheads are stripped off as the messages are passed up to the service layer.

A.2.2 STS-*N* Signals

The basic SONET transmission signal is synchronous transfer signal level 1 (STS-1), and its transmission frame is shown in Figure A.6. Each frame has 810 bytes and is organized into 9 rows of 90 bytes (transmitted from left to right, top to bottom). Frames are repeated every 125 μ s, corresponding to 8-kHz voice sampling rate. The bit rate of the STS-1 is (810 bytes)/(125 μ s), or 51.84 Mbit/s. The first three columns (27 bytes) are reserved for transport overhead, consisting of 9-byte SOH and 18-byte LOH. The remaining 87 columns form an STS-1 synchronous payload envelope (SPE), where the first column is reserved for POH.

As shown in Figure A.6, an SPE does not need to be aligned to a single STS-1 frame. In other words, it may “float” and occupy parts of two consecutive frames. There are two bytes in LOH used as a pointer to indicate the offset in bytes between the pointer and the first byte of the SPE.

When N STS-1 signals are multiplexed with byte interleaving into an STS- N signal, the STS- N frame structure is as shown in Figure A.7. The byte position in STS-1 is now “squeezed” with N bytes of information, either overhead bytes or user’s data. For instance, the first two bytes of STS-1 are A1 and A2 bytes for framing. Now, there are N bytes of A1 followed by N bytes of A2 in STS- N frames. The STS- N frames also repeat every 125 μ s, and the bit rate of the STS- N signal is N times that of the STS-1 signal.

Certain values of N have been standardized, and their corresponding bit rates are shown in Figure A.8. SONET is standardized by the American National Standards Institute (ANSI) and is deployed in North America, while Synchronous Digital Hierarchy (SDH) is standardized by ITU-T and is deployed in the other parts of the world. SONET and SDH are technically

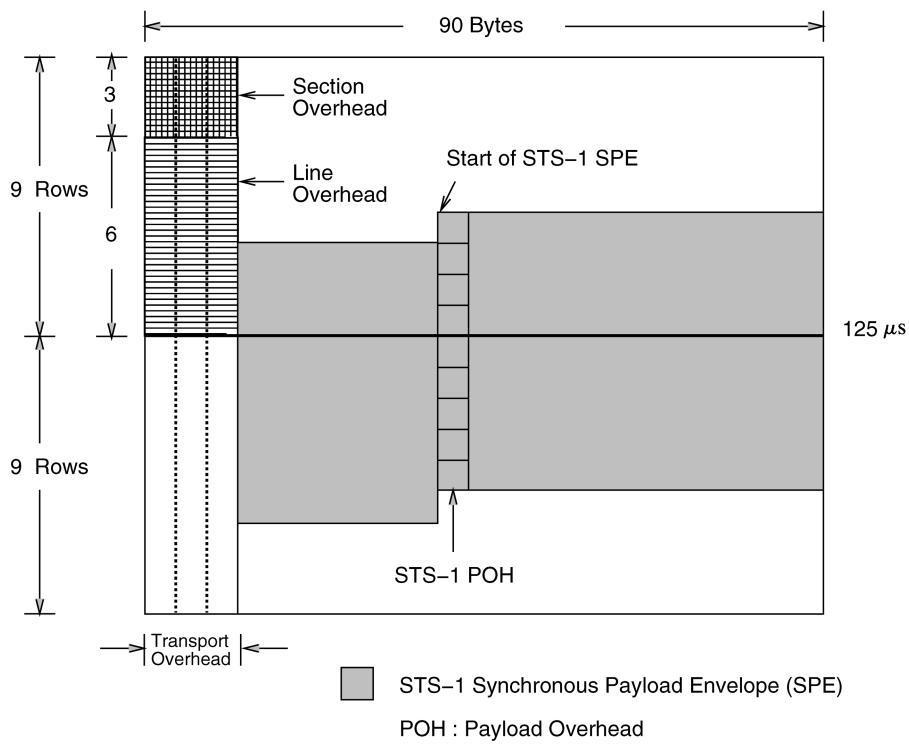


Fig. A.6 Two STS-1 (synchronous transfer signal level 1) frames.

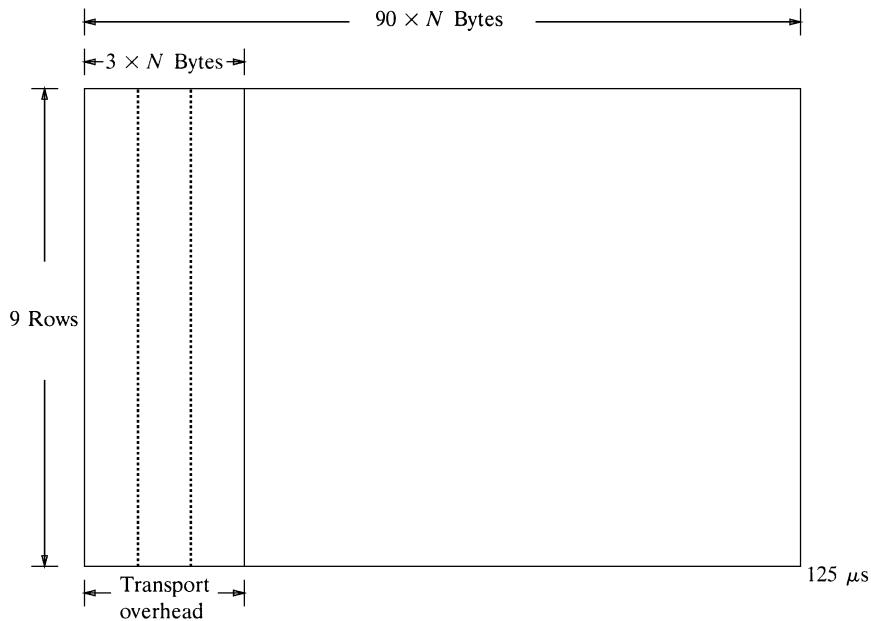


Fig. A.7 Synchronous transfer signal level N (STS- N) frames.

OC LEVEL	STS LEVEL	SDH LEVEL	LINE RATE (Mbit/sec)
OC-1	STS-1		51.840
OC-3	STS-3	STM-1	155.520
OC-12	STS-12	STM-4	622.080
OC-48	STS-48	STM-16	2488.320
OC-192	STS-192	STM-64	9953.280

OC : Optical carrier SDH: Synchronous Digital Hierarchy
STS: Synchronous Transfer Signal STM: Synchronous Transfer Mode

Fig. A.8 Standardized SONET–SDH rates.

consistent. The major difference between them is in their terminologies. Figure A.8 shows the mapping between them. Synchronous transfer mode level 1 (STM-1) is the basic transmission signal in SDH, and its bit rate is 155.52 Mbit/s. Higher-capacity STMs are formed at rates equivalent to m times this basic rate. STM capacities for $m = 4$, $m = 16$, and $m = 64$ are defined and called STM-4, STM-16, and STM-64. Their corresponding rates are shown in Figure A.8. The optical format of the STS- N signal is called optical carrier level N (OC- N).

Figure A.9 shows two examples of multiplexing 12 STS-1 signals into an STS-12 signal, one with two stages of multiplexing and one with a single stage. In order to have identical byte streams at the output of the multiplexer, it has been standardized that the output byte stream has to be the same as the one from multiplexing multiple STS-3 signals.

A.2.3 SONET Overhead Bytes

Figure A.10 shows the section, line, and path overhead bytes. A1 (11110110) and A2 (00101000) are used for frame alignment. In order to reduce the probability of having A1 and A2 bytes in the payload frame and to avoid a long stream of 0s and 1s, the entire frame except the first 9 bytes is scrambled (see Section A.2.4). The J0 byte is allocated to a section trace. This byte is used to transmit repetitively a *section access point identifier* so that a section receiver can verify its continued connection to the intended transmitter. B1 is used to monitor the error in the regenerator section. B1 is computed using bit-interleaved even parity over all the bits of the previous

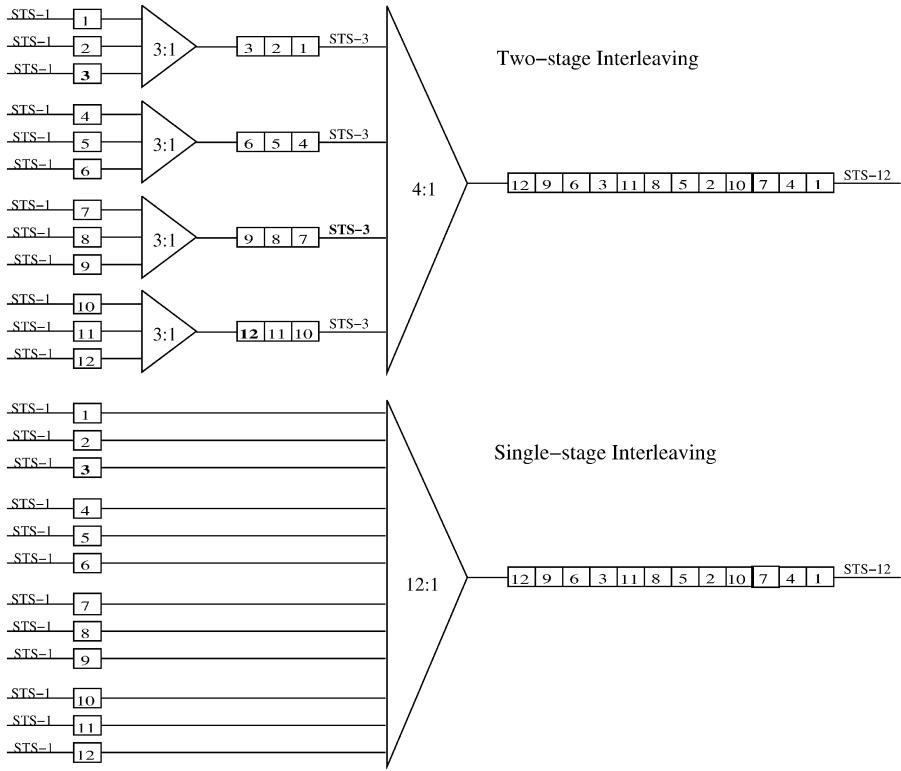
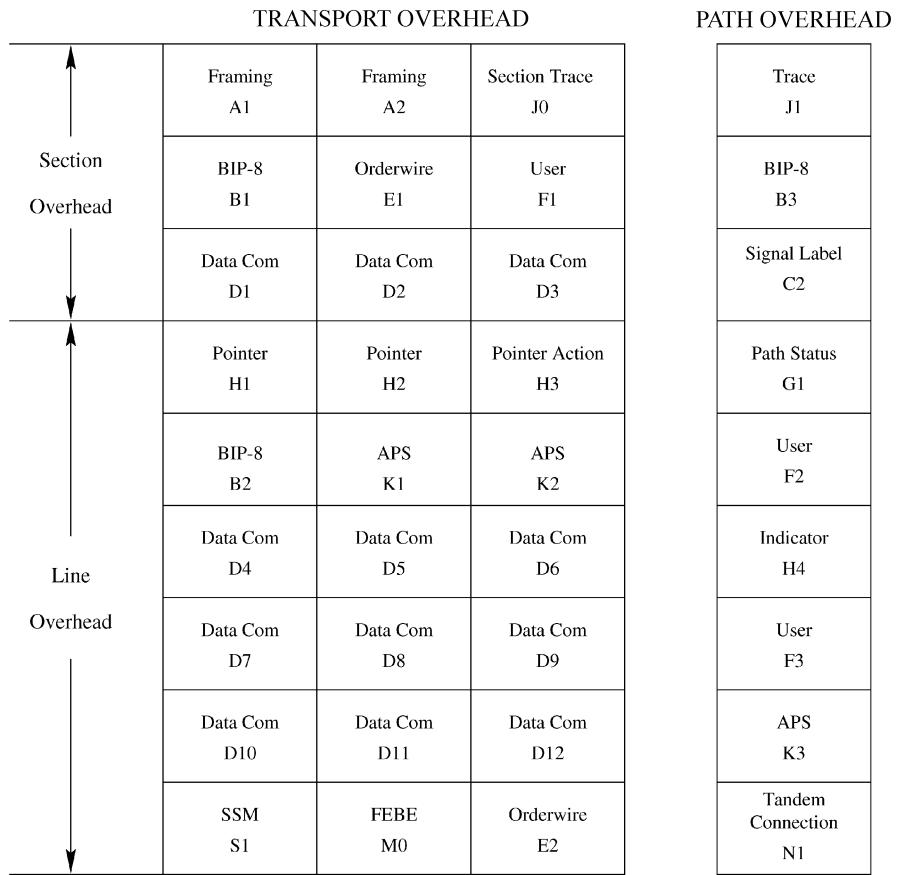


Fig. A.9 Example of byte interleaving.

STS-1 frame after scrambling and is placed in byte B1 of the current frame before scrambling. The E1 byte is allocated in the first STS-1 of an STS- N signal for a 64-kbit/s orderwire channel used for voice communication between regenerators and terminal locations. The F1 byte is allocated in the first STS-1 of an STS- N signal and is reserved for user purposes (e.g., to provide temporary data/voice channel connections for special maintenance purposes). The D1–D3 bytes are located in the first STS-1 of an STS- N signal for section data communication. These three bytes form a 192-kbit/s message-based channel for alarms, maintenance, control, monitoring, administering, and other communication needed between section-terminating equipment.

The pointer, (H1 and H2 bytes), is used to find the first byte of the SPE. When their value is zero, the SPE starts immediately following the H3 byte. H3 is used for frequency justification to compensate for clock deviations between the source and the multiplexer terminals, which may occur in some circumstances (see Section A.2.5). The B2 bytes are used to monitor bit



SSM: Synchronous Status Message

FEBE: Far End Block Error

BIP-8: Bit Interleaved Parity - 8

Fig. A.10 Section, line, and path overhead bytes.

errors at the multiplexer section. It is computed using bit-interleaved even parity over all bits of the previous STS-1 frame except for the first three rows of SOH and is placed in bytes B2 of the current frame before scrambling. The K1, K2, and M0 bytes are related to automatic protection switching (APS) operations (see Section A.2.6). The D4–D12 bytes form a 576-kbit/s channel available at the line terminating equipment. The S1 byte is allocated for the synchronous status message (SSM) function. It indicates the type of clock generating the synchronization signal. The M0 byte is allocated for the far end block error (FEBE) function, which is assigned for the *N*th STS-1 of

an STS- N signal. The E2 byte provides an orderwire channel of 64 kbit/s for voice communication between line-terminating equipment.

The J1 byte is used to transmit repetitively a path access point identifier so that a path receiving terminal can verify its continued connection to the intended transmitter. The B3 byte is allocated for a path error monitoring function. It is calculated using bit-interleaved even parity over all bits of the previous frame payload before scrambling and is placed at the B3 byte location of the current frame payload before scrambling. Bits 1–4 of the G1 byte convey the count of interleaved-bit blocks that have been detected in error (ranging from 0 to 8). This error count is obtained from comparing the calculated parity result and the B3 byte value of the incoming frame. The F2 and F3 bytes are allocated for user communication purposes between path elements and are payload-dependent. The H4 byte provides a generalized position indicator for payloads and can be payload-specific. Bits 1–4 of the K3 byte are allocated for APS signaling for protection at the path levels. The N1 byte is allocated to provide a tandem connection monitoring (TCM) function. The tandem connection sublayer is an optional sublayer that falls between the multiplex section and path layers and is application-specific.

A.2.4 Scrambling and Descrambling

Figure A.11 shows how scrambling and descrambling are done in SONET. A pseudo-random bit stream with $2^7 - 1$ bits is generated by a shift register (seven D-type flip-flops) with some feedback lines as shown in Figure A.12.

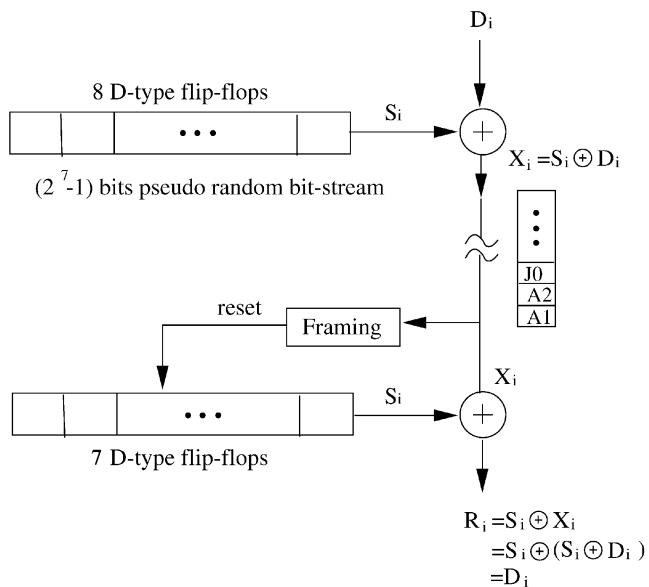


Fig. A.11 Frame-synchronous scrambling.

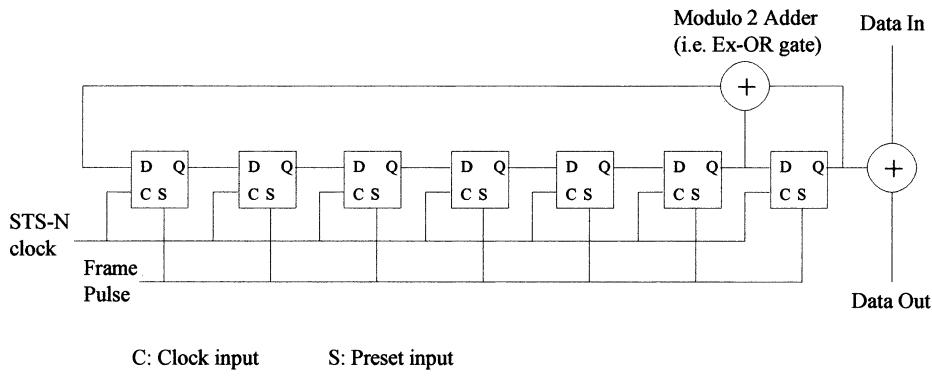


Fig. A.12 Circuit of the frame-synchronous scrambler.

The pseudo-random bit stream (S_i) is then exclusive-ORed with the data bit stream (D_i). The scrambled bitstream (X_i) is sent to the network and is descrambled with the same pseudo-random bit stream. If no error occurs in the network, the received bit stream (R_i) will be obtained by exclusive-ORing R_i with S_i . To synchronize the pseudo-random bitstreams at the transmitter and the receiver, the pseudo-random bitstream generators (the shift registers) are both reset to all ones by a frame pulse that is generated from the detection of framing bytes (A1 and A2). The scramble generating polynomial is $1 + X^6 + X^7$, resulting in the outputs of the two D-type flip-flops on the right (corresponding to X^6 and X^7) being exclusive-ORed and fed back to the input of the shift register.

A.2.5 Frequency Justification

The pointer bytes of all STS-1 signals within an STS- N signal are to align the STS-1 transport overheads in the STS- N signal as well as perform frequency justification. When multiplexing a user's data to STS-1 frames, if the user's data rate is higher than the STS-1 payload bandwidth, negative frequency justification occurs. That is, on occasion, one extra byte is added to the payload, as shown in Figure A.13. This is done by inverting the 5 bits of the H1 and H2 bytes in the D (decrement) positions for frame $n + 2$ and then decrementing the pointer by 1 in frame $n + 3$. As a result, SPE $n + 3$ starts sooner with the extra byte placed at the H3 byte location. The reason not to immediately decrement the pointer value by one is to avoid misinterpretation of the pointer value due to bit error. By inverting the five D bits of the pointer, it can tolerate up to two bit errors. On the other hand, when the user's data rate is lower than the STS-1 payload bandwidth, positive justification occurs. That is, on occasion, one byte is not made available to the payload. As shown in Figure A.14, this is done by inverting the five I (increment) bits in the H1 and H2 bytes in frame $n + 2$ and then increment-

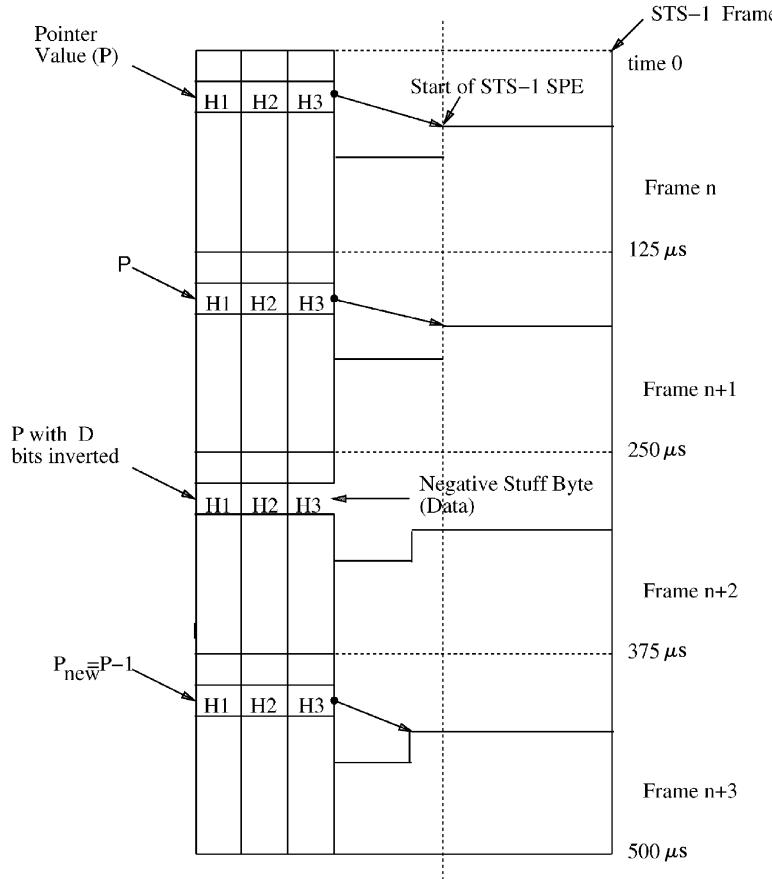


Fig. A.13 Negative STS-1 pointer justification.

ing the pointer by 1 in frame $n + 3$. As a result, SPE $n + 3$ starts later. The byte next to H3 is now left empty or “stuffed” and not allowed to be part of the SPE.

A.2.6 Automatic Protection Switching (APS)

Two types of maintenance signals are defined for the physical layer to indicate the detection and location of a transmission failure. These signals—the *alarm indication signal* (AIS) and *remote defect indication* (RDI)—are applicable at the section, line, and path layers of the physical layer. An AIS is used to alert associated termination points in the direction of transmission that a failure has been detected and alarmed. A RDI is used to alert associated termination points in the opposite direction of transmission that a defect has been detected. As shown in Figure A.15(a), when a fiber is cut or

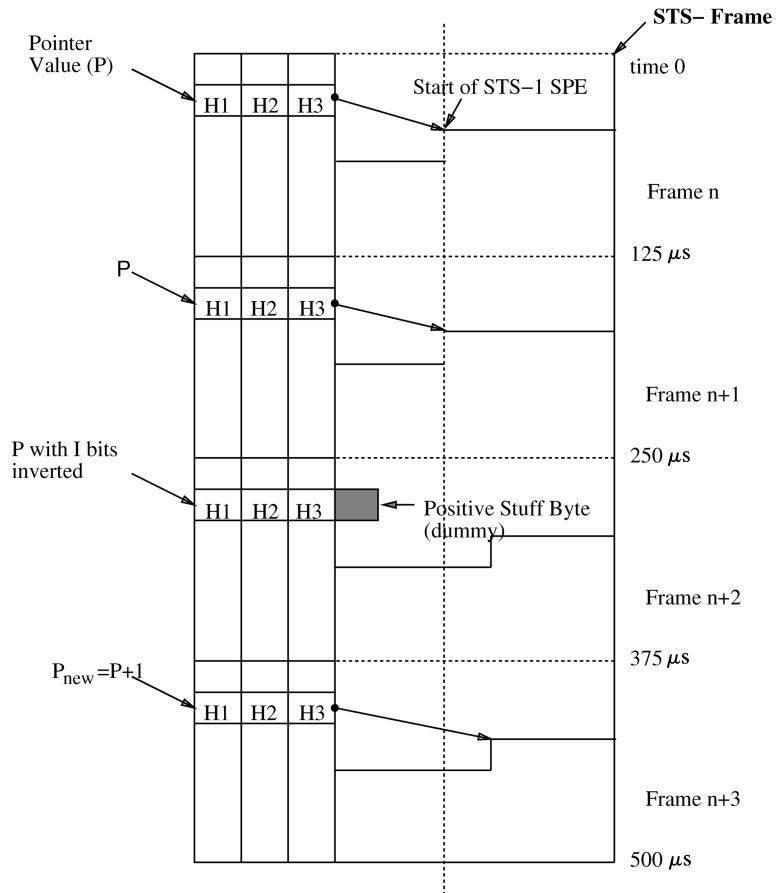


Fig. A.14 Positive STS-1 pointer justification.

the transmission equipment fails, an AIS is generated and transmitted in the downstream direction with respect to terminal A, while a RDI is generated for the upstream direction. If failures occur at both directions as shown in Figure A.15(b), the AIS and RDI are generated and transmitted in both directions.

K1 and K2 bytes are allocated for APS signaling for the protection of the SONET equipment. For instance, line AIS is generated by inserting a 111 code in positions 6, 7, and 8 of the K2 byte in the downstream direction, whereas line RDI is generated by inserting a 110 code in positions 6, 7, and 8 of the K2 byte in the upstream direction. The M0 byte is used to convey the count of interleaved bit blocks that have been detected in error in the range [0, 24]. This error count, obtained from comparing the calculated parity result and the B2 value of the incoming signal at the far end, is inserted in the M0 field and sent back. It reports to the near end line-terminating point about the error performance of its outgoing signal.

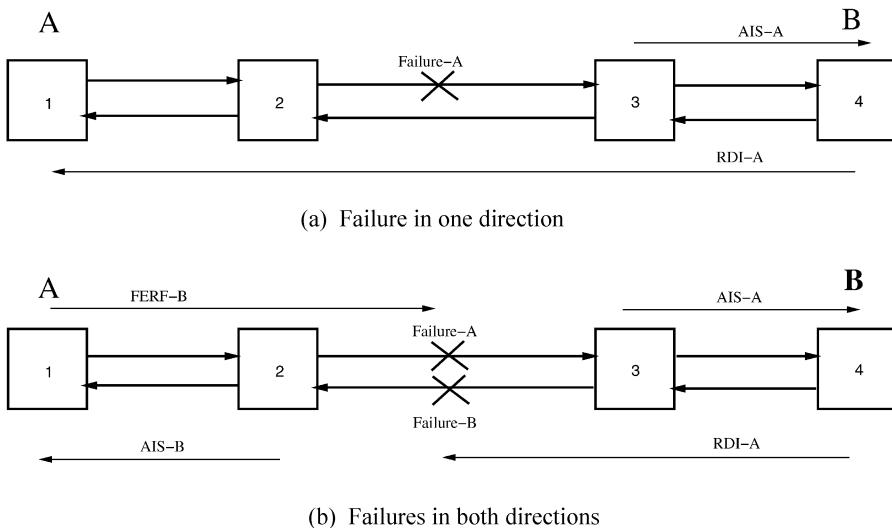


Fig. A.15 AIS (alarm indication signal) and RDI (remove defect indication).

A.2.7 STS-3 vs. STS-3c

Superrate services, such as B-ISDN ATM service, require multiples of the STS-1 rate. They are mapped into an STS-Nc SPE and transported as a concatenated STS-Nc whose constituent STS-1s are lined together in fixed phase alignment. Figure A.16 shows the comparison of the STS-3 and STS-3c: the former is multiplexed from three STS-1 signals, while the latter is

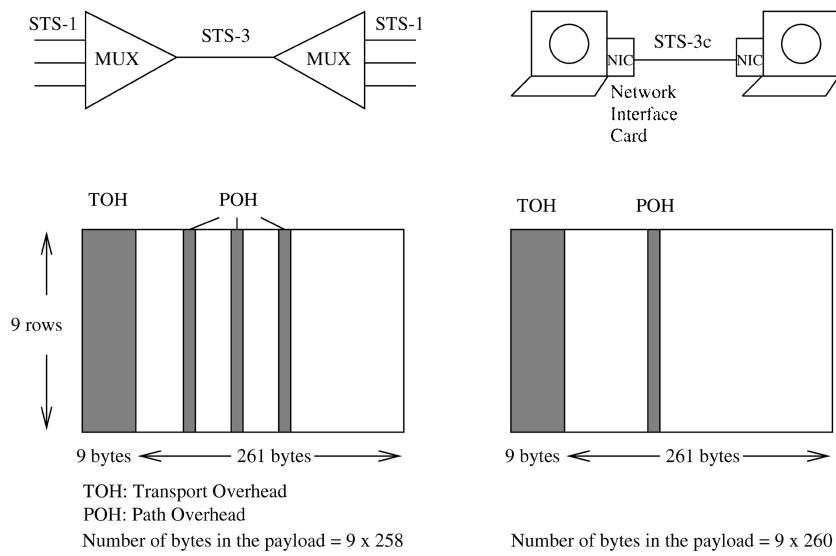


Fig. A.16 STS-3 vs. STS-3c.

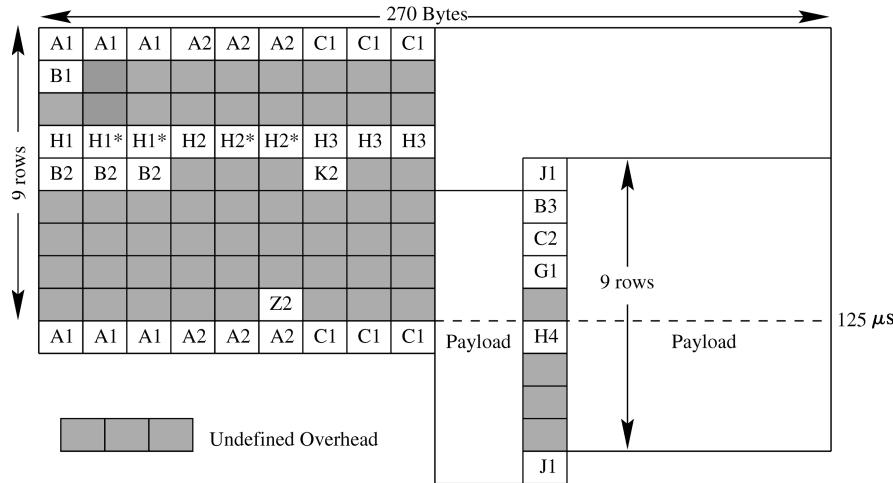


Fig. A.17 STS-3c frame format.

used to carry a single ATM cell stream. For the STS-3, there are three POHs, one for each STS-1 signal, while for the STS-3c there is only one POH. Thus, the number of bytes in the payload of the STS-3c frame is more than that of the STS-3 frame by 18 bytes.

Figure A.17 shows the frame structure of an STS-3c frame and the corresponding overhead bytes. Since there is only one SPE for each STS-3c frame, only one set of pointers (H1 and H2 bytes) in the LOH is used, while the other two sets (H1* and H2*) are not used. The entire STS-3c payload capacity (excluding SOH, LOH, and POH) can be filled with cells, yielding a transfer capacity for ATM cells of 149.760 Mbit/s ($155.52 \text{ Mbit/s} \times 260 / 270$). The remainder (5760 kbit/s) is available for the section, line, and path overhead. Because the STS-3c payload capacity (2340 bytes, or 260×9 bytes) is not an integer multiple of the cell length (53 bytes), a cell may cross a frame payload boundary.

A.2.8 OC-N Multiplexer

Figure A.18 shows how an OC-*N* signal is composed and where the functions associated with path-terminating equipment, line-terminating equipment, and section-terminating equipment are performed as the user's payload passes through the path, line, and section layers. The user's payload is first augmented with path overhead to form a SPE, and the B3 byte is calculated at the path layer. The SPE is then augmented with line overhead, and the B2 byte is calculated at the line layer. Finally, the resulting partial frame is multiplexed with byte interleaving and augmented with section overhead and the B1 byte is calculated. Note that the calculated B3, B2, and

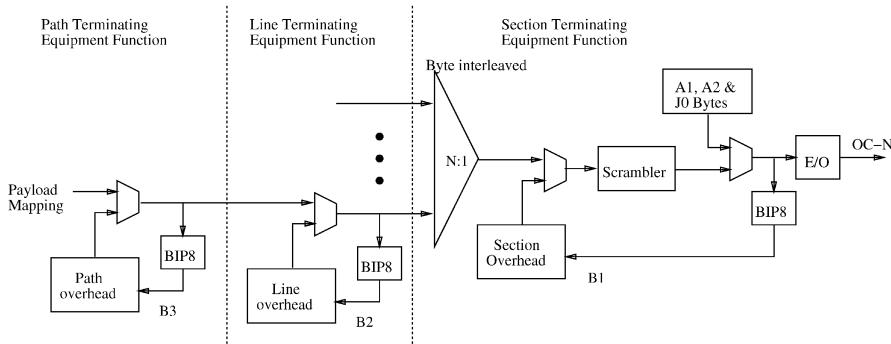


Fig. A.18 OC-N signal composition.

B1 bytes are placed at the corresponding positions in the next frame. Also note that the entire frame, except the framing bytes (A1 and A2) and the J0 byte, is scrambled.

A.3 SUBLAYER FUNCTIONS IN REFERENCE MODEL

The functions of the physical layer (U-plane) are grouped into the physical-medium-dependent (PMD) sublayer and the transmission convergence (TC) sublayer (see Fig. A.19). The PMD sublayer deals with aspects that depend on the transmission medium selected. It specifies physical medium and transmission (e.g., bit timing, line coding) characteristics and does not include framing or overhead information.

The TC sublayer deals with physical layer aspects that are independent of the transmission medium characteristics. Most of the functions constituting the TC sublayer are involved with generating and processing some overhead bytes contained in the SONET frame. On transmission, the TC sublayer maps the cells to the frame format, generates the header error control (HEC, the last byte of the ATM cell header), and sends idle cells when the ATM layer has none to send. On reception, the TC sublayer delineates individual cells in the received bit stream, and uses the HEC to detect and correct received errors.

The HEC byte is capable of single-bit error correction and multiple-bit error detection. The transmitter calculates the HEC value across the entire ATM cell header and inserts the result in the HEC field. The value is the remainder of the division (modulo 2) by the generator polynomial $x^8 + x^2 + x + 1$ of the product x^8 multiplied by the contents of the header excluding the HEC field. The remainder is added (modulo 2) to an 8-bit pattern³

³When the first four bytes of the cell header are all zeros and the remainder is also zero, modulo-2 adding the 8-bit pattern to the remainder can reduce the length of the stream of zeros.

Convergence	Convergence Sublayer (CS)	AAL
Segmentation and Reassembly	Segmentation and Reassembly Sublayer (SAR)	
Generic Flow Control Cell VPI/VCI Translation Cell Multiplex/Demultiplex Cell Rate Decoupling (with UNASSIGNED cells: ATM-Forum)		ATM
Cell Rate Decoupling (with IDLE cells: ITU-T) Header Error Check (HEC) Generation/Verification Cell Scrambling/Descrambling Cell Delineation (using HEC) Path Signal Identification Frequency Justification Frame Scrambling/Descrambling Frame Generation/Recovery	Transmission Convergence (TC)	PHY
Bit Timing Line Coding Physical Medium Dependent Scrambling/Descrambling	Physical Medium Dependent (PMD)	

Fig. A.19 Protocol reference model sublayers and functions.

(01010101) before being inserted in the HEC field. The receiver must subtract the same pattern from the 8-bit HEC before proceeding with the HEC calculation.

At the receiver two modes of operation are defined: correction mode and detection mode, as shown in Figure A.20. In *correction mode* only a single-bit error can be corrected, while *detection mode* provides for multiple-bit error detection. In detection mode all cells with detected errors in the header are discarded. When a header is examined and no error found, the receiver switches to correction mode.

Cell scrambling/descrambling permits the randomization of the cell payload to avoid continuous nonvariable bit patterns and improve the efficiency of the cell delineation algorithm.

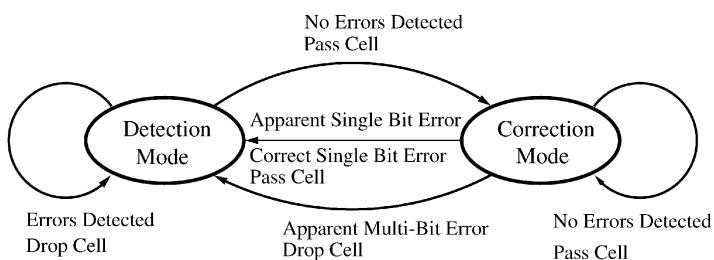


Fig. A.20 Receiver HEC bimodal operation.

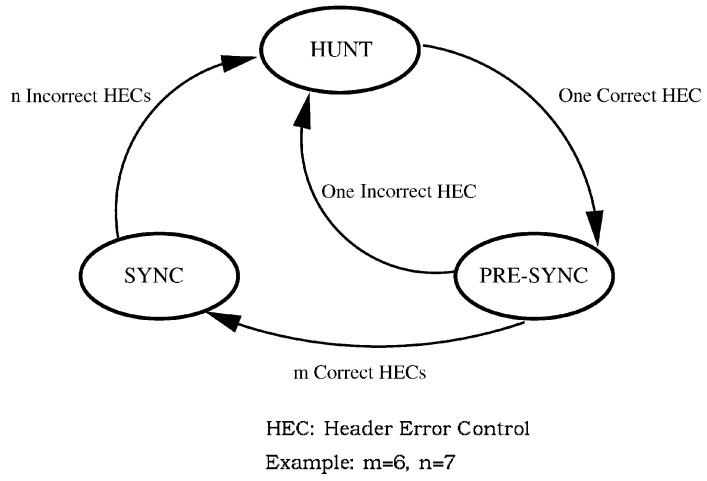


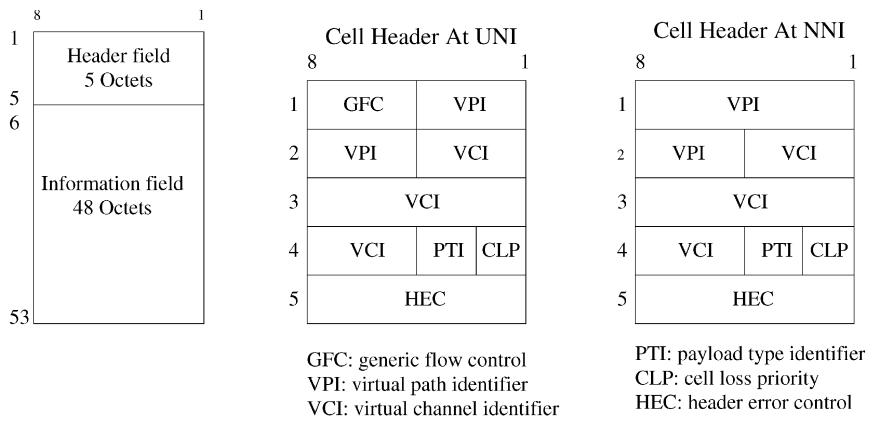
Fig. A.21 Cell delineation state diagram.

The cell delineation function identifies cell boundaries based on the HEC field. Figure A.21 shows the state diagram of the HEC cell delineation method. In the HUNT state, the delineation process is performed by checking for the correct HEC in the incoming data stream. Once a correct HEC has been detected, it is assumed that one cell header has been found, and the receiver passes to the PRE-SYNC state. In the PRE-SYNC state, the HEC checking is done cell by cell. The transition occurs from the PRE-SYNC state to the SYNC state if m consecutive correct HECs are detected, or to the HUNT state if an incorrect HEC is detected. In the SYNC state, the receiver moves to the HUNT state if n consecutive cells have an incorrect HEC. For instance, m can be 6 and n can be 7.

A.4 ASYNCHRONOUS TRANSFER MODE

The ATM layer provides for the transparent transfer of fixed-size ATM cells between communicating upper layer entities (AAL entities). This transfer occurs on a preestablished ATM connection according to a traffic contract. A traffic contract is composed of a service class, a vector of traffic parameters (e.g., peak cell rate, sustainable rate, and maximum burst size), a conformance definition, and other parameters. Each ATM end system is expected to generate traffic that conforms to these parameters. Enforcement of the traffic contract is optional at the private UNI. The public network is expected to monitor the offered load and enforce the traffic contract.

As shown in Figure A.22, an ATM cell has a 5-byte header field and a 48-byte information field. There are two types of cell headers, one for UNI

**Fig. A.22** ATM cell structure.

and one for the network-to-network interface (NNI). The UNI cell header contains the following fields: 4-bit generic flow control (GFC), 8-bit virtual path identifier (VPI), 16-bit virtual channel identifier (VCI), 3-bit payload type identifier (PTI), 1-bit cell loss priority (CLP), and 8-bit header error control (HEC). The NNI cell header does not have the GFC field, and its VPI field is 12 bits as opposed to 8 bits in the UNI cell header.

GFC can be used to provide local functions (e.g., flow control) on the customer site. It has local significance only; the value encoded in the field is not carried from end to end and will be overwritten by the ATM switches.

A.4.1 Virtual Path and Virtual Channel Identifiers

ATM has a two-level hierarchy of ATM connection, whereby a group of VCCs are usually bundled in a VPC. Within a transmission link, there can be multiple VPCs, and within each VPC there can be multiple VCCs. VPCs are logical “pipes” between any two ATM switch nodes that are not necessarily directly connected by a single physical link. Thus, it allows a distinction between physical and logical network structures and provides the flexibility to rearrange the logical structures according to the traffic requirements and end point locations. A VPC is often chosen to be a constant-rate “pipe”; it is not restricted to be so, and it can belong to any of the service categories. For instance, a corporation might establish VPCs through a network service provider to interconnect various building locations. The presence of VPCs allows for quicker establishment of switched VCCs. The VPCs also enable faster restoration.

As shown in Figure A.23, when a VPC (e.g., VPI = 3) is routed by a switch node, only the VPI will be used to look up the routing table and be replaced with a new value (VPI = 6), and its VCI values remain unchanged (VCI = 96

ATM Switch

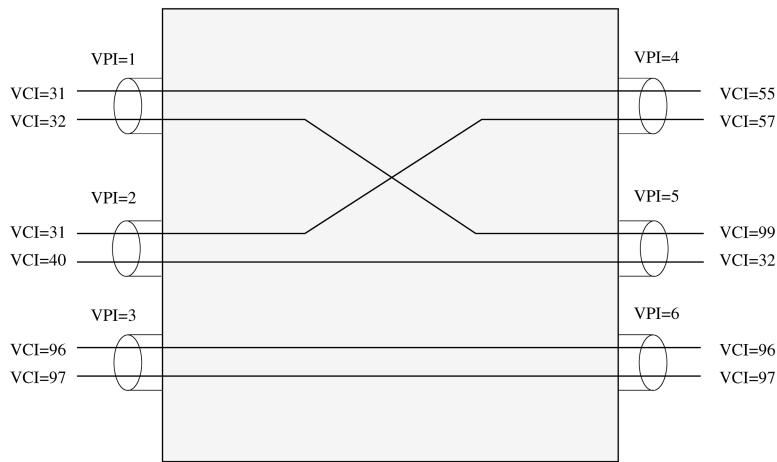


Fig. A.23 Virtual paths and virtual channels.

and 97). When a VCC (e.g., VPI = 2, VCI = 31) is routed through a switch, the combination of the VPI and VCI is used to look up the routing table and replaced with a new value (VPI = 4, VCI = 57) at the output link.

The two identifiers, together with the physical link the cells arrive from, uniquely identify connections at each ATM switch. At the UNI, there are 8 bits for the VPI, while at the NNI there are 12 bits. In general, VCI values are unique only at a particular VPI value, and VPI values are unique only in a particular physical link. The VPI/VCI has local significance only, and its value is replaced with a new value at a new physical link, which is known as *label swapping*. The reason for having the VPI/VCI change at every link is to support a large number of ATM end systems. For instance, the number of hosts supported by IPv4 is limited to 2^{32} , whereas in ATM the number is limited to 2^{24} at UNI or 2^{28} at NNI, per transmission link.

A.4.2 Payload Type Identifier

The main purpose of the payload type identifier (PTI) is to discriminate user cells (i.e., cells carrying user information) from nonuser cells. As shown in Figure A.24, PTI values of 0 to 3 indicate user cells. PTI values of 2 and 3 indicate that congestion has been experienced in the network. PTI values of 4 and 5 are used for VCC level management functions (F5 flow), where OAM cells of F5 flow have the same VPI/VCI value as the user data cells transported by the VCC. A PTI value of 4 is used for identifying OAM cells communicated within the bounds of a VCC segment (i.e., a single link segment across the UNI), while a PTI value of 5 is used for identifying

PTI	Interpretation
000	User data cell, congestion not experienced, SDU-type=0
001	User data cell, congestion not experienced, SDU-type=1
010	User data cell, congestion experienced, SDU-type=0
011	User data cell, congestion experienced, SDU-type=1
100	Segment OAM F5 flow related cell
101	End-to-end OAM F5 flow related cell
110	Resource management cell
111	Reserved for future functions

Fig. A.24 Payload type identifier (PTI) encoding.

end-to-end OAM cells. A PTI value of 6 indicates the resource management (RM) cell.

A congestion flow control scheme used in the ATM network is *explicit forward congestion notification*. When a network node detects congestion on the link, it will mark the *congestion-experienced* bit in the PTI of user data cells passing through the congested link (i.e., changing the pattern 000 to 010, or 001 to 011). Once a cell is marked congestion-experienced at a node, it cannot be modified back to congestion-not-experienced by any downstream node along the path to the destination node. The receiver can set a congestion indication bit in the RM cells to instruct the sender to slow down, increase, or maintain the rate. The RM cells are normally transmitted from the sender to the receiver every certain number of data cells it sends.

A.4.3 Cell Loss Priority

The CLP field may be used for loss priority indication by an ATM end system and for selective cell discarding in network equipment. This bit in the ATM cell header indicates two levels of priority for ATM cells. Cells with CLP = 0 are higher priority than those with CLP = 1. Cells with CLP = 1 may be discarded during periods of congestion to preserve the loss rate of the cells with CLP = 0.

A.4.4 Predefined Header Field Values

Figure A.25 shows some predefined header field values. The cell rate decoupling function at the sending entity adds unassigned cells to the assigned cell stream (cells with valid payload) to be transmitted. In other words, it transforms a noncontinuous stream of assigned cells into a continuous stream

Use	Value ^{1,2,3,4}			
	Octet 1	Octet 2	Octet 3	Octet 4
Unassigned cell indication	00000000	00000000	00000000	0000xxx0
Meta-signaling (default) ^{5,7}	00000000	00000000	00000000	00010a0c
Meta-signaling ^{6,7}	00000000	0000yyyy	00000000	00010a0c
General Broadcast signaling (default) ⁵	00000000	00000000	00000000	00100aac
General Broadcast signaling ⁶	00000000	0000yyyy	00000000	00100aac
Point-to-Point signaling (default) ⁵	00000000	00000000	00000000	00100aac
Point-to-Point signaling ⁶	00000000	0000yyyy	00000000	00100aac
Invalid Pattern	xxxx0000	00000000	00000000	0000xxx1
Segment OAM F4 flow cell ⁷	0000aaaa	aaaa0000	00000000	00110a0a
End-to-End OAM F4 flow cell ⁷	0000aaaa	aaaa0000	00000000	01000a0a

1: "a" indicates that the bit is available for use by the appropriate ATM layer function

2: "x" indicates "don't care" bits

3: "y" indicates VPI value other than 00000000

4: "c" indicates that the originating signaling entity shall set the CLP bit to 0. The network may change the value of the CLP bit.

5: Reserved for signaling with the local exchange

6: Reserved for signaling with other signaling entities (e.g., other users or remote networks)

7: The transmitting ATM entity shall set bit 2 of octet 4 to 0. The receiving ATM entity shall ignore bit 2 of octet 4.

Fig. A.25 Predefined header field values.

of assigned and unassigned cells. At the receiving entity, the opposite operation is performed for both unassigned and invalid cells. The rate at which the unassigned cells are inserted or extracted depends on the bit rate (rate variation) of the assigned cell and/or the physical layer transmission rate.

Meta-signaling cells are used by the meta-signaling protocol for establishing and releasing SVCs. For PVCs, meta-signaling is not used.

The virtual path connection (VPC) operation flow (F4 flow) is carried via specially designated OAM cells. The OAM cells of the F4 flow have the same VPI value as the user-data cells transported by the VPC but are identified by two unique preassigned virtual channels within this VPC. At the UNI, the virtual channel identified by a VCI value 3 is used for virtual-path level management functions between ATM nodes on both sides of the UNI (i.e., single virtual-path link segments), while the virtual channel identified by a VCI value 4 can be used for virtual path level end-to-end management functions.

A.5 ATM ADAPTATION LAYER

The AAL performs the functions necessary to adapt the capabilities provided by the ATM layer to the needs of higher-layer applications. Since the ATM layer provides an indistinguishable service, the AAL is capable of providing

Attribute	Service Class			
	Class A	Class B	Class C	Class D
Timing relation between source and destination	Required		Not Required	
Bit Rate	Constant	Variable		
Connection Mode	Connection-Oriented		Connectionless	
AAL (S)	AAL1	AAL2	AAL3/4 or AAL5	AAL3/4 or AAL5
Example(s)	DS1,E1, nx64 kbps emulation	Packet Video, Audio	Frame Relay, X.25	IP, SMDS

Class A – constant bit-rate (CBR) service with end-to-end timing, connection-oriented

Class B – variable bit-rate (VBR) service with end-to-end timing, connection-oriented

Class C – variable bit-rate (VBR) service with no timing requested, connection-oriented

Class D – variable bit-rate (VBR) service with no timing requested, connectionless

Fig. A.26 ITU ATM/B-ISDN service classes.

different functions for different service classes. For instance, VBR users may require such functions as protocol data unit (PDU) delineation, bit error detection and correction, and cell loss detection. CBR users typically require source clock frequency recovery, and detection and possible replacement of lost cells. These services are classified by ITU-T based on three parameters: timing relationship between source and destination, bit rate, and connection mode. Four classes have been defined and shown in Figure A.26.

- *Class A:* This class corresponds to CBR, connection-oriented services with a timing relationship between source and destination. Typical services of this class are voice, circuit emulation, and CBR video.
- *Class B:* This class corresponds to VBR (variable bit rate) connection-oriented services with a time relationship between source and destination. Packet video is a typical example of this service class.
- *Class C:* This class corresponds to VBR connection-oriented services with no timing relationship between source and destination. X.25 and frame relay are typical examples of this service class.
- *Class D:* This class corresponds to VBR connectionless services with no timing relationship between source and destination. IP data transfer is a typical example of this type of service.

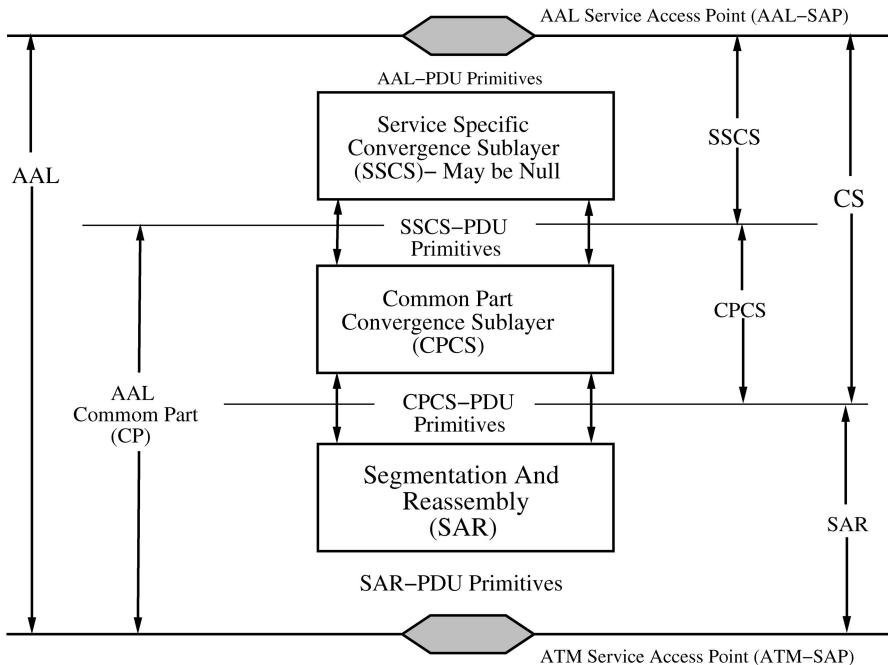


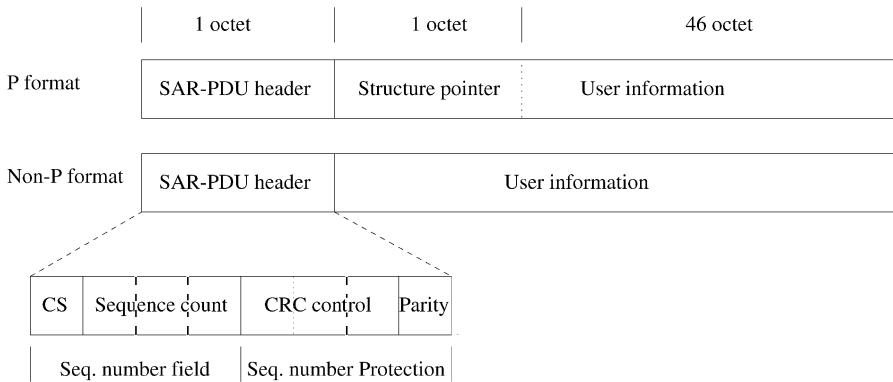
Fig. A.27 Generic AAL protocol sublayer model.

As shown in Figure A.27, the AAL is subdivided into two sublayers: the segmentation and reassembly (SAR) sublayer and the convergence sublayer (CS). The SAR sublayer performs the segmentation of user information into the ATM cell payloads. The CS maps the specific user requirements onto the ATM transport network. The functions performed at the CS differ for each of the services, whereas the SAR sublayer provides the same functions to all the services.

The CS is further divided into two parts: the common part convergence sublayer (CPCS), which is common to all users of AAL services, and the service-specific convergence sublayer (SSCS), which depends on the characteristics of the user's traffic. Figure A.27 shows how these sublayers SAR, CPCS, and SSCS are related to each other. The SSCS can be null, meaning that it need not be implemented, whereas the CPCS is always present. It is apparent that the protocol functionality performed at the SAR and CPCS is common to all AAL users.

A.5.1 AAL Type 1

AAL1 is used for CBR services that require tight delay and jitter control between the two end systems, for example, emulated $n \times 64$ kbit/s channels or T1 or E1 facilities. As shown in Figure A.28, the 47-byte SAR-PDU

**Fig. A.28** AAL1 SAR-PDU format.

payload used by CS has two formats, called P and non-P. In the non-P format, the entire SAR PDU is filled with user information.

The P-format is used in the structured data transfer mode, where a structured data set, a byte stream, is transferred between source and destination. The structured data set is pointed at by a pointer that is placed at the first byte of the 47-byte payload, and thus only 46 bytes are used to carry the user's information. The pointer is carried by even-numbered (0, 2, 4, 6) SAR PDUs, where the convergence sublayer indicator (CSI) is set to 1. Since the pointer is transferred in every two PDUs, it needs to point to any byte in the structured data set, that is, up to 93 bytes (46 + 47). Thus, 7 bits are used in the one-byte pointer to address the first byte of an $n \times 64$ kbit/s structure.

The first byte of the SAR PDU consists of a 1-bit CSI, a 3-bit sequence count, a 3-bit cyclic redundancy check (CRC), and a parity bit. The CSI bit carries the CS indication. The sequence count carries the sequence number of the SAR PDUs (0 to 7). The CSI bit and the sequence count are protected by a 3-bit CRC. The resulting 7-bit field is protected by an even-parity check bit. The CSI and the sequence count values are provided by the CS sublayer. Such a 4-bit sequence number protection field is able to correct single-bit errors and to detect multiple-bit errors. By using the sequence count, the receiver will be able to detect any cell loss if the number of lost cells is not an integer multiple of 16.

The AAL1 convergence sublayer provides two methods to support asynchronous CBR services where the clocks are not locked to a network clock. The two methods are *adaptive clock* and *synchronous residual time stamp* (SRTS). With the SRTS technique, an accurate reference network is supposed to be available at both transmitter and receiver. The timing information about the difference between the source clock and the network rate is conveyed by the CSI bit. The difference is bounded to 4-bit precision, and thus the source clock's stability tolerance is 2 parts per million (ppm). The 4

bits of timing information are carried in the CSI field of four SAR PDUs with an odd sequence count (1, 3, 5, 7). The receiver can thus regenerate the source clock rate with required accuracy by using the CSI field.

A.5.2 AAL Type 2

AAL2 is recently standardized and is designed for low-bit-rate, delay-sensitive applications that generate short, variable-length packets. A motivating application for AAL2 was low-bit-rate voice, where the delay to fill the payload of an ATM cell with the encoded speech from a single voice source would have degraded performance because of its large assembly delay. Thus, a key attribute of AAL2 is the ability to multiplex higher-layer packets from different native connections, called *logical link connections* (LLCs) (up to 255), onto a single ATM virtual channel connection (VCC) without regard to the cell boundaries. In other words, AAL2 does not require that each encapsulated packet fit within a single ATM payload, but rather allows packets to span across payloads. A connection identification (CID) field is used in the packet header to identify the LLC to which a packet belongs. A length indicator (LI) field is used to identify the boundaries of variable-length LLC packets.

Figure A.29 shows the AAL2 SAR-PDU format, where the start field (STF) is located at the beginning of each ATM cell and the packet header

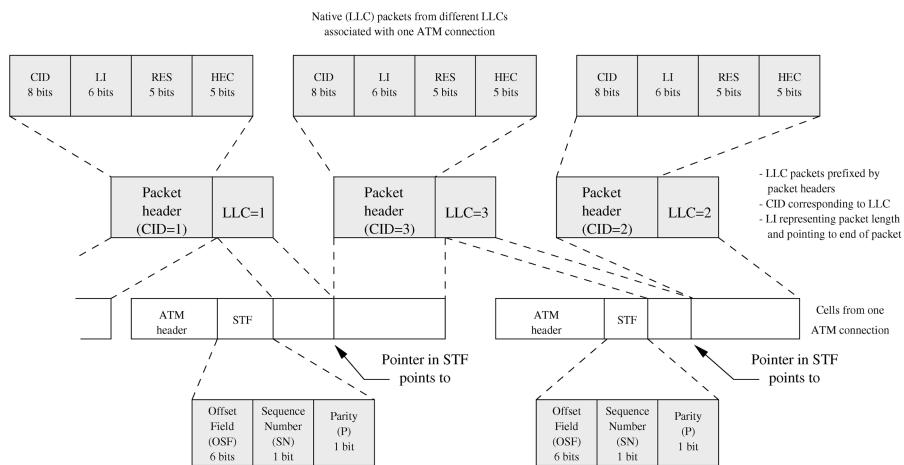


Fig. A.29 AAL2 SAR-PDU format.

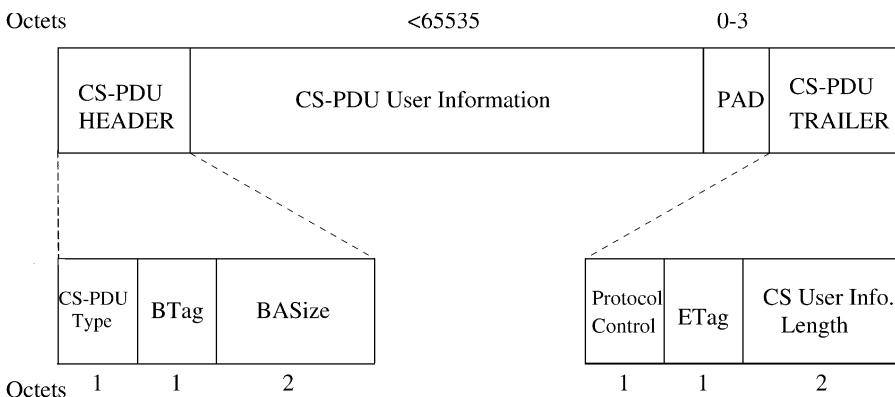
precedes each native packet. The packet header is 3 bytes long. The CID field is 8 bits long and identifies the LLC for the packet. The LI field comprises 6 bits and indicates the length of the LLC packet. When the LI points beyond the end of the current ATM cell, the packet is split between cells. The HEC field comprises 5 bits and provides error detection over the packet header. Five bits are reserved (RES) for future use.

In the STF, the offset field (OSF) is 6 bits in length. It indicates the remaining length of the packet that (possibly) started in the preceding cell from this ATM connection and is continuing in the current cell. Thus, the OSF points to the start of the first new packet and provides immediate recovery of the packet boundary after an event causing loss of packet delineation. The 1-bit sequence number (SN) field provides a modulo-2 sequence numbering of cells. The one parity (P) bit provides odd parity and covers the STF.

It may be necessary to transmit a partially filled ATM cell to limit packet emission delay. In such a case, the remainder of the cell is padded with all zero bytes. A cell whose payload contains only the STF and 47 padding bytes can also be transmitted to meet other needs, such as serving a keep-alive function and satisfying a traffic contract.

A.5.3 AAL Type 3/4

AAL3 was designed for Class C services, and AAL4 for Class D services. During the standardization process, the two AALs were merged and are now the same. The CPCS of AAL type 3/4 plays the role of transporting



CS-PDU Type=under study; set to zero for connectionless service

BTAG=sequence no. for CS-PDUs=ETag

BASize=CS-PDU payload length

Protocol Control=under study; set to zero for connectionless service

Fig. A.30 Structure of the AAL3/4 CPCS PDU.

variable-length information units through the SAR sublayer. Figure A.30 shows the structure of the AAL 3/4 CPCS PDU. The CPCS-PDU header includes the fields *common part identifier* (CPI), *beginning tag* (BTA), and *buffer allocation size* (BSA), whereas the trailer includes the fields *alignment* (AL), *ending tag* (ETA), and *length* (LEN). CPI is used to interpret the subsequent fields in the CPCS-PDU header and trailer, for example the counting units of the subsequent fields BAS and LEN. BTA and ETA are equal in the same CPCS PDU. Different bytes are used in general for different CS PDUs, and the receiver checks the equality of BTA and ETA. BAS indicates to the receiver the number of bytes required to store the whole CPCS PDU. AL is used to make the trailer a 4-byte field, and LEN indicates the actual content of the PDU's payload, whose length is up to 65,535 bytes. A padding field (PAD) is also used to make the payload an integer multiple of 4 bytes, which could simplify the receiver design. The current specification of CPI is limited to the interpretation just described for the BAS and LEN fields.

Figure A.31 shows the structure of the AAL 3/4 PDU. The segment type (ST) is a 2-bit field that indicates a SAR-PDU as containing the beginning of a message (BOM), the continuation of a message (COM), the end of a message (EOM), or a single-segment message (SSM). The 4-bit sequence number (SN) field is used to number the SAR PDUs modulo 16 (i.e., 0 to 15).

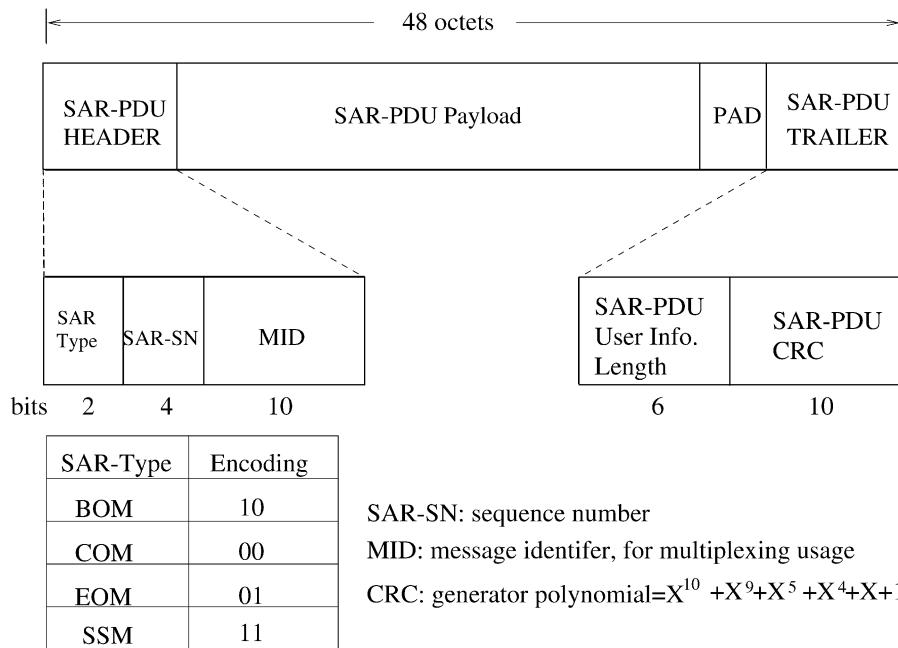


Fig. A.31 Structure of the AAL3/4 SAR PDU.

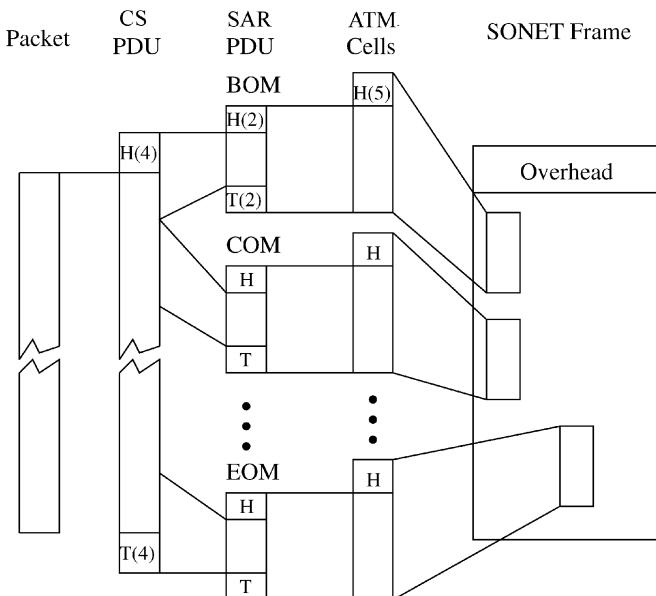


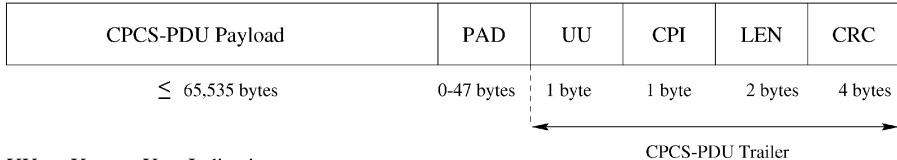
Fig. A.32 Packet-cell conversion (AAL3/4).

The multiplexing identification (MID) field is 10 bits long and is used to multiplex different connections into a single ATM connection. All SAR PDUs of a particular connection are assigned the same MID value. Using this field, it is possible to interleave and reassemble SAR PDUs of different sources that are transported using the same VPI/VCI. Cells of different sources are distinguished by carrying different MID values. The 6-bit user information length indicates the number of bytes in the SAR PDU, and its value ranges from 4 to 44. A 10-bit CRC is used over the entire SAR PDU to detect bit errors. Figure A.32 shows how a data packet is converted to ATM cells (using AAL3/4) that are then embedded in SDH frames.

A.5.4 AAL Type 5

AAL5 is the most widely used AAL type to date. For instance, AAL5 is the adaptation layer for ILMI, UNI signaling, PNNI signaling, and for IP packets.

Figure A.33 shows the structure of AAL5 CPCS PDU. Its payload length can be from 1 to 65,535 bytes. A PAD field, ranging 0 to 47 bytes, is added to make the CPCS-PDU length an integer multiple of 48 bytes. Thus, it can completely fill in the SAR-PDU payloads. CPCS *user-to-user indication* (UU) is one byte long and may be used to communicate between two AAL5 entities. The common part indicator (CPI), also one byte long, is not defined



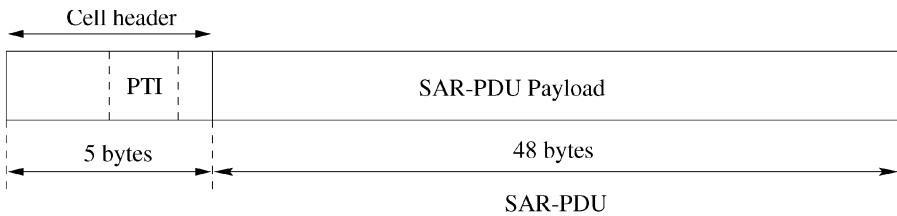
UU: User-to-User Indication

CPI: Common Part Indicator

LEN: Length

CRC: Cyclic Redundancy Check

Fig. A.33 AAL5 CPCS-PDU format.



PTI: Payload Type Identifier

Fig. A.34 AAL5 SAR-PDU format.

yet and must be set to 0. LEN indicates the actual length of the CPCS-PDU payload, so as to identify the PAD size. A 32-bit CRC, following the FDDI standards, is used to detect bit errors in the CPCS PDU. The CPCS PDU is segmented into 48-byte SAR PDUs and passed to the ATM layer, where 5-byte cell headers are inserted to the front of the data units, as shown in Figure A.34.

The main reasons that AAL5 is the most widely used are its high utilization (the entire 48-byte payload is used to carry user's information) and small processing overhead (one CRC calculation for the entire CPCS PDU). To have all 48 bytes carry user's information, the ATM layer is required to indicate the last cell of a data packet to facilitate packet reassembly, which in turn violates the protocol stack. When a cell's PTI is 000, it can be either a beginning of a message (BOM) or a continuation of a message (COM). When it is 001, the cell can be either an end of a message (EOM) or a single-segment message (SSM). A receiver will start to reassemble arriving cells when it receives the very first cell until it receives a cell with the PTI set to 001. Once an EOM or SSM is received, the receiver will start the reassembly process for the following packet.

REFERENCES

1. ATM Forum, "User-Network Interface Specification Version 3.1," ATM Forum Technical Committee, Sep. 1996.
2. ITU-T Recommendation G.707, "Network node interface for the synchronous digital hierarchy (SDH)," Geneva, Nov. 1995.
3. ITU-T Recommendation I.432, "B-ISDN User Network Interface Specification," Geneva, Aug. 1996.
4. R. Onvural, *Asynchronous Transfer Mode Networks, Performance Issues*, Artech House, Norwood, MA, 1995.
5. ITU-T Recommendation I.361, "B-ISDN ATM Layer Specification," Geneva, Nov. 1995.
6. ITU-T Recommendation I.363, "B-ISDN ATM Adaptation Layer Specification," Geneva, Mar. 1993.
7. ITU-T Recommendation I.363.1, "B-ISDN ATM Adaptation Layer Specification: Type 1 AAL," Geneva, Aug. 1996.
8. ITU-T Recommendation I.363.2, "B-ISDN ATM Adaptation Layer Specification: Type 2 AAL," Geneva, Sep. 1997.
9. J. H. Baldwin et al., "A new ATM adaptation layer for small packet encapsulation," *Bell Labs. Tech. J.*, pp. 111–131, Spring 1997.
10. ITU-T Recommendation I.363.3, "B-ISDN ATM Adaptation Layer Specification: Type 3/4 AAL," Geneva, Aug. 1996.
11. ITU-T Recommendation I.363.5, "B-ISDN ATM Adaptation Layer Specification: Type 5 AAL," Geneva, Aug. 1996.
12. ANSI T1.105, "SONET-Basic Description including Multiplex Structure, Rates, and Formats," 1995.

INDEX

- Abacus switch:
 architecture, 190–193
 ATM routing and concentration chip, 208–211
 enhanced configuration, 211–220
 buffered multistage concentration network, 214–217
 complexity comparisons, 219–220
 memoryless multistage concentration network, 212–214
 resequencing cells, 217–219
input port controller (IPC), 197–198
multicast contention resolution algorithm, 193–197
packet switching, 220–224
 cell interleaving, 222–224
 packet interleaving, 220–222
performance analysis, 198–208
 average delay, 203–205
 cell loss probability, 206–208
 maximum throughput, 199–203
research issues, 189–190
Accept pointer, input-buffered switch scheduling:
 iSLIP scheme, 60–62
 iterative round-robin matching (iRRM), 60
Access point (AP):
 SONET protocols, section access point identifier, 414–417
 wireless ATM (WATM) switches, 339
Access point control protocol (APCP), wireless ATM (WATM) switches, 339–341
Acknowledgement (ACK) packet format, wireless ATM (WATM) switches, data link control layer, 347
Adaptive homing algorithm, wireless ATM (WATM) switches, BAHAMA wireless ATM LAN, 343
AddPrefix (X, Y, Z) algorithm, internet protocol (IP) route lookups, two-trie structure, 399–402
Address broadcaster (AB):
 abacus switch, architecture, 191–193
 fault-tolerant multicast output-buffered ATM switch, fault detection, 173
 multicast grouping networks (MGNs), 157–160
Address copy, multicast shared-memory switch, 99–101
Address filter (AF):
 bus matrix switch (BMX), buffering strategy, 26
 multiple-QoS scalable distributed-arbitration switch (MSDA), 235–236
 time-division switching (TDS), shared-medium switch, 22–23
Address interval, broadcast banyan network (BBN), boolean interval splitting algorithm, 128–129

- Address queues, multicast shared-memory switch, 98–99
- Alarm indication signal (AIS), SONET protocols, 419–421
- All-optical packet switches:
 - ATMOS, 282–283
 - Duan’s switch, 283–284
 - staggering switch, 281–282
- ALOHA system, wireless ATM (WATM) switches:
 - medium access control layer, 346
 - Olivetti’s radio ATM LAN, 342
 - radio physical layer, 346
- Application-specific integrated circuit (ASIC):
 - abacus switch architecture, 208–211
 - wireless ATM (WATM) switches, mobility-support ATM switch, 354–358
- Arbiter device:
 - abacus switch, multicast contention resolution algorithm, 194–197
 - optical interconnection network (OIN), ping-pong arbitration unit (PAU), 315–324
- Arbitrating cells, input-buffered switches, output port contention, 49–50
- Arbitration control (CNTL):
 - multiple-QoS scalable
 - distributed-arbitration switch (SDA), 234–236
 - scalable distributed-arbitration (MSDA) switch, 229–231
- Arbitration schemes:
 - abacus switch, multicast contention resolution algorithm, 194–197
 - asynchronous transfer mode (ATM) switches, 16
 - Atlanta switches, distributed/random arbitration, 261–262
 - input-buffered switch:
 - dual round-robin matching (DRRM), 62–65
 - iterative round-robin matching (iRRM), 58–60
 - iterative round-robin with SLIP (iSLIP), 60–62
 - parallel iterative matching (PIM), 58
 - round-robin arbiters/selectors, 67–72
 - bidirectional arbiter (NTT), 67–70
 - token tunneling, 70–72
 - round-robin greedy scheduling (RRGS), 65–67
 - input-buffered switches, scheduling algorithms, 57
- optical interconnection network (OIN), ping-pong arbitration unit (PAU), 316–324
- Arrayed-waveguide grating (AWG) router, optical interconnection network (OIN), 309–315
- crosstalk analysis, 329–331
- tunable filters, 312–315
- Asynchronous transfer mode (ATM) switches.
 - See also* specific ATM switches, e.g., Wireless ATM
 - applications, 2
 - architecture:
 - buffering strategies, 34–37
 - classification, 21–37
 - design and performance criteria, 13–17
 - space-division switching (SDS), 24–34
 - multiple-path switches, 29–34
 - single-path switches, 25–29
 - time-division switching (TDS), 22–24
 - shared-medium switch, 22–23
 - shared-memory switch, 23–24
 - basic concepts, 15–17
 - call splitting, 20–24
 - head-of-line (HOL) blocking, 19
 - internal link blocking, 17–18
 - multicasting, 20–21
 - network basics, 35
 - optical ATMOS switch, 282–283
 - output port contention, 16–18
 - protocols:
 - background, 407–409
 - cell loss priority, 428–429
 - cell structure, 425–426
 - payload type identifier, 427–428
 - reference model, 409–410
 - virtual path and channel identifiers, 426–427
 - switch structure, 58
- Atlanta switch:
 - architecture, 261
 - configuration, 259–261
 - distributed and random and arbitration, 261–262
 - multicasting, 262–263
- ATM adaptation layer (AAL), asynchronous transfer mode (ATM) network protocol, 408–409, 425–429
- ATM routing and concentration (ARC) chip, abacus switch, 208–211
- ATM Wireless Access (AWA), wireless ATM (WATM) switches, 343
- Automatic protection switching (APS), SONET protocols, 419–421

- Automatic repeat request (ARQ), wireless ATM (WATM) switches, BAHAMA wireless ATM LAN, 343
- Backpressure (BP) scheme, abacus switch performance, cell loss probability (CLP), 206–208
- BAHAMA wireless ATM LAN, wireless ATM (WATM) switches, 343
- Banyan-based switches:
- augmented multiple-path architecture, 30
 - batcher-sorting network, 106–109
 - buffering strategies, 34–35
 - common properties, 103–106
 - deflection routing, 114–125
 - dual shuffle-exchange network, error correction, 118–125
 - shuffle-exchange network, 117–118
 - tandem switches, 114–117
 - interconnection networks, 103–106
 - multicast copy networks, 125–138
 - broadcast banyan network, 127–129
 - boolean interval splitting algorithm, 128–129
 - nonblocking condition, 129
 - self-routing algorithm, 127–128
 - concentration, 132
 - decoding, 133
 - encoding process, 129–132
 - overflow and call splitting, 133–134
 - overflow and input fairness, 134–138
 - concentration, 137–138
 - cyclic running adder network (CRAN), 135–137
 - output contention resolution algorithms, 110–112
 - single-path topologies, 28–29
 - Sunshine switch, 112–114
- Base station controller (BCS), wireless ATM (WATM) switches, 339
- Batcher-sorting network, banyan-based switches, 106–109
- ring reservation, 110–112
- Sunshine switch, 112–114
- three-phase implementation algorithm, 109–110
- Benes network topology, Washington University gigabit switch (WUGS), 95–96
- Bernoulli arrival process:
- input-buffered switch, random traffic and, 52
 - tandem-crosspoint (TDXP) switch performance, 246–247
- Bernoulli distribution, fault-tolerant multicast output-buffered ATM switch, performance analysis, 181–185
- Best-first lowest-output-occupancy-cell-first algorithm (LOOFA), input-buffered switch, 79–80
- Best-matching-prefix problem, internet protocol (IP) route lookups, binary search, 381–384
- Bidirectional arbiter (NTT), input-buffered switch, 67–70
- Binary search, internet protocol (IP) route lookups:
- best-matching prefix, 381–384
 - multiway search, cache exploitation, 385–388
- Binary tree, internet protocol route lookups, 368–369
- Bit error rate (BER):
- optical interconnection network (OIN), power budget analysis, 328
 - wireless ATM (WATM) switches, data link control layer, 346–347
- Blocking switch:
- internal link blocking, 17–18
 - three-stage Clos switches as, 31–33
- BNR switch, architecture, 289–290
- Boolean interval splitting algorithm:
- banyan network switches, multicast copy, 126
 - broadcast banyan network (BBN), 128–129
- Broadband Radio Access Networks (BRAN) project, wireless ATM (WATM) switches, medium access control layers, 346
- Broadband switching systems (BSSs), packet switch architecture, 13–14
- Broadcast banyan network (BBN), multicast copying, 125, 127–129
- boolean interval splitting algorithm, 128–129
- nonblocking condition, 129
- self-routing algorithm, 127–128
- Broadcast channel number (BCN):
- abacus switch:
- architecture, 191–193
 - input port controller implementation, 198
 - banyan network switches, multicast copy, 126
 - two-stage multicast out-put-buffered ATM switch (MOBAS), translation tables, 161–163
- Broadcast mode, asynchronous transfer mode (ATM) networks, 45
- Buffered multistage concentration network (BMCN), enhanced abacus switch, 214–217

- Buffering strategies:
 asynchronous transfer mode (ATM)
 switches, 17
 crosspoint-buffered switches, 35
 input-buffered switches, 35–36
 internally buffered switches, 34–35
 multistage shared-buffer switches, 36
 output-buffered switches, 36
 recirculated buffered switches, 35
 shared-buffer switches, 36
 virtual-output-queuing (VOQ) switches,
 36–37
- crossbar switches, 26–27
 wireless ATM (WATM) switches, handoff
 process, 350–351
- Burst length (BL), wireless ATM (WATM)
 switches, mobility-support ATM switch,
 360–362
- Bursty traffic model:
 abacus switch, maximum throughput
 performance, 202–203
 input-buffered switch, 52–53
 wireless ATM (WATM) switches,
 mobility-support ATM switch, 360–362
- Bus interface, single-stage knockout switch,
 143
- Bus matrix switch (BMX), buffering strategy,
 26–27
- Caching technique, internet protocol route
 lookups, 369
 multiway binary search, cache exploitation,
 385–388
- Call splitting:
 asynchronous transfer mode (ATM)
 switches, 20–24
 banyan-based switches, overflow and,
 133–134
- Capacity graphing, Path switching, 270–272
- Cell addition/deletion:
 shared-memory switch, linked list logical
 queues, 86–90
 tandem banyan switching fabric (TBSF),
 deflection routing, 116–117
- Cell copy, multicast shared-memory switch,
 98–99
- Cell delay variation (CDV), design and
 performance criteria, 13–14
- Cell delineation unit, 3M optical switch,
 294–296
- Cell interleaving, abacus-based packet
 switching, 222–224
- Cell loss probability (CLP):
 abacus switch performance, 206–208
- asynchronous transfer mode (ATM)
 networks, 428–429
- design and performance criteria, 14
- knockout-based switches:
 channel grouping, 152–154
 single-stage knockout switch, 144–146
- output-buffered switch, performance
 evaluation, 41–44
- shared-buffer switches, performance
 evaluation, 44–46
- two-stage multicast out-put-buffered ATM
 switch (MOBAS), 163–169
- wireless ATM (WATM) switches,
 mobility-support ATM switch, 361–362
- Cell routing, wireless ATM (WATM) switches:
 handoff process, crossover switch (COS),
 351–352
 mobility-support ATM switch, 353–358
- Cell sequence number (CSN), wireless ATM
 (WATM) switches, data link control layer,
 347
- Cell synchronization unit, 3M optical switch,
 297–301
- Centralized connection processors,
 asynchronous transfer mode (ATM)
 switches, 16
- Central switching network, Washington
 University gigabit switch (WUGS),
 95–96
- Channel grouping, knockout-based switches,
 150–154
 cell loss probability, 152–154
 maximum throughput, 150–152
- Circuit switching schemes, asynchronous
 transfer mode (ATM) switches, 15–16
- Cisneros-Brackett optical switch, architecture,
 287–288
- Classless interdomain routing (CIDR), internet
 protocol route lookups, 366
- Clos-network switches:
 Atlanta switch:
 architecture, 261
 configuration, 259–261
 distributed and random and arbitration,
 261–262
 multicasting, 262–263
 blocking switch, three-stage Clos switches
 as, 31–33
 concurrent round-robin dispatching switch:
 architecture, 264–265
 concurrent dispatching, 265–267
 configuration, 263–264
 desynchronization effect, 267–268
 dynamic routing, suboptimal straight
 matching method, 258–259

- path switch:
 - configuration, 268–272
 - heterogeneous capacity assignment, 274–277
 - homogeneous capacity and route assignment, 272–274
- research issues, 253–255
- routing properties and scheduling methods, 255–257
- Codeword array (CWA), internet protocol (IP)
 - route lookups, gigabit switch routers, construction algorithm, 393–395
- Complete partitioning, shared-memory switch, 24
- Complexity comparisons:
 - enhanced abacus switch, 219–220
 - optical interconnection network (OIN), 324–326
- Compressed next-hop array (CNHA), internet protocol (IP) route lookups, gigabit switch routers, 393–395
- Concentration modules (CM), enhanced abacus switch:
 - buffered multistage concentration network (BMCN), 214–217
 - memoryless multistage concentration network, 212–214
- Concentration principle:
 - banyan network switches, multicast copy, 132
 - cyclic running adder network (CRAN), 137–138
 - overflow fairness, 134–138
 - single-stage knockout switch, 144–146
 - construction, 146–150
- Concentrator-based growable switch
 - architecture, multistage shared-memory applications, 96
- Concurrent round-robin dispatching (CRRD) switch:
 - architecture, 264–265
 - concurrent dispatching, 265–267
 - configuration, 263–264
 - desynchronization effect, 267–268
- Congestion flow, asynchronous transfer mode (ATM) networks, 428
- Connection rerouting, wireless ATM (WATM) switches, handoff process, 348–350
- Content-addressable memory (CAM) technique, shared-memory switch, 91–93
- Contention resolution algorithms, STAR-TRACK switch, 286–287
- Contention resolution device (CRD), Cisneros-Brackett optical switch, 287–288
- Contention switches, single-stage knockout switch, concentrator construction, 146–150
- Control packets, wireless ATM (WATM)
 - switches, radio access layers, 345–346
- Control plane (C-plane), asynchronous transfer mode (ATM) network protocol, 409
- Copy number (CN), banyan network switches, multicast copy, 125–126
- cyclic running adder network (CRAN), 136–138
- encoding process, 130–132
- Critical cell first (CCF), input-buffered switch, output-queuing emulation, 74–75
- Crossbar switches:
 - architecture, 25–27
 - buffering strategies, 35
 - multicast grouping networks (MGNs), 159–160
- Crossover switch (COS), wireless ATM (WATM) switches:
 - handoff process, 350
 - buffering strategies, 350–351
 - cell routing, 351–352
 - mobility-support ATM switch, 352–362
- Crosspoint-buffered switches:
 - multiple-QoS scalable distributed-arbitration switch (MSDA):
 - performance analysis, 236–238
 - structure, 234–236
 - research issues, 227–229
 - scalable distributed-arbitration switch (SDA):
 - performance analysis, 231–233
 - structure, 229–231
- Crosspoint unit (XPU), input-buffered switch, token tunneling, 71–72
- Cross-stuck (CS) fault, fault-tolerant multicast output-buffered ATM switch:
 - fault detection, 172–174
 - location and configuration, 175–177
 - performance analysis, 182–183
 - switch element (SWE), 170–171
- Crosstalk analysis, optical interconnection network (OIN), 328–331
- Cyclic redundancy check (CRC), 3M optical switch, cell delineation unit, 294–296
- Cyclic running adder network (CRAN), banyan-based switches, overflow and input fairness, 135–138
- Data link control (DLC) layer, wireless ATM (WATM) switches:
 - NEC WATMnet prototype system, 341–342
 - protocol, 340–341
 - research and development, 337–338

- Data link control layer, wireless ATM (WATM) switches, radio access layers, 346–347
- Data packet flow, terabit IP router architecture, 305–306
- Data structure layers, internet protocol (IP) route lookups:
forwarding table construction, 374–377
gigabit switch routers, 388–395
- Decoding, banyan-based switches, multicast copy networks, 133
- Deflection routing, banyan-based switches, 114–125
dual shuffle-exchange network, error correction, 118–125
shuffle-exchange network, 117–118
tandem switches, 114–117
- Delay performance:
abacus switch, 203–205
scalable distributed-arbitration (SDA) switch, 231–233
tandem-crosspoint (TDXP) switch, 248–251
- DelPrefix (X,Y) algorithm, internet protocol (IP) route lookups, two-trie structure, 402–403
- Delta-based switches, architecture, 29
- Dense-wavelength-division-multiplexing (DWDM) equipment, channel multiplexing applications, 12
- Descrambling procedures, SONET protocols, 417–418
- Destination address (DA), banyan-based switches, Sunshine switch, 113–114
- Desynchronization, concurrent round-robin dispatching (CRRD), 267–268
- DIR-24-8-BASIC scheme, internet protocol (IP) route lookups, memory access speeds, 377–381
- Dispatching, concurrent round-robin dispatching (CRRD) switch, 265–267
- Distributed Bragg reflector (DBR), optical interconnection network (OIN), input optical module (IOM), 310
- Distributed-queuing request update multiple access (DQRUMA), wireless ATM (WATM) switches, 343
- Dual round-robin matching (DRRM):
Clos-network switches, concurrent round-robin dispatching (CRRD) switch, 264
input-buffered switch scheduling, 62–65
- Dual shuffle-exchange network (DSN), deflection routing, error correction, 118–125
- Duan's switch, properties, 283–284
- Dummy address encoder (DAE), banyan-based switches, 125
encoding process, 129–132
- Dummy address interval, banyan network switches, multicast copy, 126
- Dynamic routing, Clos network switches, suboptimal straight matching method, 258–259
- Edge coloring, Path switches, 276–277
- Electroabsorption modulators (EAM), optical interconnection network (OIN), complexity comparisons, 324–326
- Electronic interconnection network (EIN), terabit IP routers, optical packet switches, 301–303
- Encoding process, banyan-based switches, multicast copy networks, 129–132
- Enhanced abacus switch, 211–220
buffered multistage concentration network, 214–217
complexity comparisons, 219–220
memoryless multistage concentration network, 212–214
resequencing cells, 217–219
- Erbium-doped fiber amplifier (EDFA):
3M optical switch, 292–294
optical interconnection network (OIN), power budget analysis, 326–328
- Error control systems, wireless ATM (WATM) switches, data link control layer, 346–347
- Error-correcting routing, dual shuffle-exchange network (SN), 118–125
- External modulator (EM), optical interconnection network (OIN), input optical module (IOM), 310
- Far end block error (FEBE) function, SONET protocols, 416–417
- Fault detectors (FD), fault-tolerant multicast output-buffered ATM switch, 172–174
cross-stuck/toggle-stuck detection, 172–173
vertical-stuck/horizontal-stuck fault detection, 173
- Fault location and configuration, fault-tolerant multicast output-buffered ATM switch, 174–181
- Fault-tolerant multicast output-buffered ATM switch, 169–185
fault detection, 172–174
cross-stuck/toggle-stuck detection, 172–173

- vertical-stuck/horizontal-stuck fault detection, 173
- fault location and reconfiguration, 173–181
 - cross-stuck/toggle-stuck cases, 175–177
 - vertical-stuck/horizontal-stuck cases, 177–181
- performance analysis, switch reconfiguration, 181–185
 - cross-stuck/toggle-stuck cases, 182–183
 - horizontal-stuck case, 184–185
 - vertical-stuck case, 183–184
- switch element fault model, 169–172
 - cross-stuck (CS) fault, 170–171
 - toggle-stuck (TS) fault, 171–172
 - vertical/horizontal-stuck (VS/HS) fault, 172
- Feedback priority (FP) signals:
 - abacus switch:
 - input port controller implementation, 198
 - multicast contention resolution algorithm, 195–197
 - enhanced abacus switch:
 - buffered multistage concentration network (BMCN), 214–217
 - memoryless multistage concentration network, 213–214
- Fiber throughput technology, link transmission speed, 1
- Fine adjustment circuit timing, 3M optical switch cell synchronization unit, 299–301
- First-come, first-served (FCFS) principle, abacus switch, architecture, 193
- First-in-first-out (FIFO) buffer:
 - ATM switch structure, 58
 - concentrator-based growable switch architecture, 96
 - input-buffered switches:
 - performance evaluation, 37–40
 - scheduling algorithms, 57
 - 3M optical switch, 293–294
 - output-buffered switch, performance evaluation, 40–44
 - shared-memory switch, linked list logical queues, 90
 - single-stage knockout switch, 143
- terabit IP router architecture:
 - data packet flow, 305–306
 - routing module and route controller, 308
- time-division switching (TDS),
 - shared-medium switch, 22–23
 - wireless ATM (WATM) switches, mobility-support ATM switch, 355–358
- Fixed-size data units, high-end routers, IP architecture, 11–12
- Fixed wireless networks, wireless ATM (WATM) switches, 338
- Forward error correction (FEC), wireless ATM (WATM) switches, BAHAMA wireless ATM LAN, 343
- Forwarding information base (FIB), optical interconnection network (OIN), terabit IP router architecture, 308–309
- Forwarding table:
 - internet protocol (IP) route lookups, fast route lookup configuration, 373–377
 - internet protocol route lookups, 367
- Frame number, wireless ATM (WATM) switches, radio access layers, 345–346
- Frequency justification, SONET protocols, 418–419
- Full sharing, shared-memory switch, 24
- Fully interconnected switches, architecture, 26, 28
- Geom/G/1* queuing model, input-buffered switches, performance evaluation, 39–40
- Gigabit switch routers, internet protocol (IP) route lookups, 388–396
 - algorithms and data structures, 388–395
 - CBM/CNHA construction, 393–395
 - NHA construction algorithm, 392–393
 - performance analysis, 395–396
- Grant pointer, input-buffered switch scheduling:
 - iSLIP* scheme, 60–62
 - iterative round-robin matching (iRRM), 60
- Greedy lowest-output-occupancy-cell-first algorithm (LOOFA), input-buffered switch, 78–80
- Group expansion ratio, abacus switch:
 - architecture, 191–193
 - cell loss probability (CLP), 206–208
 - maximum throughput performance, 200–203
- Handoff process, wireless ATM (WATM) switches, 347–352
 - buffering, 350–351
 - connection rerouting, 348–350
 - COS cell routing, 351–352
 - mobility-support ATM switch, 353–358
- Handoff rate (HR), wireless ATM (WATM) switches, mobility-support ATM switch, 360–362
- Hardware systems, internet protocol (IP) route lookups, memory access speeds, 377–381
- Header error control (HEC):
 - design and performance criteria, 14
 - SONET protocols, 423–425

- Header field values, asynchronous transfer mode (ATM) networks, 428–429
- Head-of-line (HOL) blocking:
- abacus switch:
 - architecture, 192–193
 - delay, 203–205
 - input port controller implementation, 197–198
 - maximum throughput performance, 199–203
 - multicast contention resolution algorithm, 194–197
 - packet interleaving, 221–222
 - research issues, 189–190
 - asynchronous transfer mode (ATM) switches, 19
- Cisneros-Brackett optical switch, 288
- enhanced abacus switch, resequencing, 217–219
- input-buffered switches, 37–40
- Bernoulli arrival process and random traffic, 52
 - dual round-robin matching (DRRM) scheduling, 63–65
 - models, 51
 - throughput limitation, 49–50
 - virtual-output-queuing (VOQ)-based matching, 55–57
 - window-based lookahead selection, 54–55
- optical interconnection network (OIN), 302–303
- ping-pong arbitration unit (PAU), 315–324
- shared-memory switch:
- linked list technique, 85–90
 - multicast shared-memory switch, 97–98
- tandem-crosspoint (TDXP) switch:
- delay performance, 248–251
 - input-output-buffered switches, 239–241
 - unicasting operation, 244–245
- virtual-output-queuing (VOQ) switches, 36–37
- wave-mux switch, 291
- wireless ATM (WATM) switches,
- mobility-support ATM switch, 354–358
- Head pointer (HP):
- shared-memory switch, logical queue, 85–90
 - wireless ATM (WATM) switches,
 - mobility-support ATM switch, 356–358
- Head pointer register (HPR), shared-memory switch, linked list logical queues, 86–90
- HEC checking mechanism, 3M optical switch, cell delineation unit, 294–296
- Heterogeneous capacity assignment, Path switching, 274–277
- edge coloring, 276–277
 - roundoff procedure, 275–276
 - virtual path capacity allocation (VPCA), 274–275
- High-end routers, IP architecture, 10–12
- Homogeneous capacity, Path switching, 272–274
- Homowavelength crosstalk, optical interconnection network (OIN), 328–331
- HPS finite state machine, 3M optical switch, cell delineation unit, 295–296
- Hungarian algorithm, Path switches, 276–277
- HUNT state, 3M optical switch, cell delineation unit, 295–296
- HYPASS optical switch, configuration, 284–286
- Idle address FIFO (IAF), shared-memory switch:
- content-addressable memory (CAM) technique, 92–93
 - linked list logical queues, 86–90
- IDLE interface, banyan-based switches, ring head-end (RHE), 111–112
- Incoherent crosstalk, optical interconnection network (OIN), 328–331
- Index reference (IR), banyan network switches, multicast copy, 126
- Input-buffer delay, abacus switch performance, 203–205
- cell loss probability (CLP), 206–208
- Input-buffered switch:
- asynchronous transfer mode (ATM):
 - buffering strategies, 35–36
 - defined, 16
 - lowest-output-occupancy-cell-first algorithm (LOOFA), 78–80
 - models:
 - head-of-line blocking phenomenon, 51
 - traffic models, throughput results, 52–53
 - output-queuing emulation, 72–78
 - Chang algorithms, 73–74
 - critical cell first (CCF), 74–75
 - last in, highest priority (LIHP), 75–78
 - most-urgent-cell-first algorithm (MUCFA), 72–73
- performance evaluation, 37–40
- performance improvement:
- internal capacity increase, 53–54
 - scheduling efficiency, 54–57
- research issues, 49–50

- scheduling algorithms, 57–71
 - dual round-robin matching (DRRM), 62–65
 - iterative round-robin matching (iRRM), 58–60
 - iterative round-robin with SLIP (iSLIP), 60–62
 - parallel iterative matching (PIM), 58
 - round-robin arbiters/selectors, 67–72
 - bidirectional arbiter (NTT), 67–70
 - token tunneling, 69–72
 - round-robin greedy scheduling (RRGS), 65–67
- Input concentration, banyan network switches, multicast copy, 132
- cyclic running adder network (CRAN), 137–138
- overflow fairness, 134–138
- Input forwarding engine (IFE), terabit IP routers, optical packet switches:
 - data packet flow, 305
 - routing module and route controller, 306
- Input group module (IGM), wave-mux switch, 290–291
- Input line interface (ILI), terabit IP router architecture:
 - data packet flow, 305–306
 - routing module and route controller, 306
- Input optical modules (IOM), optical
 - interconnection network (OIN), 309–315
 - crosstalk analysis, 328–331
- Input-output-buffered switches, tandem-crosspoint (TDXP) switch, 239–241
- Input packet filter (IPF), terabit IP router architecture, data packet flow, 305–306
- Input port controllers (IPCs):
 - abacus switch:
 - architecture, 190–193
 - enhanced configuration, 211–220
 - implementation, 197–198
 - multicast contention resolution algorithm, 194–197
 - ATM switch structure, 58
 - banyan-based switches, Sunshine switch, 113–114
 - fault-tolerant multicast output-buffered ATM switch, fault detection, 173
 - two-stage multicast out-put-buffered ATM switch (MOBAS), 154–157
 - translation tables, 160–163
- Input port processors (IPPs), Washington University gigabit switch (WUGS), 94–96
- Input queuing, crossbar switches, buffering strategy, 26
- Input routing module (IRM), optical interconnection network (OIN), terabit IP router architecture, 303
- Input smoothing, input-buffered switch, 53
- Input switch interface (ISI), terabit IP router architecture:
 - data packet flow, 305–306
 - routing module and route controller, 306–308
- Input thread (IT), input-buffered switch, output-queuing emulation, 74
- Integrated local management interface (ILMI), asynchronous transfer mode (ATM)
 - network protocol, 409–410
- Interconnection complexity, optical
 - interconnection network (OIN), 326
- Intermediate stage controller (ISC), buffered multistage concentration network (BMCN), 216–217
- Internal blocking, banyan-based switches, 105
- Internal capacity, input-buffered switch:
 - multiline (input smoothing), 53
 - parallel switching, 54
 - speedup factor, 54
- Internal link blocking, asynchronous transfer mode (ATM), 17–18
- Internet protocol (IP):
 - dominance of, 2
 - route lookups:
 - caching technique, 369
 - design issues, 368–369
 - gigabit switch routers, 388–396
 - algorithms and data structures, 388–395
 - CBM/CNHA construction, 393–395
 - NHA construction algorithm, 392–393
 - performance analysis, 395–396
 - hardware, memory access speeds, 377–381
 - multiway search, 381–388
 - binary search, best-matching prefix, 381–384
 - cache line exploitation, 385–388
 - performance analysis, 388
 - precomputed 16-bit prefix table, 384–385
 - Patricia tree, 372
 - research issues, 365–366
 - small forwarding tables, fast lookups, 373–377
 - standard trie structure, 369–372
 - two-trie structure, 396–404
 - AddPrefix (X, Y, Z) algorithm, 399–402
 - DelPrefix (X, Y) algorithm, 402–403
 - IPLookup (X) algorithm, 397–398
 - performance analysis, 403–404
 - prefix update algorithm, 398–399

Internet protocol (*Continued*)

router systems:

- architectures, 9–13, 366–367
 - high-end routers, 10–12
 - low-end routers, 9–10
 - middle-end routers, 10
- switch fabric for high-end routers, 12–13
 - function, 89
- terabit routers, optical interconnection network:
 - complexity issues, 324–326
 - crosstalk analysis, 328–331
 - network configuration, 309–315
 - ping-pong arbitration (PPA) unit, 315–324
 - power budget analysis, 326–328
 - research issues, 301–303
 - router module and route controller, 306–309
 - terabit architecture, 303–306

Interworking function (IWF), wireless ATM (WATM) switches, 339

IPLookup (X) algorithm, internet protocol (IP) route lookups, two-trie structure, 397–398

IP route lookup algorithm, internet protocol route lookups, multiway binary search, cache exploitation, 387–388

Iterative round-robin matching (iRRM), input-buffered switch scheduling, 58–60 with SLIP (*iSLIP*), 60–62

Knockout-based switches:

- channel grouping, 150–154
- cell loss probability, 152–154
- maximum throughput, 150–152
- concentration principle, 144–146
- concentrator construction, 146–150
- fault-tolerant multicast output-buffered ATM switch, 169–185
- fault detection, 172–174
 - cross-stuck/toggle-stuck detection, 172–173
 - vertical-stuck/horizontal-stuck fault detection, 173
- fault location and reconfiguration, 173–181
 - cross-stuck/toggle-stuck cases, 175–177
 - vertical-stuck/horizontal-stuck cases, 177–181
- performance analysis, switch reconfiguration, 181–185
 - cross-stuck/toggle-stuck cases, 182–183
 - horizontal-stuck case, 184–185
 - vertical-stuck case, 183–184

switch element fault model, 169–172

- cross-stuck (CS) fault, 170–171
- toggle-stuck (TS) fault, 171–172
- vertical/horizontal-stuck (VS/HS) fault, 172

research issues, 141–142

single-stage architecture, 142–143

two-stage multicast out-put-buffered ATM switch, 154–169

- multicast grouping network, 157–160
- multicast knockout principle, 163–169
 - translation tables, 160–163
- two-stage configuration, 154–157

Knockout principle:

- channel grouping, 152–154
- development of, 141–142

Label-routing methods, ATM switch structure, 79

Label switching routers (LSRs), applications, 2

LambdaRouter, Internet protocol (IP) over wavelength networks, 12

Last in, highest priority (LIHP) scheme, input-buffered switch, output-queuing emulation, 75–78

Latin square assignment, Path switching, 272–274

Line interface card (LIC), ATM switch structure, 58

Linked list technique, shared-memory switch, 84–90

- vs.* content-addressable memory (CAM) technique, 92

Local priority (LP), abacus switch, multicast contention resolution algorithm, 195–197

Logical queues:

- shared-memory switch:
 - linked list technique, 84–90
 - multicast shared-memory switch, 97–98
 - wireless ATM (WATM) switches,
 - mobility-support ATM switch, 355–358

Lookup algorithms:

- gigabit switch routers, 388–395
- IP route lookup algorithm, multiway binary search, 387–388
- two-trie IP route lookup structure, 397–398

Low-end routers, IP architecture, 910

Lowest-output-occupancy-cell-first algorithm (LOOFA), input-buffered switch, 78–80

Magic WAND, wireless ATM (WATM) switches, 343–344

- Management plane, asynchronous transfer mode (ATM) network protocol, 410
- Markov chain modeling, output-buffered switch, performance evaluation, 41–44
- Markov process, input-buffered switches, performance evaluation, 38–40
- Maximal matching, input-buffered switches, virtual-output-queuing (VOQ)-based matching, 56–57
- Maximum matching, input-buffered switches, virtual-output-queuing (VOQ)-based matching, 56–57
- Maximum throughput:
abacus switch performance, 199–203
knockout-based switches, channel grouping, 150–152
- MEDIAN system, wireless ATM (WATM) switches, 344
- Medium access control (MAC) layer, wireless ATM (WATM) switches:
NEC WATMnet prototype system, 341–342
protocol, 340–341
radio access layers, 346
research and development, 337–338
- Memory access speeds, internet protocol (IP) route lookups, hardware systems, 377–381
- Memoryless multistage concentration network (MMCM), enhanced abacus switch, 212–214
- Memory-space-memory (MSM), Atlanta switch configuration, 259–263
- Merge networks, banyan-based switches, batcher-sorting network, 106–109
- Microelectromechanical systems (MEMS), LambdaRouter technology, 12
- Middle-size routers, IP architecture, 10
- Mobility-support ATM switch, wireless ATM (WATM) switches, 352–362
design issues, 353–358
performance analysis, 358–362
- 3M optical switch:
architecture, 291–294
cell delineation unit, 294–296
cell synchronization unit, 297–301
VCI-overwrite unit, 296–297
- Most significant bit (MSB), abacus switch, packet interleaving, 221–222
- Most-urgent-cell-first algorithm (MUCFA), input-buffered switch, output-queuing emulation, 72–73
- Multicast cell counters (MCCs), multicast shared-memory switch, 99–101
- Multicast contention resolution algorithm, abacus switch, 193–197
- Multicast contention resolution unit (MCRU), abacus switch, input port controller implementation, 197–198
- Multicast copy networks, banyan-based switches, 125–138
broadcast banyan network, 127–129
boolean interval splitting algorithm, 128–129
nonblocking condition, 129
self-routing algorithm, 127–128
concentration, 132
decoding, 133
encoding process, 129–132
overflow and call splitting, 133–134
overflow and input fairness, 134–138
concentration, 137–138
cyclic running adder network (CRAN), 135–137
- Multicast grouping networks (MGNs):
abacus switch:
architecture, 190–193
enhanced configuration, 211–220
multicast contention resolution algorithm, 195–197
two-stage multicast output-buffered ATM switch (MOBAS), 154–157
cell loss rates, 163–169
switch module, 157–160
translation tables, 160–163
- Multicasting:
asynchronous transfer mode (ATM) networks, 45, 19–20
asynchronous transfer mode (ATM) switches, call splitting, 20–21
Atlanta switches, 262–263
tandem-crosspoint (TDXP) switch, 246
- Multicast output-buffered ATM switch (MOBAS):
abacus switches, architectural comparisons, 190–193
two-stage structure, 154–169
multicast grouping network, 157–160
multicast knockout principle, 163–169
translation tables, 160–163
two-stage configuration, 154–157
- Multicast pattern maskers (MPMs):
abacus switch, architecture, 191–193
fault-tolerant multicast output-buffered ATM switch, fault detection, 172–173
multicast grouping networks (MGNs), 157–160
- Multicast pattern (MP):
abacus switch:
architecture, 191–193

- Multicast pattern (*Continued*)
 input port controller implementation, 198
 terabit IP router architecture, routing
 module and route controller, 307–308
- Multicast shared-memory switch, 96–101
 address copy, 99–101
 cell copy, 98–99
 logical queuing, 97–98
- Multicast translation tables (MTTs):
 abacus switch, architecture, 190–193
 fault-tolerant multicast output-buffered ATM switch, 173
 two-stage multicast out-put-buffered ATM switch (MOBAS), 154–157, 160–163
- Multiline internal capacity, input-buffered switch, 53
- Multiplane switches, space-division switching (SDS) architecture, 33
- Multiple-path switches:
 architecture, 29–34
 space-division switching (SDS), 29–34
 augmented banyan switches, 30
 multiplane switches, 33
 recirculation switches, 33–34
 three-stage Clos switches, 30–33
- Multiple-QoS scalable distributed-arbitration switch (SDA):
 performance analysis, 236–238
 structure, 234–236
- Multi-protocol label switching (MPLS), advantages, 2
- Multistage shared-memory switch, 94–96
 concentrator-based growable switch architecture, 96
 Washington University gigabit switch, 94–96
- Multiway search, internet protocol (IP) route lookups, 381–388
 binary search, best-matching prefix, 381–384
 cache line exploitation, 385–388
 performance analysis, 388
 precomputed 16-bit prefix table, 384–385
- NEC WATMnet prototype system, wireless ATM (WATM) switches, 341–342
- Network interface card (NIC):
 asynchronous transfer mode (ATM) switches, 15
 internet protocol route lookups, 366–367
- Network node interface (NNI), asynchronous transfer mode (ATM) networks, 35
- Network processor, internet protocol route lookups, 366–367
- Next-generation internet (NGI), terabit IP routers, optical interconnection network:
- complexity issues, 324–326
 crosstalk analysis, 328–331
 network configuration, 309–315
 ping-pong arbitration (PPA) unit, 315–324
 power budget analysis, 326–328
 research issues, 301–303
 router module and route controller, 306–309
 terabit architecture, 303–306
- Next-hop array (NHA), internet protocol (IP)
 route lookups, gigabit switch routers, 388–395
 construction algorithm, 392–393
- Next pointers (NP), shared-memory switch, linked list logical queues, 86–90
- Nonblocking conditions:
 broadcast banyan network (BBN), 129
 Clos network switches, 255
- Nonblocking switch:
 banyan-based switches, 105–106
 internal link blocking, 17–18
 three-stage Clos switches as, 31–33
- Not-acknowledgment (NACK) signal:
 multiple-QoS scalable distributed-arbitration switch (SDA), 236
 scalable distributed-arbitration (SDA) switch, 229–231
- tandem-crosspoint (TDXP) switch:
 delay performance, 248–251
 multicasting operation, 246
 unicasting operation, 243–245
- NTT's wireless ATM access, wireless ATM (WATM) switches, 343
- OC-*N* multiplexer, SONET protocols, 422–423
- Olivetti's radio ATM LAN, wireless ATM (WATM) switches, 342
- One-shot scheduling, asynchronous transfer mode (ATM) switches, multicast call splitting, 20–21
- On-off model:
 abacus switch performance, 198–199
 input-buffered switch, bursty traffic and, 52–53
 optical interconnection network (OIN), power budget analysis, 327–328
- Open systems interconnection (OSI) reference model, asynchronous transfer mode (ATM) networks, 35
- Optical cross connect (OXC) system, LambdaRouter technology, 12
- Optical delay line, 3M optical switch cell synchronization unit, 298–301
- Optical interconnection network (OIN), terabit IP routers, optical packet switches:

- complexity issues, 324–326
- crosstalk analysis, 328–331
- network configuration, 309–315
- ping-pong arbitration (PPA) unit, 315–324
- power budget analysis, 326–328
- research issues, 301–303
- router module and route controller, 306–309
- terabit architecture, 303–306
- Optically transparent switches:**
 - properties, 280
 - staggering switch, 281–282
- Optical packet switches:**
 - all-optical packet switches:
 - ATMOS, 282–283
 - Duan's switch, 283–284
 - staggering switch, 281–282
- 3M switch:**
 - architecture, 291–294
 - cell delineation unit, 294–296
 - cell synchronization unit, 297–301
 - VCI-overwrite unit, 296–297
- optical interconnection network, terabit IP routers:
 - complexity issues, 324–326
 - crosstalk analysis, 328–331
 - network configuration, 309–315
 - ping-pong arbitration (PPA) unit, 315–324
 - power budget analysis, 326–328
 - research issues, 301–303
 - router module and route controller, 306–309
 - terabit architecture, 303–306
- Optoelectronic packet switches:**
 - BNR switch, 289–290
 - Cisneros and Brackett architecture, 287–288
 - HYPASS, 284–286
 - STAR-TRACK, 286–287
 - Wave-Mux switch, 290–291
 - research issues, 279–280
- Optical transmission technology, link transmission speed, 1
- Optoelectronic integrated circuit (OIEC), optical interconnection network (OIN), tunable filters, 312–315
- Optoelectronic packet switches:**
 - BNR switch, 289–290
 - Cisneros and Brackett architecture, 287–288
 - HYPASS, 284–286
 - STAR-TRACK, 286–287
 - Wave-Mux switch, 290–291
- Output-buffer delay, abacus switch performance, 203–205
- cell loss probability (CLP), 206–208
- Output-buffered switch:**
 - asynchronous transfer mode (ATM),
 - defined, 16
 - buffering strategies, 36
 - performance evaluation, 40–44
- Output contention resolution algorithms,
 - banyan-based switches, 110–112
 - ring reservation, 110–112
 - three-phase implementation, 110
- Output cushion (OC), input-buffered switch, output-queuing emulation, 74
- Output forwarding engine (OFE), terabit IP routers, optical packet switches:**
 - data packet flow, 305
 - routing module and route controller, 306
- Output group module (OGM), wave-mux switch, 290–291
- Output line interface (OLI), terabit IP router architecture, routing module and route controller, 306
- Output occupancy (OCC), input-buffered switch, lowest-output-occupancy-cell-first algorithm (LOOFA), 78–80
- Output optical modules (OOM), optical interconnection network (OIN), 309–315
- complexity comparisons, 324–326
- crosstalk analysis, 328–331
- power budget analysis, 326–328
- Output port contention:**
 - asynchronous transfer mode (ATM)
 - switches, 16–18
 - head-of-line blocking, 19
 - input-buffered switches, arbitrating cells, 49–50
- Output port controllers (OPCs):**
 - abacus switch, architecture, 190–193
 - ATM switch structure, 58
 - banyan-based switches, Sunshine switch, 114
 - fault-tolerant multicast output-buffered ATM switch, fault detection, 173
 - two-stage multicast out-put-buffered ATM switch (MOBAS), 154–157
 - cell loss rates, 164–169
 - translation tables, 160–163
- Output port processors (OPPs), Washington University gigabit switch (WUGS), 95–96
- Output-queuing emulation, input-buffered switch, 72–78
 - Chang algorithms, 73–74
 - critical cell first (CCF), 74–75
 - last in, highest priority (LIHP), 75–78
 - most-urgent-cell-first algorithm (MUCFA), 72–73

- Output routing module (ORM):
 optical interconnection network (OIN),
 terabit IP router architecture, 303
 terabit IP router architecture, data packet flow, 305–306
- Output switch interface (OSI), terabit IP router architecture, routing module and route controller, 306–308
- Output switch module (OSM), optical interconnection network (OIN), crosstalk analysis, 328–331
- Overflow, banyan-based switches:
 call splitting and, 133–134
 input fairness and, 134–138
- Overhead bytes, SONET protocols, 414–417
- Packet interleaving, abacus-based packet switching, 220–222
- Packet reassembly unit (PRU), terabit IP router architecture, routing module and route controller, 308
- Packet switches, optical packet switches:
 all-optical packet switches:
 ATMOS, 282–283
 Duan's switch, 283–284
 staggering switch, 281–282
- 3M switch:
 architecture, 291–294
 cell delineation unit, 294–296
 cell synchronization unit, 297–301
 VCI-overwrite unit, 296–297
- optical interconnection network, terabit IP routers:
 complexity issues, 324–326
 crosstalk analysis, 328–331
 network configuration, 309–315
 ping-pong arbitration (PPA) unit, 315–324
 power budget analysis, 326–328
 research issues, 301–303
 router module and route controller, 306–309
 terabit architecture, 303–306
- optoelectronic packet switches:
 BNR switch, 289–290
 Cisneros and Brackett architecture, 287–288
 HYPASS, 284–286
 STAR-TRACK, 286–287
 Wave-Mux switch, 290–291
 research issues, 279–280
- Packet switching:
 abacus switch, 220–224
 cell interleaving, 222–224
 packet interleaving, 220–222
- architecture:
 buffering strategies, 34–37
 classification, 21–37
 design and performance criteria, 13–17
 space-division switching (SDS), 24–34
 multiple-path switches, 29–34
 single-path switches, 25–29
 time-division switching (TDS), 22–24
 shared-medium switch, 22–23
 shared-memory switch, 23–24
- call splitting, 20–21
- head-of-line blocking, 19
- internal link blocking, 17–18
- multicasting, 19–20
- output port contention, 18
- performance:
 input-buffered switches, 37–40
 output-buffered switches, 40–44
 shared-buffer switches, 44–46
- Parallel iterative matching (PIM),
 input-buffered switch, 58
- Parallel switching, input-buffered switch
 internal capacity, 54
- Path switch:
 configuration, 268–272
 heterogeneous capacity assignment, 274–277
 homogeneous capacity and route assignment, 272–274
- Patricia tree, internet protocol (IP) route lookups, 372
- Payload identifier, asynchronous transfer mode (ATM) networks, 427–428
- Peripheral component interconnect (PCI) bus,
 optical interconnection network (OIN), 302–303
- Permanent virtual connections (PVCs),
 asynchronous transfer mode (ATM) networks, 45, 407–410
- Personal communications service (PCS) access,
 wireless ATM (WATM) switches, 339
- Photonic packet switches, research issues, 279–280
- Physical medium-dependent (PMD) sublayer, SONET protocols, 423–425
- Ping-Pong arbitration unit (PAU):
 optical interconnection network (OIN):
 architecture, 309–315
 development, 302–303
 implementation, 318–321
 performance analysis, 318
 priority PPA, 321–324
 terabit router architecture, 303
- terabit IP router architecture, routing module and route controller, 307–308

- Pipelined schedulers, input-buffered switches, round-robin greedy scheduling (RRGS), 65–67
- Planar lightwave circuits (PLCs), optical interconnection network (OIN), 302–303
- Pointer index, internet protocol (IP) route lookups, forwarding table construction, 375–377
- Poisson distribution, input-buffered switches, performance evaluation, 38–40
- Portable base stations (PBSs), wireless ATM (WATM) switches, BAHAMA wireless ATM LAN, 343
- Power budget analysis, optical interconnection network (OIN), 326–328
- Precomputed 16-bit prefix table, internet protocol (IP) route lookups, multiway searching, 384–385
- Prefix tree, internet protocol (IP) route lookups, forwarding table construction, 373–377
- Prefix update algorithms, internet protocol (IP) route lookups, two-trie structure, 398–403
- PRESYNC state, 3M optical switch, cell delineation unit, 295–296
- Priority field (P):
 abacus switch, architecture, 191–193
 buffered multistage concentration network (BMCN), 216–217
 optical interconnection network (OIN), priority ping-pong arbitration, 321–324
- Priority ping-pong arbitration, optical interconnection network (OIN), 321–324
- Private network-network interface (PNNI)
 routing, asynchronous transfer mode (ATM) networks, 45, 407–410
- Push-in arbitrary out (PIAO) queue, input-buffered switch, output-queuing emulation, 73–74
- Push-in queue, input-buffered switch, output-queuing emulation, 73–74
- Quality-of-service (QoS) control:
 banyan-based switches, Sunshine switch, 113–114
 design and performance criteria, 13–14
 multiple-QoS scalable distributed-arbitration switch (SDA):
 performance analysis, 236–238
 structure, 234–236
 optical interconnection network (OIN), priority ping-pong arbitration, 321–324
 terabit switching technology, 2
- Quaternary phase-shift keying (QPSK), wireless ATM (WATM) switches, Olivetti's radio ATM LAN, 342
- Queue loss (QL), abacus switch performance, cell loss probability (CLP), 206–208
- Radio access layers, wireless ATM (WATM) switches:
 data link control layer, 346–347
 medium access control layer, 346
 physical layer, 344–346
- Radio physical layer, wireless ATM (WATM) switches, 344–346
- Radio physical medium dependent (RPMD)
 layer, wireless ATM (WATM) switches, 344–346
- Radio port identifier, wireless ATM (WATM) switches, radio access layers, 345–346
- Radio transmission convergence (RTC) layer, wireless ATM (WATM) switches, 344–346
- Random access memory (RAM):
 internet protocol route lookups, 368–369
 DIR-24-8-BASIC scheme, 378–381
 multiway binary search, cache exploitation, 385–388
 shared-memory switch, content-addressable memory (CAM) technique, 91–93
 wave-mux switch, 290–291
- Random early detection (RED), high-end routers, IP architecture, 11–12
- Random selection, input-buffered switches, scheduling algorithms, 57
- Random traffic model, input-buffered switch, 52
- READ process, shared-memory switch, linked list logical queues, 86–87
- Read sequence RAM (RSRAM), shared-memory switch, content-addressable memory (CAM) technique, 92–93
- Recirculation switches:
 buffering strategies, 35
 space-division switching (SDS) architecture, 33–34
- Recursive arbitration, optical interconnection network (OIN), ping-pong arbitration unit (PAU), 319–324
- Remote defect indication (RDI), SONET protocols, 419–421
- Request (REQ) signal:
 HYPASS optical switch, 285–286
 multiple-QoS scalable distributed-arbitration switch (SDA), 235–236

- Request (REQ) signal (*Continued*)
 scalable distributed-arbitration (SDA)
 switch, 229–231
 tandem-crosspoint (TDXP) switch:
 multicasting operation, 246
 unicasting operation, 242–245
 Resend signals, abacus switch, multicast
 contention resolution algorithm, 196–197
 Resequencing buffer (RSQB), enhanced
 abacus switch, 217–219
 Resequencing cells, enhanced abacus switch,
 217–219
 Reverse banyan network, concentration, 132
 Ring head-end (RHE), banyan-based switches,
 output contention resolution algorithms,
 110–112
 Roundoff procedure, Path switches, 275–276
 Round-robin arbitration:
 abacus switch:
 architecture, 193
 multicast contention resolution algorithm,
 195–197
 Clos network switches, concurrent
 round-robin dispatching, 263–268
 crosspoint-buffered switches, 228–229
 input-buffered switch:
 dual round-robin matching (DRRM),
 62–65
 iterative round-robin matching (iRRM),
 58–60
 iterative round-robin with SLIP (iSLIP),
 60–62
 parallel iterative matching (PIM), 58
 round-robin arbiters/selectors, 67–72
 bidirectional arbiter (NTT), 67–70
 token tunneling, 70–72
 round-robin greedy scheduling (RRGS),
 65–67
 scheduling algorithms, 57
 optical interconnection network (OIN),
 ping-pong arbitration unit (PAU),
 315–324
 scalable distributed-arbitration (SDA)
 switch, 231–233
 terabit IP router architecture, data packet
 flow, 305–306
 Round-robin greedy scheduling (RRGS),
 input-buffered switches, 65–67
 Route controller (RC):
 3M optical switch, 293–294
 optical interconnection network (OIN),
 terabit IP router architecture, 303,
 306–309
 Route decoder (RT DEC), shared-memory
 switch, linked list logical queues, 86–90
 Route lookups, internet protocol (IP):
 caching technique, 369
 design issues, 368–369
 gigabit switch routers, 388–396
 algorithms and data structures, 388–395
 CBM/CNHA construction, 393–395
 NHA construction algorithm, 392–393
 performance analysis, 395–396
 hardware, memory access speeds, 377–381
 multiway search, 381–388
 binary search, best-matching prefix,
 381–384
 cache line exploitation, 385–388
 performance analysis, 388
 precomputed 16-bit prefix table, 384–385
 Patricia tree, 372
 research issues, 365–366
 small forwarding tables, fast lookups,
 373–377
 standard trie structure, 369–372
 two-trie structure, 396–404
 AddPrefix (X,Y,Z) algorithm, 399–402
 DelPrefix (X,Y) algorithm, 402–403
 IPLookup (X) algorithm, 397–398
 performance analysis, 403–404
 prefix update algorithm, 398–399
 Routing algorithms:
 Clos network switches, 255–257
 suboptimal straight matching method,
 258–259
 internet protocol route lookups:
 gigabit switch routers, 388–396
 algorithms and data structures, 388–395
 CBM/CNHA construction, 393–395
 NHA construction algorithm, 392–393
 two-trie structure, 396–404
 AddPrefix (X,Y,Z) algorithm, 399–402
 DelPrefix (X,Y) algorithm, 402–403
 IPLookup (X) algorithm, 397–398
 performance analysis, 403–404
 prefix update algorithm, 398–399
 Path switch, 269–272
 Path switching, 272–274
 Routing delay, enhanced abacus switch,
 memoryless multistage concentration
 network, 212–214
 Routing information base (RIB), optical
 interconnection network (OIN), terabit IP
 router architecture, 308–309
 Routing information table (RIT),
 asynchronous transfer mode (ATM)
 networks, 35
 Routing module (RM):
 abacus switch:
 architecture, 190–193

- enhanced configuration, 211–220
- multicast contention resolution algorithm, 194–197
- optical interconnection network (OIN), 302–303
 - terabit IP router architecture, 303
 - terabit IP routers, optical interconnection network, 301–303
 - terabit IP routers, optical packet switches, optical interconnection network (OIN), 306–309
- Running adder network (RAN), banyan-based switches, 125
 - concentration, 132
 - cyclic running adder network (CRAN), 135–138
 - encoding process, 129–132
 - overflow fairness, 134–138
- SAMBA system, wireless ATM (WATM) switches, 344
- Scalable distributed-arbitration (SDA) switch:
 - performance analysis, 231–233
 - structure, 229–231
- Scheduling algorithms:**
 - Clos network switches, 255–257
 - suboptimal straight matching method, 258–259
 - input-buffered switch:
 - dual round-robin matching (DRRM), 62–65
 - iterative round-robin matching (iRRM), 58–60
 - iterative round-robin with SLIP (iSLIP), 60–62
 - parallel iterative matching (PIM), 58
 - round-robin arbiters/selectors, 67–72
 - bidirectional arbiter (NTT), 67–70
 - token tunneling, 70–72
 - round-robin greedy scheduling (RRGS), 65–67
 - input-buffered switches, 57–72
- Scheduling efficiency, input-buffered switches:
 - head-of-line (HOL) blocking model, 51
 - virtual output queue (VOQ)-based matching, 55–57
 - window-based lookahead selection, 54–55
- Scrambling procedures, SONET protocols, 417–418
- Segment header processor (SHP), terabit IP router architecture, routing module and route controller, 308
- Self-routing algorithm, broadcast banyan network (BBN), multicast copying, 127–128
- Self-routing methods:**
 - ATM switch structure, 79
 - banyan-based switches, 29
 - crossbar switches, 26
- Semiconductor optical amplifier (SOA):
 - optical ATMOS switch, 282–283
 - optical interconnection network (OIN), 309–315
 - complexity comparisons, 324–326
 - crosstalk analysis, 328–331
 - input optical module (IOM), 310
 - output optical module (OOM), 311
 - tunable filters, 312–315
- Service-time distribution, input-buffered switches, performance evaluation, 39–40
- Shared-buffer memories (SBMs):
 - multicast shared-memory switch, 99–101
 - space-time-space (STS) approach, 93–94
- Shared-buffer switches:
 - buffering strategies, 36
 - performance evaluation, 44–46
- Shared-medium switch, architecture, 22–23
- Shared-memory switch:
 - architecture, 23–24
 - content-addressable memory technique, 91–93
 - linked list technique, 84–90
 - memory access cycle, 83
 - multicast switches, 96–101
 - address copy, 99–101
 - cell copy, 98–99
 - logical queuing, 97–98
 - multistage switches, 94–96
 - concentrator-based growable switch
 - architecture, 96
 - Washington University gigabit switch, 94–96
 - research issues, 83–84
 - space-time-space (STS) approach, 93–94
 - wireless ATM (WATM) switches,
 - mobility-support ATM switch, 354–358
- Shuffle-exchange network (SN), deflection routing, 117–118
 - dual SN, error correction, 118–125
- Simple network management protocol (SNMP), asynchronous transfer mode (ATM) network protocol, 410
- Simulated traffic model, wireless ATM (WATM) switches, mobility-support ATM switch, 358–362
- Single-path switches, space-division switching (SDS), 25–29
 - Banyan-based switches, 28–29

- Single-path switches (*Continued*)
 crossbar switches, 25–27
 fully interconnected switches, 27, 29
- Single-stage knockout switch, architecture, 142–143
- Single tree-structured competition,
 single-stage knockout switch, 148–150
- Skip value, internet protocol (IP) route lookups, Patricia tree structure, 372
- Slackness, input-buffered switch,
 output-queuing emulation, 74
- Small switch modules (SSMs):
 abacus switch:
 architecture, 190–193
 performance delay, 204–205
 enhanced abacus switch, resequencing cells, 219
- Sorting network, banyan-based switches,
 batcher-sorting network, 106–109
- Space-division multiplexing (SDM), optical packet switches, 280
- Space-division switching (SDS):
 architecture, 24–34
 multiple-path switches, 29–34
 augmented banyan switches, 30
 multiplane switches, 33
 recirculation switches, 33–34
 three-stage Clos switches, 30–33
- single-path switches, 25–29
 Banyan-based switches, 29–30
 crossbar switches, 25–27
 fully interconnected switches, 27, 29
- Space switches (SWi), 3M optical switch, 292–294
- Space-time-space (STS) approach,
 shared-memory switch, 93–94
- Speedup factor:
 input-buffered switch internal capacity, 54
 terabit IP router architecture, 303–304
- Staggering switch, properties, 281–282
- Starting copy number (SCN), banyan-based switches, cyclic running adder network (CRAN), 136–138
- STAR-TRACK switch, configuration, 286–287
- Strict priority, multicast shared-memory switch, 97–98
- Strict-sense (SS) call splitting, asynchronous transfer mode (ATM) switches, 20–21
- Sunshine switch, banyan-based switches, 112–114
- SWAN prototype system, wireless ATM (WATM) switches, 343
- Switched virtual connections (SVCs), asynchronous transfer mode (ATM) networks, 45, 407–408
- Switch element (SWE):
 abacus switch:
 architecture, 191–193
 ATM routing and concentration (ARC)
 chip, 208–211
 enhanced configuration, 211–220
 fault-tolerant multicast output-buffered
 ATM switch, 169–172
 cross-stuck (CS) fault, 170–171
 performance analysis, 181–185
 toggle-stuck (TS) fault, 171–172
 vertical/horizontal-stuck (VS/HS) fault, 172
 multicast grouping networks (MGNs), 157–160
- Switch fabric, high-end IP routers, 12–13
- Switching elements (SEs), Washington University gigabit switch (WUGS), 95–96
- Switching module (SWM), wave-mux switch, 290–291
- Switch module (SM):
 fault-tolerant multicast output-buffered
 ATM switch, performance analysis, 181–185
 two-stage multicast out-put-buffered
 ATM switch (MOBAS), cell loss rates, 163–169
- Switch priority (SP), banyan-based switches, Sunshine switch, 113–114
- Synchronization, 3M optical switch, cell synchronization unit, 297–301
- Synchronous Optical Network (SONET):
 asynchronous transfer mode (ATM), 2
 ATM switch structure, 58
 3M optical switch, 291–294
 protocols:
 automatic protection switching (APS), 419–421
 background, 407–409
 frequency justification, 418–419
 OC- N multiplexer, 422–423
 overhead bytes, 414–417
 reference model, sublayer functions, 423–425
 scrambling and descrambling, 417–418
 STS-N signals, 412–414
 STS-3 vs. STS3c, 421–422
 sublayer protocols, 410–412
- Synchronous status message function, SONET protocols, 416
- Synchronous transfer signals (STS), SONET protocols, 412–414
 frequency justification, 417–418
 STS-3 vs. STS-3c, 421–422

- SYNC state, 3M optical switch:
 cell delineation unit, 295–296
 VCI overwrite unit, 296–297
- Tail pointer register (TPR), shared-memory switch, linked list logical queues, 86–90
- Tail pointer (TP):
 shared-memory switch, 85–90
 wireless ATM (WATM) switches,
 mobility-support ATM switch, 356–358
- Tandem banyan switching fabric (TBSF),
 deflection routing, 114–117
- Tandem connection monitoring (TCM),
 SONET protocols, 417
- Tandem-crosspoint (TDXP) switch:
 architecture, 241–242
 input-output-buffered switches, 239–241
 multicasting operation, 246
 performance analysis, 246–251
 research issues, 239
 unicasting operation, 242–245
- TBL24 entry format, internet protocol route lookups, DIR-24-8-BASIC scheme, 378–381
- TBLLong entry format, internet protocol route lookups, DIR-24-8-BASIC scheme, 378–381
- Terabit switching technology:
 link utilization, 2
 optical interconnection network, IP routers:
 complexity issues, 324–326
 crosstalk analysis, 328–331
 network configuration, 309–315
 ping-pong arbitration (PPA) unit, 315–324
 power budget analysis, 326–328
 research issues, 301–303
 router module and route controller, 306–309
 terabit architecture, 303–306
 data packet flow, 305–306
 speedup, 303–304
- Three-phase algorithm, banyan-based switches, batcher-sorting network, 109–110
- Three-stage Clos switches, space-division switching (SDS) architecture, 30–33
- Throughput limitation:
 input-buffered switch, 49–50
 traffic models, 52–53
 knockout-based switches, channel grouping, maximum throughput, 150–152
- Time-division multiplexing (TDM), optical packet switches, 280
- Time-division switching (TDS):
 shared-medium switch, 22–23
 shared-memory switch, 23–24
- Time-space-time (TST) switches, Clos network switches, 256–257
- Time to leave (TL) urgency, input-buffered switch:
 most-urgent-cell-first algorithm (MUCFA), 72–73
 output-queuing emulation, 74
- Time-to-live (TTL) field, internet protocol route lookups, 365–366
- Toggle-stuck (TS) fault, fault-tolerant multicast output-buffered ATM switch:
 fault detection, 172–174
 location and configuration, 175–177
 performance analysis, 182–183
 switch element (SWE), 171–172
- Token tunneling, input-buffered switch, 70–72
- Total cell loss rate, two-stage multicast out-put-buffered ATM switch (MOBAS), 169–172
- Traffic models:
 input-buffered switches, 52–53
 wireless ATM (WATM) switches,
 mobility-support ATM switch, 358–362
- Translation tables, two-stage multicast output-buffered ATM switch, 160–163
- Transmission convergence (TC) sublayer, SONET protocols, 423–425
- Trie structure, internet protocol route lookups:
 standard trie structure, 369–372
 two-trie structure, 396–404
 AddPrefix (X,Y,Z) algorithm, 399–402
 DelPrefix (X,Y) algorithm, 402–403
 IPLookup (X) algorithm, 397–398
 performance analysis, 403–404
 prefix update algorithm, 398–399
- Trunk number translator (TNT), banyan network switches, multicast copy, 125–126
 encoding process, 129–132
- Tunable filters, optical interconnection network (OIN), 311–315
- Two-stage multicast out-put-buffered ATM switch, 154–169
 multicast grouping network, 157–160
 multicast knockout principle, 163–169
 translation tables, 160–163
 two-stage configuration, 154–157
- Two-trie structure, internet protocol (IP) route lookups, 396–404
 AddPrefix (X,Y,Z) algorithm, 399–402
 DelPrefix (X,Y) algorithm, 402–403

- Two-trie structure (*Continued*)
 IP Lookup (X) algorithm, 397–398
 performance analysis, 403–404
 prefix update algorithm, 398–399
- Unfairness:
 banyan-based switches, overflow and input fairness, 134–138
 scalable distributed-arbitration (SDA) switch, 231–233
- Unicasting operation:
 asynchronous transfer mode (ATM) networks, 35
 tandem-crosspoint (TDXP) switch, 242–245
- Uniform source distribution, wireless ATM (WATM) switches, mobility-support ATM switch, 360–362
- Unshuffle-exchange network (USN), deflection routing, 118–125
- Unspecified bit rate (UBR):
 asynchronous transfer mode protocols, 425–426
 multiple-QoS scalable distributed-arbitration switch (SDA), 234–236
 wireless ATM (WATM) switches, data link control layer, 346–347
- User network interface (UNI), asynchronous transfer mode (ATM) networks, 35, 407–410
- User plane (U-plane), asynchronous transfer mode (ATM) network protocol, 409
- Variable-length data packets, high-end routers, IP architecture, 12
- Vertical/horizontal-stuck (VS/HS) fault, fault-tolerant multicast output-buffered ATM switch:
 fault detection, 173–174
 location and configuration, 177–181
 performance analysis, 183–185
 switch element (SWE), 172
- Virtual channel identifier (VCI):
 abacus switch, input port controller implementation, 198
 asynchronous transfer mode (ATM) networks, 35
 protocols, 426–427
 ATM switch structure, 79
 3M optical switch, 291–294
 cell delineation unit, 294–296
 overwrite unit, 296–297
 two-stage multicast out-put-buffered ATM switch (MOBAS), 154–157
- translation tables, 161–163
- wireless ATM (WATM) switches:
 BAHAMA wireless ATM LAN, 343
 handoff process, 348–350
 mobility-support ATM switch, 353–362
- Virtual connection tree, wireless ATM (WATM) switches, 342–343
 handoff process, 348–350
- Virtual-output-queuing (VOQ) switches:
 buffering strategies, 36–37
 continuous round-robin dispatching (CRRD), 265–267
 input-buffered switch:
 dual round-robin matching (DRRM), 62–65
 most-urgent-cell-first algorithm (MUCFA), 73
 parallel iterative matching (PIM), 58
 input-buffered switches, matching, scheduling efficiency, 55–57
- Virtual path capacity allocation (VPCA), Path switches, 274–275
- Virtual path identifier (VPI):
 abacus switch, input port controller implementation, 198
 asynchronous transfer mode (ATM) networks, 35
 protocols, 426–427
 ATM switch structure, 79
 3M optical switch, 291–294
 VCI overwrite unit, 296–297
- wireless ATM (WATM) switches,
 BAHAMA wireless ATM LAN, 343
- Virtual queues, abacus switch, maximum throughput performance, 201–203
- Washington University gigabit switch (WUGS), multistage shared-memory applications, 94–96
- Waveguide grating router (WGR), 3M optical switch, 293–294
- Wavelength converters (WCs), 3M optical switch, 292–294
- Wavelength division-multiplexing (WDM):
 3M optical switch, 292–294
 optical interconnection network (OIN), 302–303
 crosstalk analysis, 329–331
 optical packet switches, 279–280
- Wave-Mux switch, architecture, 290–291
- WDM ATM switch. See 3M optical switch
- Weighted round-robin greedy scheduling (WRRGS), input-buffered switches, 67
- Wide-sense (WS) call splitting, asynchronous

- transfer mode (ATM) switches, 20–21
- Window-based lookahead selection,
 - input-buffered switches, scheduling efficiency, 54–55
- Windowing, input-buffered switch, buffering strategies, 35–36
- Winner output, single-stage knockout switch, concentrator construction, 146–150
- Wireless access layer (WAL), wireless ATM (WATM) switches, 340–341
- Wireless ad hoc networks, wireless ATM (WATM) switches, 338
- Wireless ATM (WATM) switches:
 - BAHAMA wireless ATM LAN, 343
 - European projects, 343–344
 - handoff, 347–352
 - buffering, 350–351
 - connection rerouting, 348–350
 - COS cell routing, 351–352
 - mobility-support ATM switch, 352–362
 - design issues, 353–358
 - performance analysis, 358–362
 - NEC WATMnet prototype system, 341–342
- NTT's wireless ATM access, 343
- Olivetti's radio ATM LAN, 342
- radio access layers:
 - data link control layer, 346–347
 - medium access control layer, 346
 - physical layer, 344–346
- research issues, 337–338
- structural overviews:
 - protocols, 340–341
 - system components, 338–340
 - virtual connection tree, 342–343
- Wireless physical layer (PHY), wireless ATM (WATM) switches, NEC WATMnet prototype system, 341–342
- WRITE process, shared-memory switch:
 - linked list logical queues, 86
 - space-time-space (STS) approach, 93–94
- Write sequence RAM (WSRAM),
 - shared-memory switch, content-addressable memory (CAM) technique, 91–93
- Y-junction switches, 3M optical switch cell synchronization unit, 298–301