# Switching Algorithms and Buffer Management in Asynchronous Transfer Mode Networks

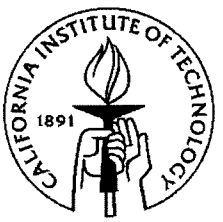
Thesis by

Bahadir Erimli

In Partial Fulfillment of the Requirements

for the Degree of

Engineer




California Institute of Technology

Pasadena, California

1996

(Submitted February 22, 1996)

in memory of,

Professor Edward C. Posner

# Acknowledgements

First of all, I would like to express my sincere gratitude for my advisor, Professor Robert J. McEliece, who with his knowledge and wisdom provided me with crucial guidance and made this work possible. It was a great pleasure and opportunity to have him as my mentor and work with him during these past years. His valuable advice and enthusiastic encouragement provided me inspiration and made this research a great learning process.

I also would like to thank my colleagues Dr. Kuo-Hui Liu from Pacific Bell who initiated the idea for the first part of the research covered in chapter 3 and Dr. John Murphy who introduced me to the concept in chapter 4 and with whom I co-authored a paper on some parts of that chapter. It was a great pleasure to work with both of them. Moreover, many thanks to Dr. Laif Swanson for providing us the proof of Theorem 3.4 during a lunchtime at JPL.

Furthermore, I thank the members of my defense committee, Professors R. J. McEliece, Y. S. Abu-Mostafa, and P. P. Vaidyanathan.

During my research, I was financed with a grant from Pacific Bell and I commend them for their interest in new technologies and advancement of telecommunications science and thank them for their support.

Professor Edward C. Posner is one other person who earned my deep respect. Words fall short of describing the sorrow and shock everyone experienced when an unfortunate accident took him away from us. However, he will always be a part of us and his memory and excellent work will not be forgotten.

My groupmates and friends also deserve my gratitude for the support they provided. I would like to mention the names Dr. Sanjeev K. Deora, Zhong (John) Yu, Jung-Fu (Tommy) Cheng, Zehra Cataltepe, Hongyu (Sally) Piao, and Paul LeMahieu (some soon-to-be doctors). Robert Freeman, our system administrator and my friend, deserves special recognition for keeping our computers up and running all the time

with his hard work and continuous extras hours he puts in with utmost self-sacrifice. My good friends Fabienne Breton, Laure Granie, and Ashish Bansal created a great environment with their irreplaceable, priceless friendship. Parandeh Kia provided priceless support with her friendship and wisdom.

Finally I would like to express my greatest appreciation to those people whom I'm deeply indebted to: My mother and father, my grandparents, and my beloved fiancé, Svetlana Lyapina. They encouraged me every step of the way; they have always been there for me in my times of need, during hardships, during all the rough times. Their optimism, good will, and love kept me going. I can't imagine it without their support, love, and affection. Thank you all for everything.

# Abstract

In this thesis, two different but related concepts in Asynchronous Transfer Mode (ATM) are discussed. Due to its multirate nature, ATM creates new problems in terms of switching and buffering. In the first part, the switching problems are investigated. The situation is rooted upon the multirate connections in a 'circuit-switching-like' environment. The multirate nature of ATM results in the loss of strictly nonblocking three stage space-division switches unless a 'speed-up factor' is provided between the outside ports and the internal links of the switch. To keep this factor to a minimum, call routing algorithms are considered as a possible solution. Several call routing algorithms are compared in terms of their blocking probability under various circumstances. A simple algorithm, named fixed priority routing algorithm, stands out among these, both in terms of simplicity and low blocking rate. Afterwards a bin packing model is used to investigate the reasons behind this.

In the second part, buffer management in ATM nodes is considered. In the traditional sense, the burstier the traffic is, the higher, it was believed, the cell loss will be at a buffer into which a number of these sources are transmitting. It is shown that this is not always the case and under the circumstances defined – the worst-case model – other types of sources that output traffic that is less bursty might create higher cell loss than burstier sources. All sources considered are leaky-bucket controlled and stay within their contract limits with the network at all times. Initially greedy on-off source and the three-state source types are compared. After establishing that the comparison between these two in terms of cell loss rate is highly dependent on the size of the buffer being transmitted onto, other source types that might create even higher cell loss rates are searched for. One such characteristic group of sources is found and is presented.

# Contents

# List of Figures

# List of Tables

# Chapter 1 Introduction

With the rapid advance of communication technologies and on-line services, the last decade observed skyrocketing bandwidth demands (Fig. 1.1) [23]. With the emergence of new technologies, current telecommunication networks evolved towards the Integrated Broadband Communication Network or as it is called these days, the Broadband Integrated Services Digital Network (BISDN). The direction taken recently by BISDN is influenced by a number of parameters. The emergence of a large number of communication services with widely varying and sometimes unknown requirements is one of the most important.

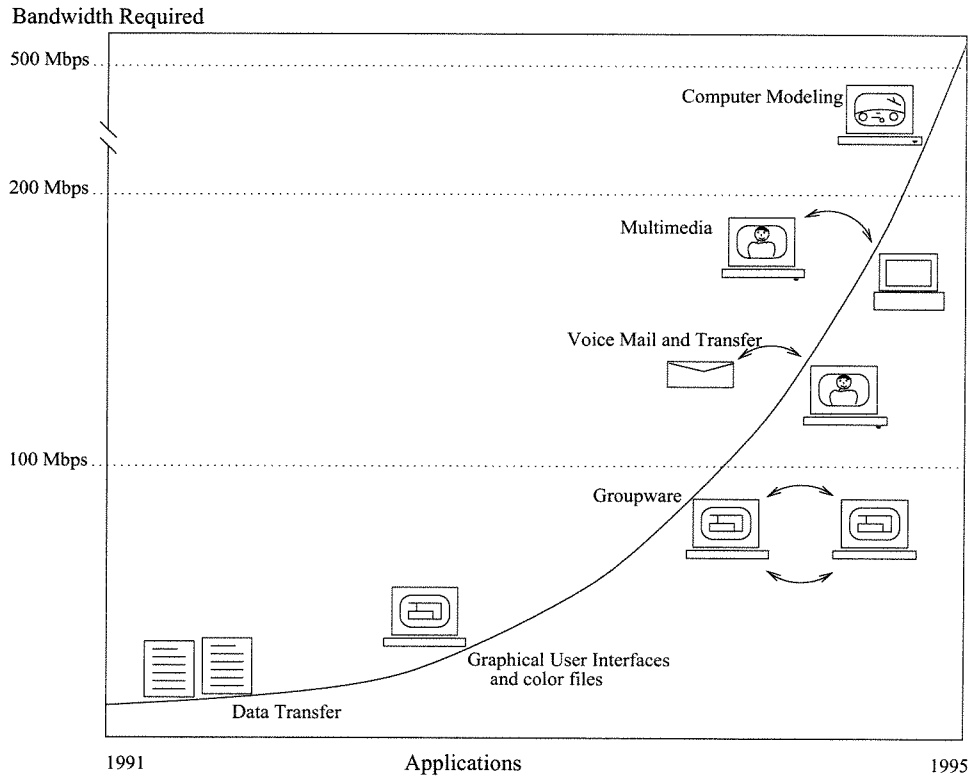



Figure 1.1: Emerging Technologies

Some of these new applications are video-on-demand, voicemail, video conferencing, videophony, high-speed data transfer, high-definition TV (HDTV), on-line shop-

ping and educational services, on-line libraries, and networked parallel high-speed computers. Each of these services have different requirements of bandwidth, delay, jitter, error, loss, and quality of service (QoS) from the network. Furthermore, these services will exist on different network levels, varying all the way from local-area networks (LAN) to wide-area networks (WAN) and backbone networks (Fig. 1.2).

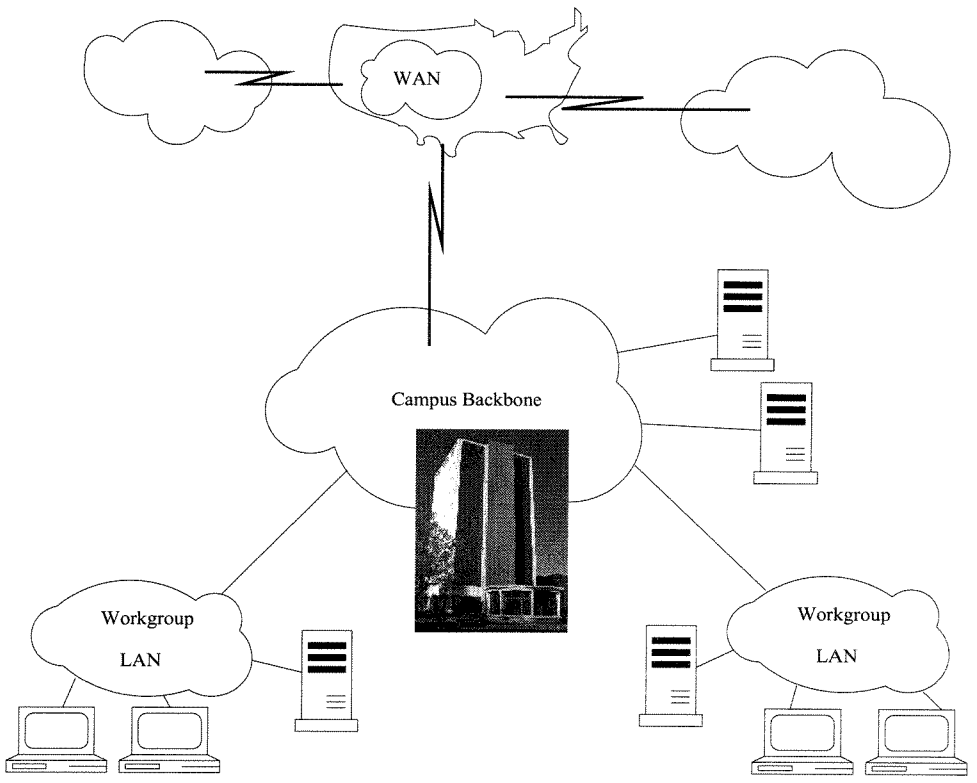



Figure 1.2: Global Network

Under these circumstances, there are a number of services that the network is required to offer: high bandwidth and bandwidth on demand; a variety of QoS level parameters and guaranteed service levels; point to point, point to multipoint, and multipoint to multipoint connections; continuous and variable bit services; connection oriented and connectionless services; swift adaptation to varying parameters on the network. Among the possible methods for time division multiplexing for multi-rate services, not all of them can provide many of these necessary characteristics efficiently [33]. In fact, most fall far short of accomplishing the goal due to the vast differences in the parameters of various services and the bursty nature of the overall

– and also single-source – traffic. Those that satisfy the multi-rate concept, though, fail in some other aspect; complexity, flexibility, reliability, etc. (Fig. 1.3).

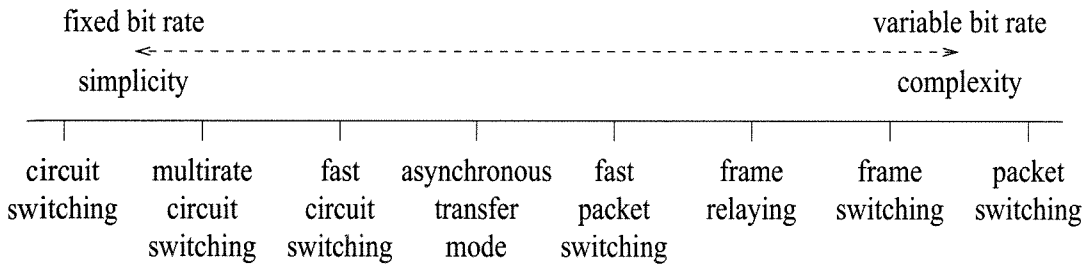



Figure 1.3: Switching Techniques

As an example, one can consider the one that's been in use the longest: Circuit switching or time domain multiplexing. In this technique, since every source has a single spot allocated, there is no allowance for sources with different transmission speeds, let alone a source with a varying speed. Its more modern version, multi-rate TDM, can accommodate sources with different speeds up to a certain degree, but is still too restrictive for general purposes (Fig. 1.4). The other end of the spectrum, packet switching, though, is too unreliable in terms of delays due to its connectionless nature[1].

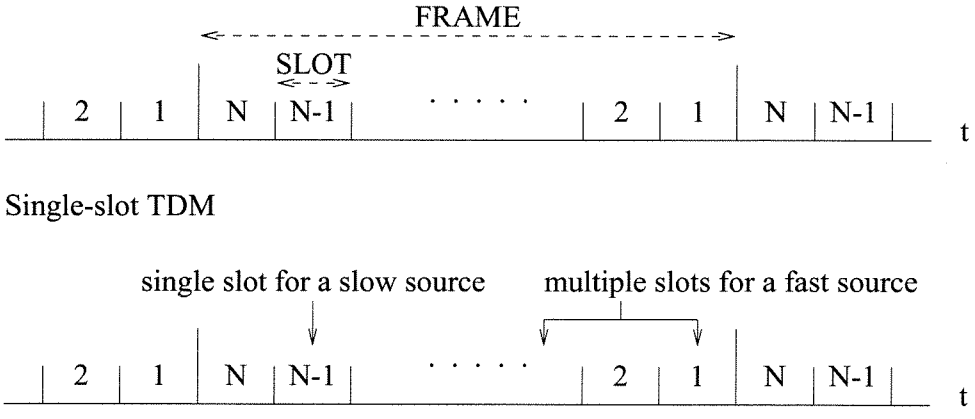



Figure 1.4: TDM (Circuit Switching) and Multirate TDM

[1]In connection-oriented traffic, a link between the source and the destination is established before transmission begins. In connectionless traffic, there is no established link between the two and the source simply transmits into the network with the address of the destination contained in the message.

Among the various modes of transmission, Asynchronous Transfer Mode (ATM) was selected by CCITT (currently ITU-T) in 1987 to be the transfer mode of the future B-ISDN. Since then CCITT issued many recommendations, specifying the details of ATM for use in B-ISDN. In addition, the ATM Forum was formed by companies in the field and research institutions, including many universities and this organization has been working to define the necessary specifications for ATM operation and interoperability.

ATM is an attempt to meet all of the properties that the transfer mode for B-ISDN was envisioned to have. Namely, supporting all existing services as well as any possible future ones with yet-unknown characteristics; utilizing network resources as efficiently as possible; minimizing switching complexity; minimizing processing time at the intermediate nodes (to support very high transmission speeds); minimizing the amount of buffering at the intermediate nodes (to bound transmission delays and reduce buffer management complexity); guaranteeing performance requirements of existing and expected applications.

ATM's power, economy, scalability, and flexibility optimize it for such a role. In essence, it is a type of virtual-circuit packet switching with fixed size packets. It has various features that, by incorporating the most desired features of circuit switching, extend the capabilities of current packet-switching networks to support real-time traffic in an efficient manner.

Of course, since ATM is new, it comes with issues that are different from the ones that apply to other previous systems. In this thesis, two of these problems will be considered. The second chapter will introduce ATM in detail and provide a background for the problems that will be in the later chapters. Chapter 3 will be dedicated to the switching issues in ATM. Since the multi-rate nature of ATM doesn't allow the existence of strictly nonblocking space-division switches, the effort will go into trying to minimize the blocking probability through the utilization of clever call-routing algorithms within the switch. A number of call-routing algorithms will be compared in performance, with call-blocking being the main criteria. In the fourth chapter, a buffer management issue will be discussed. For buffers being fed by

multiple leaky-bucket controlled sources, the worst-case situations will be considered. The last chapter will include a conclusion for these issues.

# Chapter 2 Asynchronous Transfer Mode – A New Technology

Asynchronous Transfer Mode is enabling the fast introduction of a wide range of advanced high-speed communications applications. ATM will provide a forward-looking investment in the networks required to respond to tomorrow's service and revenue opportunities. By offering scalable rates from 1.5 Mbps to 155 Mbps and higher, according to customer's needs, ATM will make the WAN diminish in these applications. On the other hand, in contrast to frame relay or other similar data services, ATM will also accommodate delay sensitive traffic such as voice and video easily, creating the foundations for emerging multimedia applications. Because of its networking flexibility, open-ended nature, and potential to gather widely different applications under a common roof, ATM has gained broad based support.

| | | |
|---|---|---|
| STS-1/OC-1 | 51.84 Mbps | 28 DS1s or 1 DS3 |
| STS-3/OC-3 | 155.52 Mbps | 3 STS-1s, byte interleaved |
| STS-3c/OC-3c | 155.52 Mbps | Concatenated, indivisible payload |
| STS-12/OC-12 | 622.08 Mbps | 12 STS-1s, 4 STS-3cs, or any combination |
| STS-12c/OC-12c | 622.08 Mbps | Concatenated, indivisible payload |
| STS-48/OC-48 | 2488.32 Mbps | 48 STS-1s, 16 STS-3cs, or any combination |

Table 2.1: SONET Hierarchy

ATM is an outgrowth of B-ISDN standards and ATM traffic is intended to be carried on the synchronous optical network (SONET) (Table 2.1 [22]). It is one of the general class of packet technologies that relay traffic with respect to an address contained within the packet itself. Unlike some other packet technologies like X.25 and frame relay, ATM uses very short, fixed-length packets called cells.

## 2.1 The ATM Cell

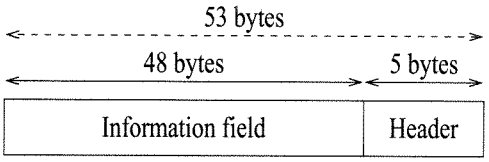



Figure 2.1: ATM Cell

The ATM cell (Fig. 2.1) is 53 bytes long, with a five-byte header and a 48-byte information field. According to ITU-T recommendation, the bytes are sent in the increasing order and the bits are sent in the decreasing order (Fig. 2.2). In all the fields, the first bit to be sent is the most significant bit [18].

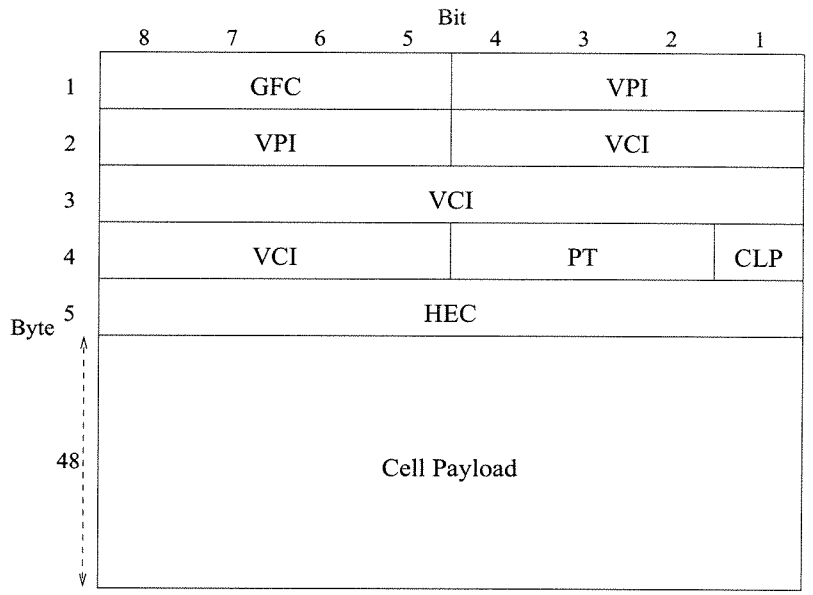



Figure 2.2: ATM Cell (UNI)

At the User-Network Interface (UNI), the header contains 4 bits for Generic Flow Control (GFC), 8 bits for Virtual-Path Identifier (VPI), 16 bits for Virtual-Channel Identifier (VCI), 3 bits for Payload Type (PT), one bit for Cell Loss Priority (CLP), and 8 bits for Head Error Control (HEC) [29]. At the Network-Node Interfaces (NNI),

GFC does not exist and its four bits are used to expand the VPI field to 12 bits. In some cases, up to four bytes from the information field can be taken up from the information field, depending on the ATM Adaptation Layer (AAL).

GFC provides flow control at the UNI for the traffic coming from the user and going into the network (but does not control traffic in the opposite direction). The GFC field has no use inside the network because the same task is supposed to be handled by other mechanisms, so this field is used by the VPI to enhance virtual-path addressing. The next two fields, VPI and VCI provide the ATM address. PT indicate the type of information carried by the cell. This is to identify various types of user data and also transfer operations and maintenance messages across the network, between users and between user and service provider. The CLP bit tags the cell as either high priority (CLP = 0) or eligible to be discarded (CLP = 1). The final byte, HEC, is the header error control field. It can detect multiple bit errors and correct single bit errors.

## 2.2 The Layered Model

A logical hierarchical architecture is used for ATM B-ISDN network (Fig. 2.3). The model uses the concept of separated planes for the segregation of user, control, and management functions [18], [25], [22].

### 2.2.1 Physical Layer

The physical layer transports the ATM cells between points. It is composed of two sublayers, the Physical Medium (PM) that supports the bit-transmission capabilities, including the generation and reception of suitable waveforms, insertion and extraction of symbol timing information, and electrical-optical and optical-electrical transformations and the Transmission Convergence (TC) that converts the ATM stream into bits to be transported over the PM, including tasks of packing cells into appropriate PM format and insertion of idle cells.

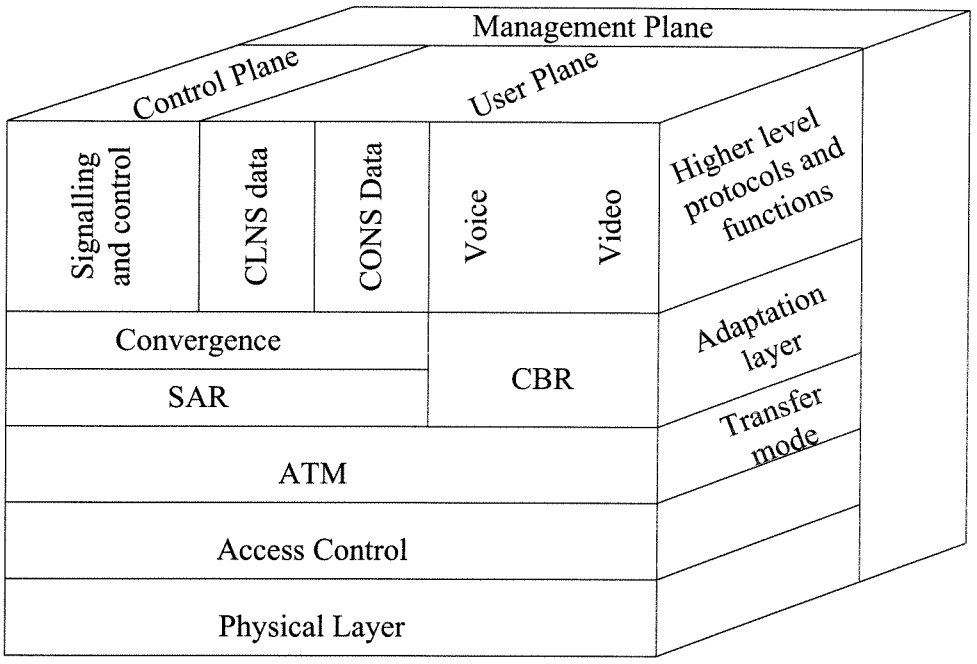



Figure 2.3: B-ISDN ATM Protocol Reference Model

Basically, the physical medium sublayer is responsible for the correct transmission and reception of the bits. It is medium dependent (optical, electrical, etc.). Since it is also responsible for bit timing, the transmitting entity needs to include the proper bit timing information and line coding.

In the transmission convergence sublayer, cell delineation, recognizing the cell boundaries, is a major task. For this purpose, the HEC field of the ATM cell is used. The incoming stream is monitored for a 5-byte chunk with the correct CRC-8. Whenever one is found, the monitoring is continued for another predetermined number of cells to make sure that it really is a header. If this also is successful, a correct cell header is assumed to be detected and the cell separations are defined accordingly. The device stays in synch until a predetermined number of 5-byte assumed-headers with errors are detected. This feature is also enhanced by scrambling the information field to prevent any part of it from being misidentified as a header. Furthermore, since once a header is detected, the system expects cells to continue coming in 53-byte intervals, rate decoupling by insertion of empty cells is performed at this level.

There are different interfaces defined for possible transmission systems. These

| | |
|---|---|
| Transmission Convergence | Cell rate decoupling (with IDLE cells)<br>Header error check (HEC) generation/verification<br>Cell Scrambling/Descrambling<br>Cell delineation (using HEC)<br>Path signal identification<br>Frequency justification<br>Frame scrambling/descrambling<br>Frame generation/recovery |
| Physical Medium | Bit timing<br>Line coding<br>Physical medium dependent scrambling/descrambling |

Table 2.2: Physical Layer Sublayers and Functions

are SONET STS-3 interface for Synchronous Digital Hierarchy (SDH) and DS3, 100 Mbps multimore fiber, and 155 Mbps multimode fiber.

### 2.2.2 ATM Layer

The ATM layer, with characteristics fully independent of the physical medium used, is the boundary between the functions related to the header and the functions to do with the information field.

| |
|---|
| Generic flow control |
| Cell VPI/VCI translation |
| Cell multiplex/demultiplex |
| Cell rate decoupling (with UNASSIGNED cells) |

Table 2.3: ATM Layer Functions

This layer provides the translation of the cell identifier (either only separately on VPI or VCI or on both), required for demultiplexing/multiplexing or switching, and performs the former. This layer is also where the cell header gets either included or stripped off the cell before the cell is either received from or sent to the ATM adaptation layer. Furthermore, this layer also performs management functions: It provides the user with one of the available QoS classes. It monitors and implements

the flow control mechanism, making sure that the connections stay within limits and are not affected by connections that do not. Furthermore, it monitors the network for congestion indication – whether it be from the header of an information cell or from a management cell – and generates management cells accordingly for congestion control (Rate-based Flow Control, Fig. 2.4, is the ATM Forum standard [21]). Other than traffic management, some fault management is also implemented at this level. This includes invalid VPI/VCI detection, connectivity verification, and alarm surveillance for detection of VP/VC errors and generation and propagation of relevant messages.

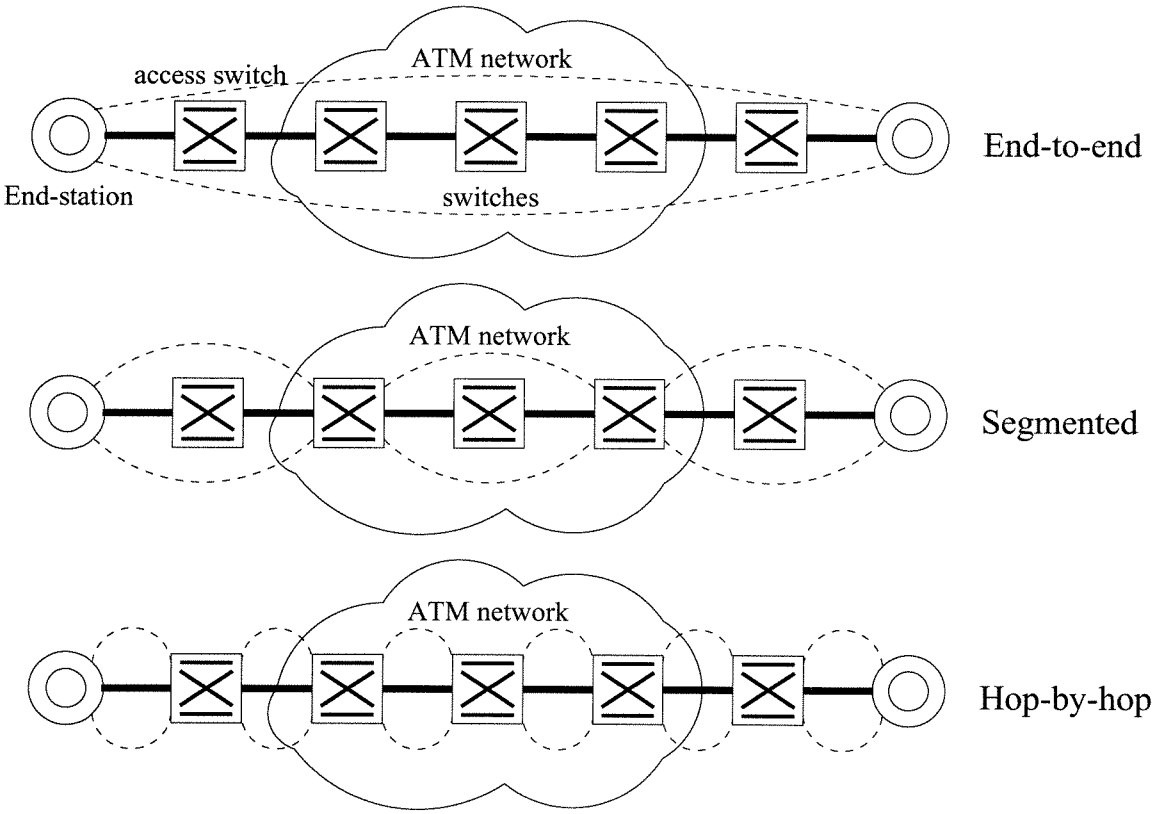



Figure 2.4: Rate-Based Flow Control Types

### 2.2.3 ATM Adaptation Layer

The ATM adaptation layer (AAL) is certainly the most important feature of the process. It is this level that provides ATM the versatility to carry many different types of service from continuous processes like voice to the highly bursty transmis-

sions generated by LANs within the same format by providing each service class the functionalities required in reaching its desired quality of service. Since AAL is not a network process but is performed by end-user equipment, it frees the network from worrying about different types of traffic.

AAL supports higher layer functions of the user, control, and management planes and supports connections between ATM layer and the higher level non-ATM layers. Data received from higher level protocols is segmented or collected (according to the class of information) to be placed into ATM cells. Cells received from the ATM layer are reassembled to be forwarded in the proper format to higher layers.

AAL layer consists of two sublayers: segmentation and reassembly (SAR) and convergence layer (CS). The intent of these sublayers is to convert whatever type of data is to be transmitted into 48-byte payloads while maintaining the integrity and a certain amount of identity of the data involved. The result of each sublayer's process is called a protocol data unit (PDU). The CS-PDU is variable length, according to the particular AAL and/or the length of the higher layer data block provided. The SAR-PDU is always 48 bytes long to fit in a cell payload field.

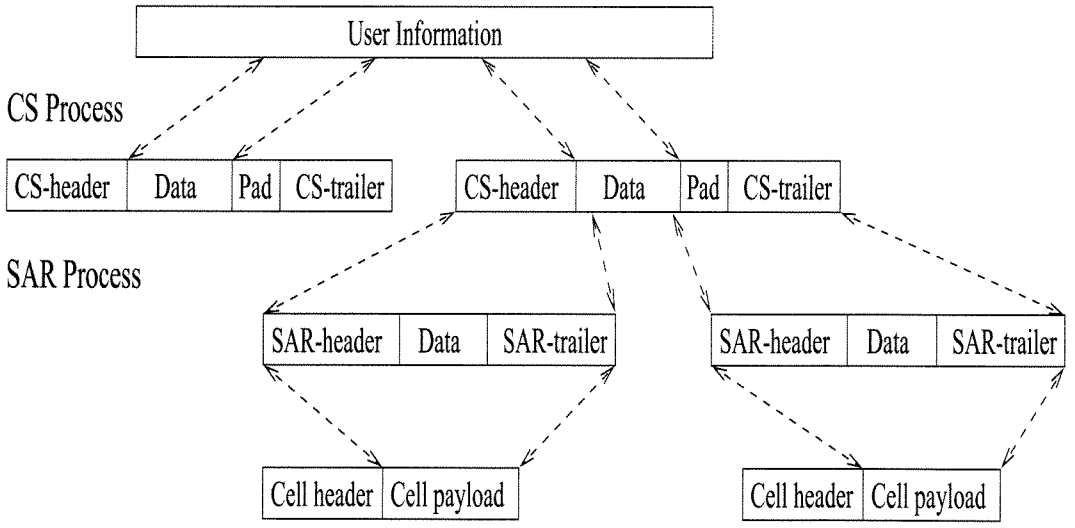



Figure 2.5: AAL Process

In the AAL process on the transmitter side, the higher layer data is first passed to the CS of the AAL. CS is meant to portion user data into fixed length PDUs and wrap it with a header and a trailer containing information about the type and

size of the CS-PDU as well as information to provide checks of PDU integrity at the receiving end. The CS sublayer is further subdivided into two parts: common part CS (CPCS), performing all CS functions common to all AALs, like multiplexing and cell loss detection and service specific CS[1] (SSCS), performing service-specific requirements of different applications, like timing recovery for real-time applications. SAR receives the CS-PDUs and segments them (or gathers them in constant bit rate services) so that when the SAR header and trailer are added, it becomes 48 bytes to fit into the payload field of a cell. On the receiving side, SAR layer reconstructs the CS-PDUs and passes them onto the CS sublayer (Fig. 2.5).

There are three sets of criteria used when defining different classes of traffic types to be serviced by AAL:

- Time relation versus no time relation between the source and the destination.
- Constant versus variable bit rate.
- Connection-oriented nature versus connectionless.

These criteria provide an opportunity to classify all possible services into eight classes out of which four are defined. (Table 2.4)

Corresponding to these four classes, standards define five different AAL protocols:

1. AAL Type 1; Constant Bit Rate (CBR) Services:

   This AAL is used for CBR services that require a timing relation between the two ends of the transmission. This allows the network to carry applications like high quality audio, video, and telephony and also emulate DSn services. In AAL 1, out of the 48 bytes of payload, four bits are used for sequence numbering (SN) and another 4 bits for SN protection (SNP), leaving a net payload of 47 bytes for the cell. Out of the four SN bits, one bit belongs to the CS and the other three are used to number the cells from zero to seven. This way cell loss is immediately detected unless it occurs in multiples of eight cells. The CS bit is used to let the receiver know that the cell contains clock information.

[1]This part may even be null if the service doesn't require any specific functions

| | |
|---|---|
| Class A | Constant bit rate (CBR)<br>Connection-oriented<br>With timing relation<br>Ex. 64-Kbps voice and constant bit-rate video |
| Class B | Variable bit rate (VBR)<br>Connection-oriented<br>With timing relation<br>Ex. Encoded video |
| Class C | Variable bit rate (VBR)<br>Connection-oriented<br>No timing relation<br>Ex. Connection-oriented data transfer |
| Class D | Variable bit rate (VBR)<br>Connectionless<br>No timing relation<br>Ex. Data transfer between two LANs over a WAN |

Table 2.4: B-ISDN Service Classes

2. AAL Type 2; Variable Bit Rate (VBR) Timing Sensitive Services:

   This AAL is not totally defined yet. It is reserved for data services requiring transfer of timing information between end-points as well as data. Compressed video with its bursty nature is likely to be the main application. This way there will be no need for the complex buffers and rate smoothing circuits now used to produce DS0 or DS1 interfaces. SAR structures for AAL 2 will most probably include SN, SNP, IT (information type: beginning, continuation, or end of a message), LI (length indicator: the number of useful bytes in padded cells), and CRC fields.

3. AAL Type 3; Connection-Oriented VBR Data Transfer:[2]

   This AAL is designed to transfer VBR services over pre-established connections. It is intended for large, long period data transfer, like a file transfer or backup. The SAR structure contains LI, CRC, IT, and SN fields. A further 10 bits are reserved for use by higher layer protocols, leaving 44 bytes as the net payload

[2]AAL 3 and 4 are now obsolete since they were first joined together under the name AAL 3/4, but even this was outclassed by AAL 5 later on.

in the cell.

4. AAL Type 4; Connectionless VBR Data Transfer:

   AAL 4 is intended for transmission of VBR data without pre-established connections. It is useful for short, highly bursty LAN traffic. It is especially efficient in short message situations when the call set-up time could be comparable to the message transfer time. It provides capabilities for both point-to-point and point-to-multipoint services.

5. AAL Type 5; Simple and Efficient Adaptation Layer:

   AAL 5 is the newest (but still well-defined) AAL providing improved efficiency over AAL 3/4. Compared to the other two, it has the same effective payload usage for CS-PDU sizes of 88 bytes or less and has smaller overhead for larger CS-PDUs. It also simplifies the SAR portion to pack all 48 bytes of the cell information field with data. To allow these improvements, it assumes that only one message crosses the UNI at a time, not allowing for interleaving of messages on the same VC, but instead queueing them for sequential transmission.

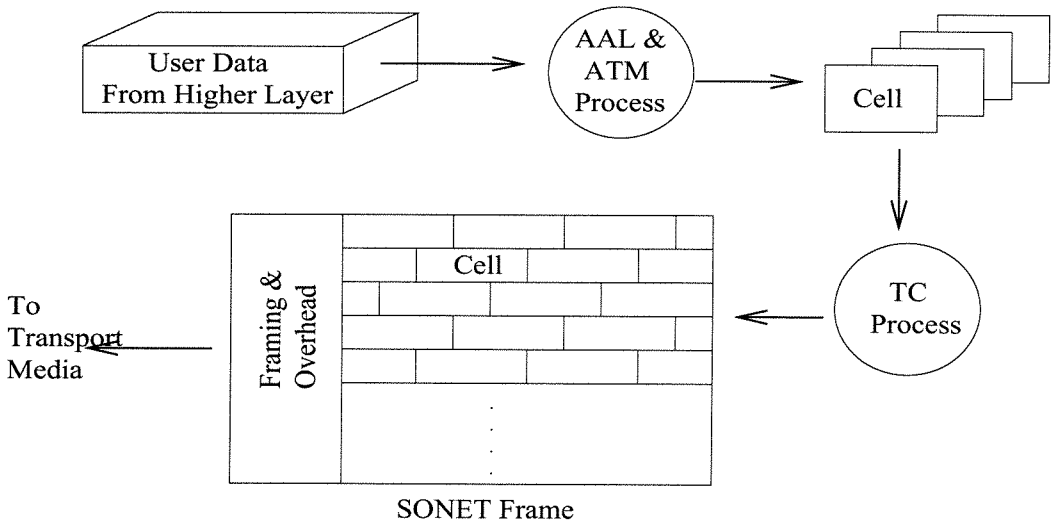



Figure 2.6: Overall Lower Level Processes

## 2.3 ATM Networking

ATM technology is connection-oriented, requiring end-to-end connections to be established before the beginning of transmission. Communications are accomplished by setting up a virtual channel between the sender and the receiver(s) and then transmitting the information. (This allows network management to gather end-to-end data on the network traffic and charge users only for the actual time they've accessed the network.) These connections may be permanent (PVC – permanent virtual circuit) or established on demand (SVC – switched virtual circuit).

An ATM connection is a concatenation of links that pass or process the signal. This routing is accomplished by the concept of virtual channels and virtual paths. A virtual channel is a connection between two communicating ATM transmitters. It might consist of more than one hop and all communications proceed along this same virtual channel, preserving the cell sequence and the QoS. Virtual paths are bundles of virtual channels between two points and may contain many ATM links (Fig. 2.7 [24]). While the virtual channels are associated with a virtual path, they are not unbundled or processed separately (virtual-path routing). This routing information, most of the time, has local meaning only because of the difficulty of end-to-end addressing in the short address field size at the header.

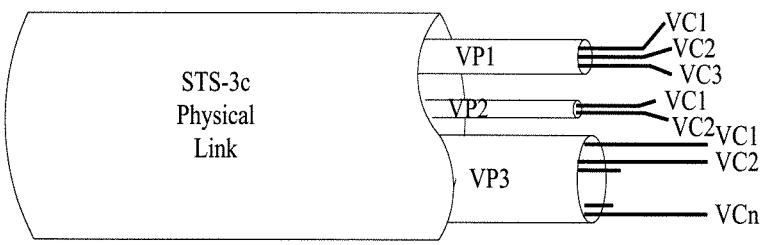
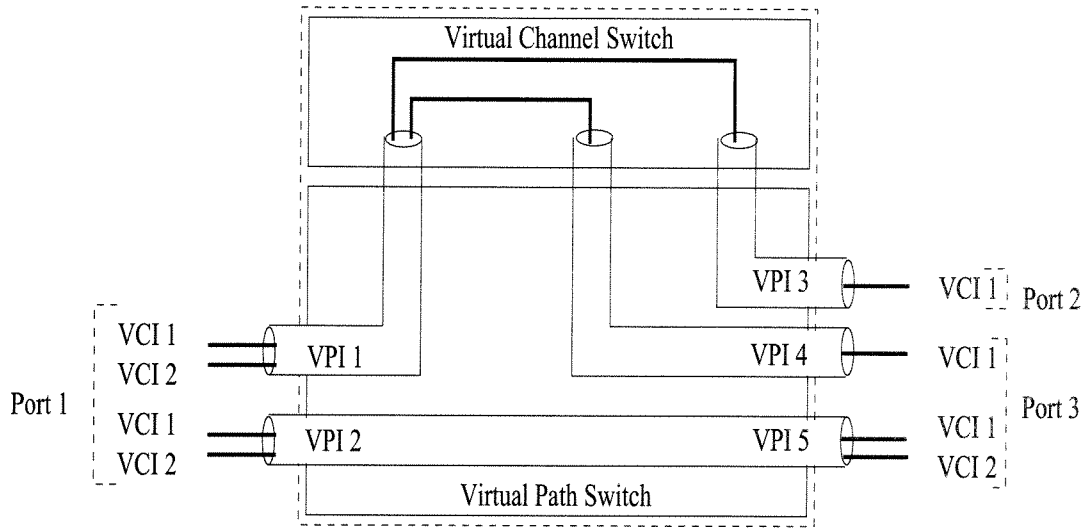



Figure 2.7: Virtual Paths and Channels

Virtual paths provide a convenient way of bundling the traffic going in the same direction and simplify switching by allowing the traffic to be switched according to a shorter field. Apart from the fact that same VCIs can be used unambiguously with different VPIs, assignment of VCIs and VPIs can be done on a per-node basis to increase the availability of these identifiers (Fig. 2.8 [24]). Most of the time, VCI's

are changed on each hop, but VPIs last at least a couple of hops.




| Input | | Output | |
|---|---|---|---|
| Port | VPI/VCI | Port | VPI/VCI |
| 1 | 11 | 2 | 31 |
| 1 | 12 | 3 | 41 |
| 1 | 21 | 3 | 51 |
| 1 | 22 | 3 | 52 |
| 2 | 31 | 1 | 11 |
| 3 | 41 | 1 | 12 |
| 3 | 51 | 1 | 21 |
| 3 | 52 | 1 | 22 |

Figure 2.8: An Example of Virtual Path and Channel Switching

At the switch, switching is performed by reading the routing field of the incoming cell, performing a look-up on a table that gets updated every time connections are set up or torn down to determine the outgoing port and the new routing identifiers, and delivering the cell with the new header to the corresponding output port.

## 2.4 Why ATM ?

The reason ATM generated so much enthusiasm is because it offers solid benefits. The first one is the bandwidth efficiency, which ATM achieves by allowing sources to seize bandwidth when a sufficient number of bits are generated. With no slots being

assigned owners, no bandwidth gets wasted simply because the owner is not active at the time.

ATM is also scalable; same 53-byte cell format is used over many different systems, rates, and formats. Vastly different systems like LANs, switches, and public networks can use the same format. This way, a cell generated by a 100 Mbps LAN (its outside connection, that is) can be carried over a 45 Mbps DS3 to the central office and switched into a 2.4 Gbps SONET transport system.

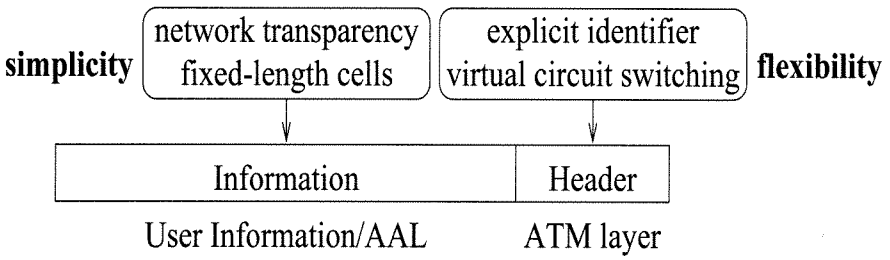



Figure 2.9: Simplicity and Flexibility

Moreover, ATM is application transparent. The cell size is a compromise to accommodate the whole range of application from short periodic needs of speech to long file transfers of data applications.[3] This is one of the chief benefits of ATM in allowing mixture of data, voice, and video within the same application without having to worry about compatibility between LANs and WANs.

ATM's granularity allows the applications to be not limited to the fixed restrictive granularity of today's TDM networks. On a TDM system, if an application requires more than DS0, but less than DS1, one has to either buy an entire DS0 or find some other applications to bundle them together on the DS1. Fractional T1 services are not available on the local access level and even on the network level, they are in increments of DS0. In other words, the user has to fit the network. In fact, this gets even worse between DS1 and DS3 levels. With ATM, though, the network can be tailored to fit the application by providing any speed with the same delay, continuity and synchronization guarantees.

[3]In fact, it is also a compromise between European and US standards. The information field length of 48 bytes is compatible with both 32-byte and 64-byte systems.

Another advantage of ATM is networking options. Apart from being the means for virtual private data networking – bringing to data traffic the same advantages carried-based virtual private networks brought to corporate voice traffic – the uniform cell format will greatly simplify interconnecting the LANs and the WANs. The protocol conversions will be avoided and only address manipulation will remain. Furthermore, time-dependent traffic will be easily dealt with without any outside intervention into the network.

ATM will pave the way for efficient high-speed networking necessary for most of today's and tomorrow's applications. Its very fast switching and routing based on the cell header, the characteristic nature of no processing above the cell level within the network – unlike X.25 – short, fixed cell size – unlike frame relay – simplify and speed up the handling of the messages and lead the way towards very high-speed self-routing switches.

# Chapter 3 Call-Routing Algorithms

The switching concept in ATM brought with it some very new details. Even though, neither circuit switching nor packet switching is new in idea or implementation, the combination of these two has a flavor in itself. Circuit switching in itself is a quite well-understood area. Along the same lines, packet switching has a lot of coverage in queueing theory applications. However, ATM makes the application of the latter more difficult by its introduction of fixed duration packets (cells) with non-poison interarrivals and complicates the former with the possibility of a link being shared by multiple connections. This chapter will focus on this second problem.

## 3.1 Circuit Switching

In the circuit switching case, the studies on space-division switches (Ex: Fig. 3.2) made up of crossbars (Ex: Fig. 3.1) provide a lot of insight into building and operating efficient switches (including most of the cases when time-division switches or even a combination of the two can be modelled as a space-division only).

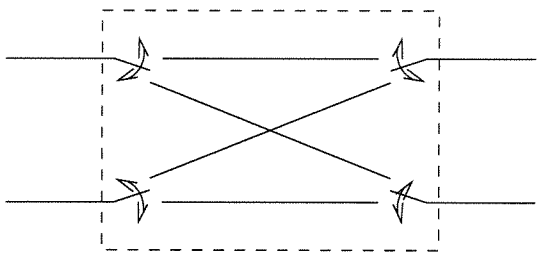


Figure 3.1: A Two by Two (2x2) Crossbar

From the beginning, it was obvious that crossbars with $n$ terminals and $n^2$ crosspoints are nonblocking. Later on, C. Clos [7] published a paper on a family of switches, named after him, where same nonblocking could be obtained with a network of crossbars with far fewer crosspoints. This was obtained by a necessary and sufficient

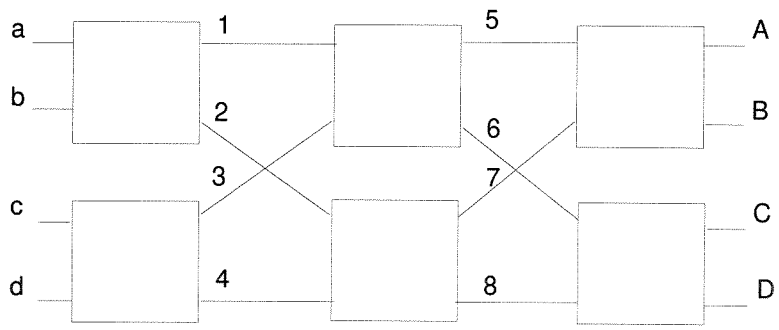



Figure 3.2: Three-Stage, 4x4, Space-Division Switch Made up of 2x2 Crossbars

condition imposed on the crossbars of the different stages. A three-stage $N$x$N$ minimal Clos switch contains $m$ $n$x$(2n-1)$ crossbars on the first-stage ($N = m\,n$), $2n-1$ $m$x$m$ crossbars on the center-stage, and $m$ $(2n-1)$x$n$ crossbars on the final-stage. The actual condition is on the input and output (considering that the switch is symmetric around the center) crossbars. If the first-state contains $n$x$r$ crossbars, then $r \geq 2n-1$ is required for nonblocking (strict-nonblocking, actually, which will soon be defined). In the later years, Benes [6] came up with his switch architecture made up of 2x2 crossbars that provided a very convenient and simple architecture that led the way to Banyan and similar other types; Cantor [8] provided efficient distribution and collection networks; and Lee [5] introduced a simple close-approximation method to calculate the blocking probability inside an given structured switch architecture. This, later on, was modified by Pippenger [9] to provide more accurate results under all conditions by relaxing the assumptions a bit. The idea was further modified to be used in multi-rate networks [12], [13], [14], but only after a huge leap in the level of complexity.

The key idea over here is that it is possible to have strictly nonblocking switches in the circuit switching case but happens to be harder to accomplish for ATM space-division switches. To understand this better, one should look into the different types of nonblocking:

In circuit switching (CS), a call request is a doublet $\{x, y\}$, where $x$ is an input port and $y$ is an output port. In the multirate case (ATM), this becomes a triplet $\{x, y, w\}$ with $w$, $b \leq w \leq B$, being the weight or the bandwidth of the connection

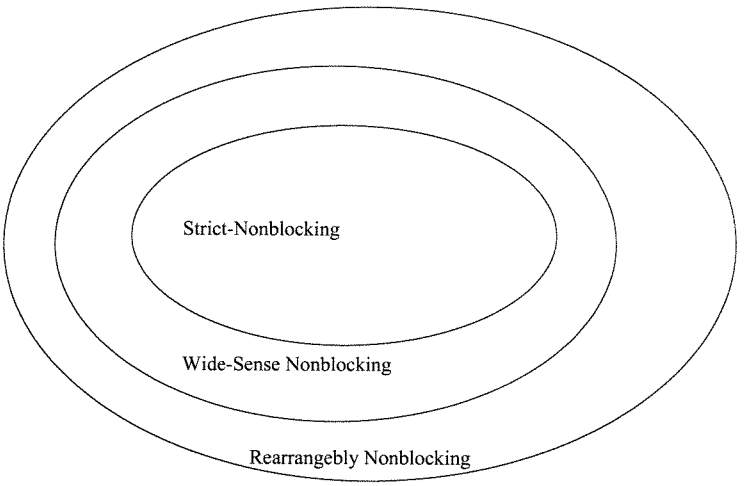



Figure 3.3: Different Types of Nonblocking

as a percentage of the link capacity and possibly bounded by two network-imposed limits. To define blocking, a couple of further definitions for the switch are needed: (along the lines of Pippenger's [9] terminology) For CS, a *route* is a set of contacts, one in each stage, that when closed simultaneously, establish a continuous path from an input to an output. A route is completely determined by specifying the nodes it involves, so one may speak of a route as if it were a set of nodes rather than a set of contacts. Two routes are *compatible* if they have no node in common. A *state* is a set of routes each two of which are compatible. In a given state, a route is *admissible* if it is compatible with every route in the state or in other words, the contacts it involves are open and all the nodes it involves are idle. Under these definitions, for a given network, a state, and a call request $\{x, y\}$, if there is an admissible route from the input to the output, it is said that the request is *linked*; if not, it is *blocked*. Under these definitions, a nonblocking network is termed *strict-sense nonblocking* if a route can be found between an idle transmitter and an idle receiver without disturbing the already set up calls no matter what the state of the network is and regardless of the past history of the network.

In the ATM case (converting these definitions in the Turner and Melen sense [10], [11]), a *route* will again be a path joining an input to an output, but together with a weight. A route satisfies a request $\{x, y, w\}$ if it connects $x$ to $y$ and has weight $w$.

A set of call requests is said to be *compatible* if for any input and/or output $x$, the sum of the weights of all connections involving $x$ is $\leq \beta$. A set of routes is *compatible* if for all links $l$, the sum of the weights of all routes involving $l$ is $\leq 1$. A *state* is a set of compatible routes for which for every internal link $l$, the sum of the weights of all routes including $l$ is at most 1. A link or a switch $y$ is said to be *w-accessible* in a given state from an input $x$, if there is a path from $x$ to $y$ such that the weight on every link in the path is at most $1 - w$. Under these definitions, a network is *strictly nonblocking* if for any state and any call request $\{x, y, w\}$ compatible with it, there exists a route, compatible with the state, to realize the request.

In other words, in both cases, strict-nonblocking is when, provided that the requested input and output ports are available, the call can be routed through the switch regardless of the state of the switch while the paths are being chosen randomly among the available ones. The notion of randomness is important here when placing the calls, though, because the next region that will be under inspection in this chapter uses certain algorithms to place the calls to leave more space for the future call requests. That way when choosing a path through the switch for each incoming call request, a certain algorithm is adhered to. In CS there are families of switches that are not strictly-nonblocking, but are wide-sense nonblocking under this definition (e.g. Moore switch). The final field of nonblocking is when the state of the switch can be altered to accommodate a new call request. A number of already existing calls need to be rerouted to make room for a new call. This is named rearrangebly nonblocking.

## 3.2 Call Blocking

As can be seen from the definitions in the previous section, the major difference between CS and ATM is the bandwidth or the weight, $w$, of the ATM connection. This is because in ATM, unlike CS, a user might be occupying a certain portion of the link enabling users to transmit at almost any speed up to the link capacity. Furthermore, only call rejection due to inability to route the call inside the switch is

taken into consideration when defining blocking, due to the CS tradition. The call request that is legally accepted and tried to be routed by the switch is the one that complies with bandwidth requirements at the input and at the output ports – the one that is compatible in both cases. This can be considered to be a very legitimate exclusion to a great extent for the CS case because under normal circumstances only one subscriber will be connected to an input port $x$; however, under ATM operation, many transmitters may be connected to the very same input port, $x$, and many might have access but not have a connection at that moment. Therefore, call requests that are rejected due to this incompatibility should also be given attention to.

Here are three types of blocking that will be considered: *Internal blocking* occurs when at a state $s$, for a call request $\{x, y, w\}$ that is compatible with input $x$ and output $y$, there does not exist a route inside the switch that can satisfy it. This is the kind of blocking that is mentioned in the previous section and discussed in all the sources quoted. Unless specifically noted otherwise, from this point on the term 'blocking' refers to this type. The other two types looked into, though, fall under a category that would be termed incompatible under the definitions of the previous section. *Input blocking* happens when for a request $\{x, y, w\}$ with the addition of $w$, the sum of the weights of all connections involving input $x$ exceeds $\beta$. Furthermore, *output blocking* is when for a request $\{x, y, w\}$, with the addition of $w$ the sum of the weights of all connections involving output $y$ exceeds $\beta$.

Consider an $nxn$ switch with input and output ports of capacity $\beta$. Each input port has $\leq N$ independent sources connected on it. A call request from source $i$ will be expressed as $\{x(i), y(i), w(i)\}$, with $x(i)$ being the input port source $i$ is connected to and $y(i)$ an output port among all $n$ equally likely output ports and $(0 \leq b \leq)$ $w(i)$ $(\leq B \leq \beta \leq 1)$ the fractional bandwidth of the connection.

**Definition 3.1** *Input blocking occurs when a new call request, $\{x, y(j), w(j)\}$, has a weight such that*

$$\sum_{\substack{i \\ x(i)=x}} w(i) > \beta \tag{3.1}$$

*where*

$$\sum_{\substack{i \\ x(i)=x \\ i \neq j}} w(i) \leq \beta \tag{3.2}$$

In this case, this new call, $\{x, y(j), w(j)\}$, is rejected because the input cannot accommodate that much more bandwidth at that time. Output blocking is the same phenomena on output port $y$:

**Definition 3.2** *Output blocking occurs when a new call request, $\{x(j), y, w(j)\}$, is such that*

$$\sum_{\substack{i \\ y(i)=y}} w(i) > \beta \tag{3.3}$$

*where*

$$\sum_{\substack{i \\ y(i)=y \\ i \neq j}} w(i) \leq \beta \tag{3.4}$$

*Ex:* Consider the 4x4 switch in Fig. 3.2 with $b = 0.1$, $B = 0.8$, $\beta = 0.95$ and the present state with the following list of calls in progress:

$\{a, C, 0.4\}$ on links 1 and 6 $\{b, C, 0.2\}$ on links 1 and 6

$\{b, A, 0.7\}$ on links 2 and 7 $\{d, D, 0.3\}$ on links 4 and 8

$\{d, B, 0.5\}$ on links 3 and 5

- Input blocking:

  Call request $\{a, D, 0.65\}$ cannot be accommodated since and input port $a$, $0.4 + 0.65 > \beta$ while

$$\sum_{\substack{i \\ x(i)=a}} w(i) = 0.4 \leq \beta = 0.95$$

- Output blocking:

  Call request $\{c, A, 0.3\}$ cannot be accommodated since at output port $A$, $0.7 + 0.3 > \beta$ while

$$\sum_{\substack{i \\ y(i)=A}} w(i) = 0.7 \leq \beta$$

- Internal (switch) blocking:

  Call request $\{a, B, 0.45\}$ cannot be accommodated because the path over links 1 & 5 is closed because $0.45 + 0.6 > 1$ and the path over links 2 & 7 is closed because $0.45 + 0.7 > 1$ .

## 3.3 Routing Algorithms

The necessity of efficient routing algorithms for ATM switching originates from the very fact that ATM is a multirate environment. On top of the two-dimensional switching problem is another dimension added with the introduction of weight. This results in the possibility one port blocking up a percentage of multiple internal links inside the switch because that single port can support many calls – unlike the CS case. At this point, the limiting factor, $\beta$, at the input and output ports becomes important. This speed limit is given as a fraction with respect to the speed of the links inside the switch itself. This means that the internal links are faster than the input and output ports by a factor of $1/\beta$, called the speed-up factor. This, it turns out, is an absolute necessity for strict-nonblocking in ATM.

**Theorem 3.1 (Turner-Melen)** *A three-stage Clos switch made up of dxm first-stage crossbars, $(n/d)x(n/d)$ center-stage crossbars, and mxd third-stage crossbars is strictly-nonblocking if*

$$m > 2 \max_{b \le w \le B} \left\lfloor \frac{\beta d - w}{max\{1 - w, b\}} \right\rfloor \tag{3.5}$$

In fact, another similar but simpler theorem provides more insight into the $\beta$ factor being considered:

**Theorem 3.2** *A three-stage Clos switch made up of dxm first-stage crossbars, $(n/d)x(n/d)$ center-stage crossbars, and mxd third-stage crossbars is strictly-nonblocking if*

$$\beta \le \frac{\lceil m/2 \rceil}{\lceil m/2 \rceil + d - 1} \tag{3.6}$$

*for $b = 0$, $B = \beta$.*

*Proof:*

To an arbitrary state, a new connection $\{x, y, w\}$ needs to be added. On the input crossbar of $x$, there total sum of inputs can be $(d-1)\beta + (\beta - w) = d\beta - w$. Since the network is symmetric, making less than half the center stage crossbars accessible from $x$ and $y$ will create blocking since this will ensure that there is no center stage crossbar accessible by both. To prevent this, the weight that can be shared on these links should be able to accommodate $w$:

$$1 - \frac{\beta d - w}{\lceil m/2 \rceil} \geq w$$

or

$$1 \geq \frac{\beta d}{\lceil m/2 \rceil} + w\left(1 - \frac{1}{\lceil m/2 \rceil}\right)$$

Since $0 \leq w \leq \beta$ and this equation needs to be satisfied for any value of $w$, the value of $w$ that gives the largest right hand side is $w = \beta$.

$$1 \geq \frac{\beta d}{\lceil m/2 \rceil} + \beta\left(1 - \frac{1}{\lceil m/2 \rceil}\right)$$

This can be rearranged to give

$$\beta \leq \frac{\lceil m/2 \rceil}{\lceil m/2 \rceil + d - 1}$$

This means that for a 16x16 three-stage switch made up of 4x4 crossbars on all stages, the biggest possible $\beta$ is 0.4, meaning that the internal links have to be at least 2.5 times faster than the ports. If a minimal 16x16 Clos switch with 4x4 center-stage crossbars is considered under this light, it can be seen that the maximum value for $\beta$ is at most 4/7. An increase that doesn't live up to the almost doubling of the crosspoint count. In other words, on top of the necessary distribution stage in the CS switches, there is now the speed-up factor added. The internal links have to be operating at faster speeds than the external ports. This is certainly possible at around OC-3 speeds, but as one goes higher up the hierarchy, the task gets harder to

realize.

The reason why things change in this fashion is due to the possibility of having multiple connections on a single port with ATM. In CS, since only one connection could exist on any one port and that single connection had to go through one certain path through the switch, only one path was used up per active port. On the other hand, in ATM many connections can exist one a single port and these connections might be routed through different paths or through the same path. Since it is possible to have these connections going through different paths inside the switch, one port can block a fraction of multiple internal paths, resulting in future connections to have limited bandwidth resources left through the switch.

It is intuitive that a more orderly way of positioning the call requests on the switch will provide more free space than a random positioning as was envisioned in the idea of strict-nonblocking. There comes the idea of using call-routing (or call-packing) rules. This creates the larger set of wide-sense nonblocking networks. The goal in this section, though, is not to create nonblocking, since it seems to be quite difficult for ATM, but to reduce blocking probability for a given value of $\beta$ so that lower speed-up factors can be utilized.

There are four main call-routing algorithms that will be investigated:

**Balanced Routing Algorithm:** For a call-request, $\{x, y, w\}$, among all the existing routes between input $x$ and output $y$, the one that has the most available bandwidth on is chosen to route the call provided that the available bandwidth is greater than $w$. If no such path exists, the call is rejected.

**Maximum Utilization Algorithm:** For a call-request, $\{x, y, w\}$, among all the existing routes between input $x$ and output $y$, the one that has the least available bandwidth on is chosen to route the call provided that the available bandwidth is greater than $w$. If this link does not have enough free bandwidth, then the algorithm is repeated without taking this link into consideration. This procedure is repeated until either a path with sufficient available bandwidth is found or no more possible paths are left, in which case the call is rejected.

**Fixed-Priority Algorithm:** All the center-stage crossbars are prioritized at the beginning. For a call-request, $\{x, y, w\}$, among all the existing routes between input $x$ and output $y$, the one that goes through the highest priority center-stage crossbar is chosen to route the call provided that the available bandwidth on it is greater than $w$. If this link does not have enough free bandwidth, then the algorithm is repeated without taking this link into consideration. This procedure is repeated until either a path with sufficient available bandwidth is found or no more possible paths are left, in which case the call is rejected.

**Maximum Switch-Utilization Algorithm:** For a call-request, $\{x, y, w\}$, among all the existing routes between input $x$ and output $y$, the one that goes through the center-stage crossbar that has the highest aggregate traffic on is chosen to route the call provided that the available bandwidth on it is greater than $w$. If this link does not have enough free bandwidth, then the algorithm is repeated without taking this crossbar into consideration. This procedure is repeated until either a path with sufficient available bandwidth is found or no more possible paths are left, in which case the call is rejected.

*Ex:* Going back to the switch in the same state as in the example on page 24, a call request $\{a, B, 0.2\}$ will be routed through the path

- 1 & 5 under fixed priority algorithm;
- 2 & 6 under maximum utilization algorithm;
- 1 & 5 under balanced loading algorithm;
- 1 & 5 under maximum switch-utilization algorithm;

whereas a quite similar call request $\{a, B, 0.3\}$ will have to use the path 1 & 5 under all algorithms since the path 2 & 6 can't support it.

Together with these four a few other routing algorithms will be considered occasionally as well as random routing where a path is chosen at random and checked to see if it has enough available bandwidth.

## 3.4 Comparison Of The Algorithms

These call-routing algorithms were compared over extended simulations on various switches, but predominantly on an 8x8 three-stage space division switch with 2x4s on the first stage, 4x4s on the center stage, and 4x2s on the final stage (Fig. 3.4). For the most part, ten sources are connected to each input port even though there are instances when this number will be varied intentionally. The output ports are simply sinks. A silent source comes alive at any time instance, $n$, with probability $\alpha$ and an active source ends transmission at any time instance with probability $\delta$, giving the active and silent periods average holding times of $(1-\delta)/\delta^2$ and $(1-\alpha)/\alpha^2$ respectively. When a source becomes active, it assumes a weight, $w$, randomly from a uniform distribution between $b$ ($\geq 0$) and $B$ ($\leq \beta \leq 1$). An output port is also randomly chosen from among all the output ports, each being equally likely.

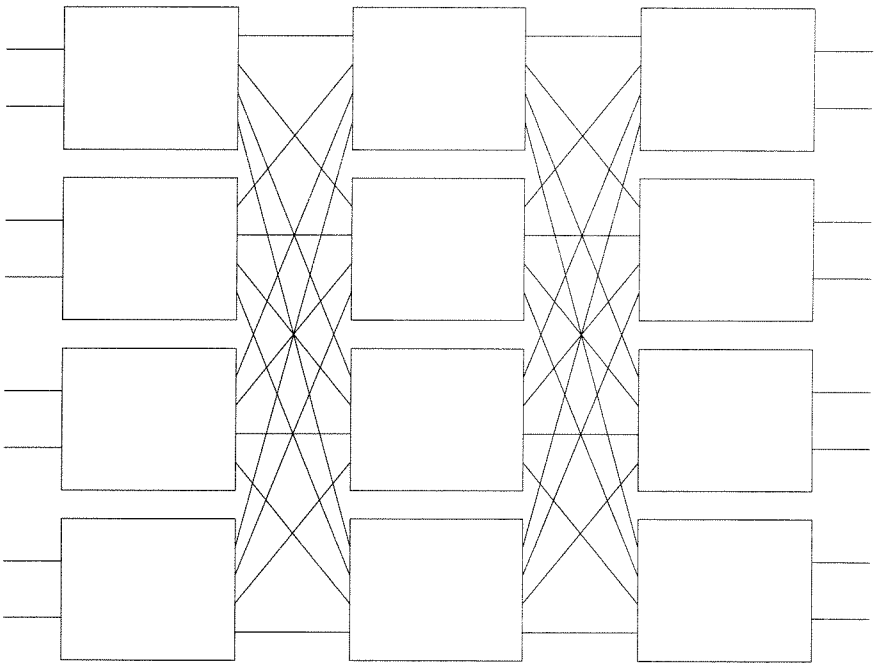


Figure 3.4: Sample Switch

When such a call request, $\{x, y, w\}$ arrives at the input port $x$, the system first checks to see if with the addition of $w$, the total weight at port $x$ exceeds $\beta$ and rejects the call if it does while recording the attempt for input blocking. If there is enough space at the input, then the output port $y$ is checked for the same purpose and if the

total weight exceeds $\beta$ after the addition of $w$, the call is rejected and the attempt is taken down as output blocking. If the output port has enough space to accommodate this new call, then the system attempts to find a path, able to accommodate weight, $w$, through the switch, using the call-routing algorithm under investigation. If such a path exists, the link is established and $w$ amount of resource is written off as reserved at the input port, output port, and on the internal links for the duration of the transmission and if no such path is found, then the call is rejected, recording the attempt as switch blocking.

### 3.4.1 Simple is Good ?

Later on, the complexity of these algorithms will briefly be mentioned, but it is not very difficult to see that among these algorithms, fixed-priority algorithm is the simplest one. The first and main result is about this algorithm – and the other ones, too.

**Conjecture 3.1** *Among the four routing algorithms considered, for a given set of parameters,* $b, B, \beta$ *where* $0 \leq b < B \leq \beta \leq 1$, *fixed-priority algorithm provides the least blocking.*

The most striking result of this chapter is that, the fixed priority algorithm, with all its simplicity provides the least blocking. The experimental data that supports this conjecture can be seen in the following three figures (Figs. 3.5, 3.6, 3.7). As can be seen, the least blocking is experienced when operating with fixed-priority. The second best is maximum switch blocking. The worst is balanced routing and in fact it is on occasion even worse than random routing. This pattern holds true on all three graphs (and on all other data obtained).

Couple of other interesting concepts can be observed on this set of data that surface on every other simulation:

- For a fixed $b$, the blocking probability increases with increasing $B = \beta$.

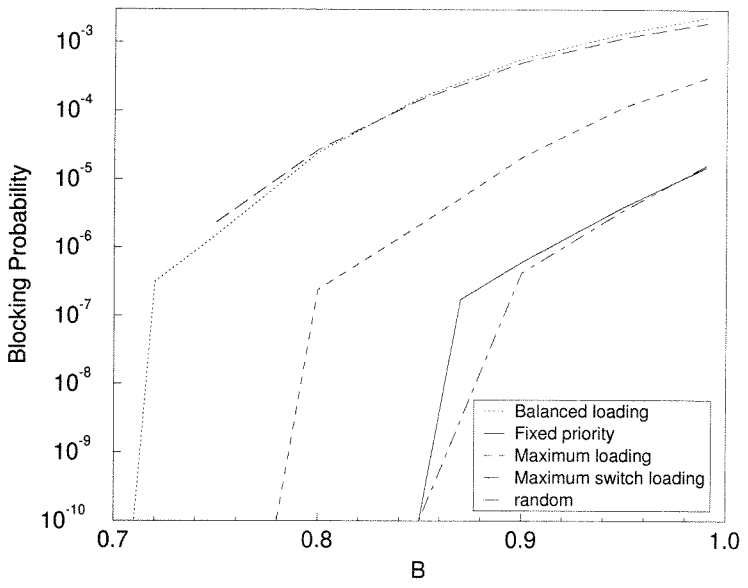



Figure 3.5: Blocking Probability Varying With Respect to $B$. $b = 0.10$, $\beta = B$

- For a fixed $B = \beta$, the blocking probability first increases and then decreases with increasing $b$.
- Algorithms based on center-stage crossbars do better than algorithms based on individual links.

The reasons why all these are happening are occasionally not so apparent but understandable when they are figured out. First of all, the reason why balanced routing does such a lousy job is simple: It litters the links with low bandwidth calls and due to this, cannot find available bandwidth for high bandwidth calls in the future. As a simple example, a situation when four calls of bandwidths $b + \epsilon$ arrive at the same input can be considered. Now these will most probably be routed through four different links. In the future all calls arriving before any of these calls get disconnected that have a bandwidth greated than $1 - b - \epsilon$ will have to be rejected because of these earlier low-bandwidth calls. In fact, the reason why balanced routing is slightly worse than random routing is that in random routing, these four calls have better probability of not being on four different links, leaving behind bigger chunks

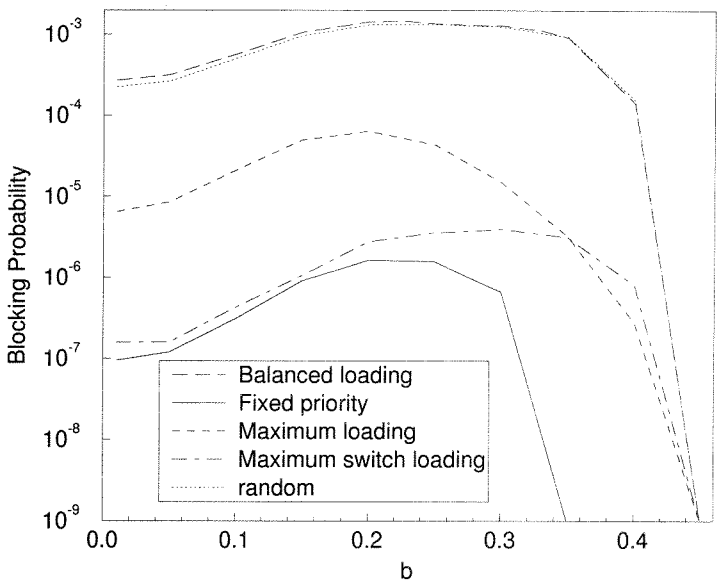



Figure 3.6: Blocking Probability Varying With Respect to $b$. $B = 0.90$, $\beta = 0.90$

of available bandwidth.

The real questions are why fixed priority is the best and why maximum loading is worse than the two crossbar-based algorithms. It is extremely hard to answer these questions, especially the first one. Some insight into these questions will be provided, though. The latter question is a bit easier to handle. Maximum loading algorithm takes links one at a time. In other words, it makes a decision based on one link between either the first and the center stages or the center and the third stages. However, this decision affects both links on either side of the center stage. It is quite likely that this decision is made based on a link that was heavily loaded because of other calls that are connected to links on the other stage other than the link that the call in question will be on. (To clarify it a bit, consider the top center-stage crossbar and the third link out of it connecting to the third stage. This link might be heavily loaded due to the first and fourth links coming from the first stage, but the call request in question might be coming from the second link.) Then it is likely that one link might be getting loaded because of a decision based on another link. On the other hand, with the crossbar based algorithms, the decision is being made

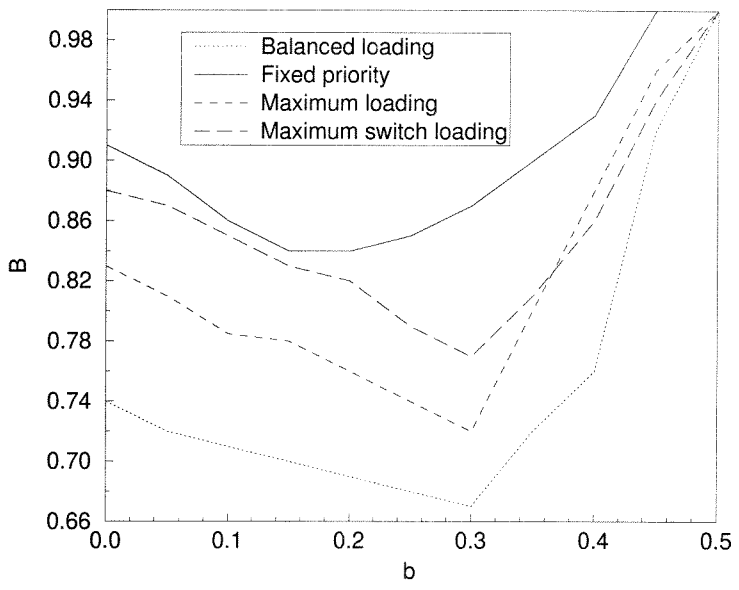



Figure 3.7: Zero-Blocking Lines
The area below each line contains that $\{b, B\}$ pairs that do not create any blocking.

based on neither of the links, but the crossbar. This provides a longer global decision rule rather than rules based on short-lived transients on links. The links on both sides of the center-stage crossbar is likely to be loaded if the crossbar is loaded in the maximum switch loading algorithm and certain crossbars are always loaded together with their links in the fixed priority.

This leaves some space for explaining why fixed priority does marginally better than maximum switch routing. Even though this question remains to be investigated further, one reasonable assumption is that the latter rule creates more frequent transients when one crossbar gains supremacy over the others while the former is a stable situation on all times.

Among the three minor observations listed above, the first two deserve some explanation, too. For a fixed $b$, as $B$ increases, the possibility of a small-bandwidth call occupying a link and blocking a high-bandwidth call increases. (It becomes more probable that the second call will have a bandwidth $1 - b - \epsilon$ after a former one of $b+\epsilon$.) This explains the increase in blocking as $B$ increases. The other observation is a

bit more complicated since the blocking probability first increases and then decreases with increasing $b$. The high end is easier to consider. For a fixed $B$, there comes a point where the value of $b$ is such that there can only be a single call on any port. This is exactly like the circuit switching case, which does have strictly nonblocking switches. Of course, the situation is helped with the existence of routing algorithms. As an example consider $\beta = 1.0$ and $b = 0.5 + \epsilon$. There can exist only a single call on one port. This not only reduces the blocking probability, but also provides nonblocking after a certain value. Of course, this effect of increasing difficulty of having many calls on one port with increasing $b$ is trying to pull the blocking probability all the time. However, it is losing to some other factor for low values of $b$. That other factor is the increase in the average bandwidth rejected when a call is blocked as $b$ increases. This also affects the previous situation explained at the top of the paragraph. At one point, though, the effect of the possible number of calls on one port gets more prominent and the blocking probability begins to decrease. (This makes the number of simultaneous possible connections on a port important. It will most probably be large on NNI nodes, but might be small at UNIs.)

### 3.4.2 Light Traffic

The results of the previous section are obtained under fairly heavy traffic conditions where $\alpha = 0.9$ and $\delta = 0.55$ for each source and with ten sources per port. If the same situations are simulated under lighter traffic the same results hold but with greatly reduced blocking probabilities (Fig. 3.8).

In fact, not only that the blocking probabilities of the algorithms balanced routing and maximum loading are reduced, but those of maximum switch loading and fixed priority disappear altogether. This effect is also observed when the number of sources connected to a single port is reduced. The latter is due to the effect of both the reduced traffic intensity and the reduced number of possible simultaneous connections. Even though it appears as if the performance difference between different algorithms reduces as the offered traffic is decreased, it should be noted that balanced routing

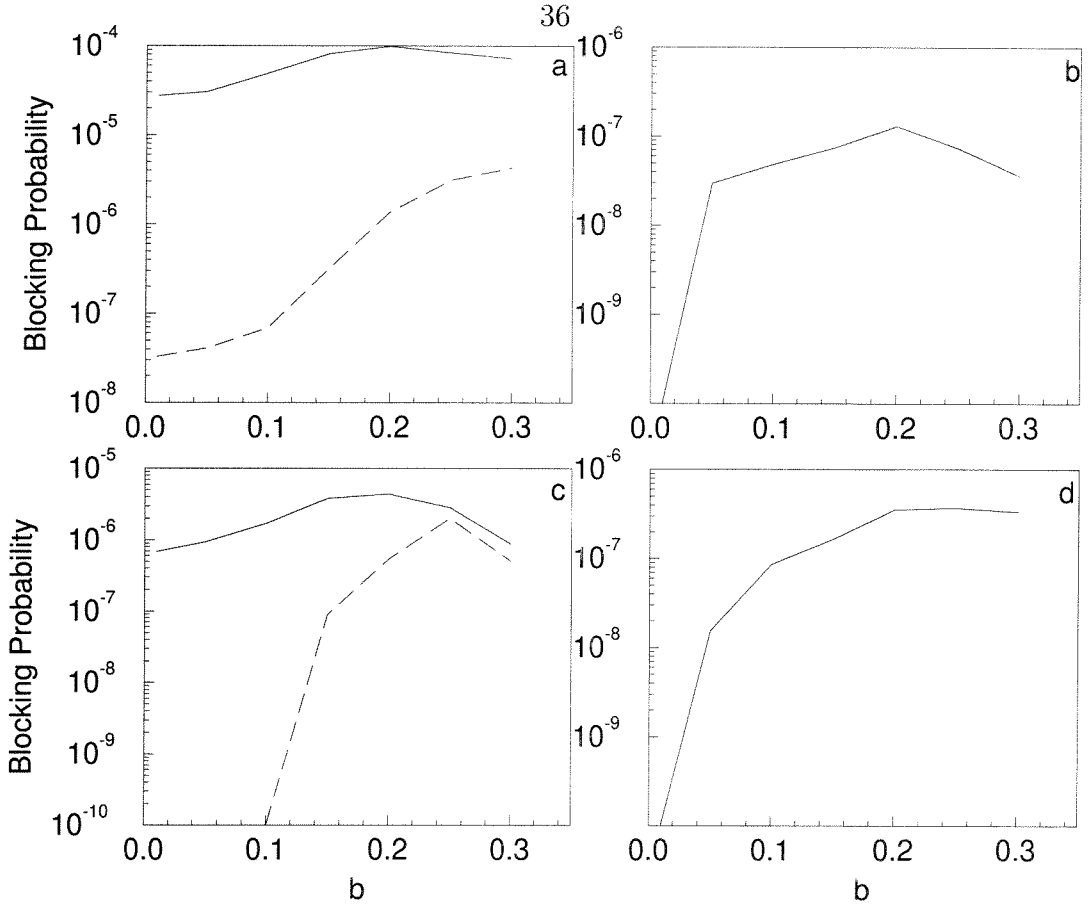



Figure 3.8: Effects of Traffic Intensity
For the solid line $\alpha = 0.85$, $\delta = 0.55$. For the dashed line $\alpha = 0.70$, $\delta = 0.55$. a) Balanced loading, b) Fixed priority, c) Maximum loading, d) Maximum switch loading.

still performs very poorly compared to others. In the situation when only two sources per port are investigated, not only does balanced routing still provide blocking, but it provides blocking on the order of $10^{-5}$ whereas none of the others go anywhere above $10^{-8}$.

### 3.4.3 Other Types of Blocking

This is the point where other types of blocking come in and stain the picture. When taken into consideration, due to the high number of persistent sources, input and output blocking happen to dominate over internal blocking. Same data displayed above also shows that internal blocking is as low as about one per cent of all the bandwidth

blocked and that when looked at under this light all four routing algorithms have absolutely comparable blocking probabilities. This seems to make the algorithms appear useless. However, one must consider that at user-network interface switches, the number of users per port will most probably not be very high and the users will not be extremely insistent. Even for light traffic, the algorithms provide better results. Inside the network itself, at the node-network interfaces, though, the number of users will be high and the traffic will be heavy most of the time because it is multiplexed traffics of many users. However, at these junctions, the switches and the switch port will be operating at higher hierarchy speeds and each single connection will have a share as a fraction of $\beta$. This will prevent input and output blocking to a great extent and justify the usage of an efficient routing algorithm; namely, fixed-priority.

### 3.4.4 Other Requirements

One other area of comparison is the set-up delay created by these algorithms and also their memory requirements. Inspecting each one-by-one, balanced routing, for each arriving call on an $dxm$ first stage crossbar, needs to make $m$ comparisons first between the two links of each path and $m-1$ more to find the one with the lightest weight on. This is the same for maximum loading. Fixed priority does not need to do any of these while maximum switch loading needs to do the $m-1$ comparisons of the sort. Each algorithm has to do two additions and two comparisons to see if the links of the path found can support the new connection, but this is common to all so need not be considered. Clearly fixed priority is the fastest. One way to speed up balanced routing, maximum loading, and maximum switch loading is to keep the necessary aggregate sums in a sorted lookup table. This brings up the topic of memory requirements. (It also puts some non-realtime work on the switch controller, but since it is not realtime, it is favorable against a sort that has to be done while the call request is waiting.)

Each algorithm has to keep track of the available bandwidth on each link at all times to start with. Other than that, if one decides to keep the necessary sorted items

in the memory for faster call set-up, then all three algorithms other than fixed priority require $m$ memory segments that need to be updated with adjusted and resorted data after every connection or hang-up while fixed priority doesn't need any.

## 3.5 On-Line Bin Packing

In an attempt to explain the behaviour of the routing algorithms, the classical computer science bin packing problem can be taken as an analogy. In the general bin packing problem, there are one-dimensional bins and incoming one-dimensional bricks that need to be placed inside the bins. The number of bins are unlimited and a new bin is created every time a brick cannot be placed in the already existing bins. The incoming bricks have their sizes as random variables with a specified distribution (mostly specific to the problem being considered). Since in the general problem, both the number of bricks and the number of bins are indefinite, as a comparison, the tail-end distributions of the number of bins used are considered.

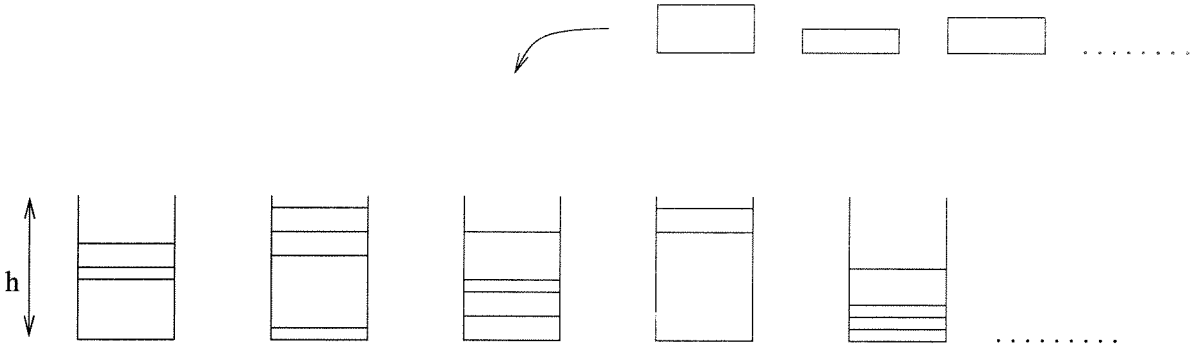



Figure 3.9: On-line Bin Packing

The on-line bin packing problem has a number of widely-known algorithms for brick placement. The two that will be considered are quite similar to two that were used for call-routing in the previous section. These two algorithms are known as First Fit (FF) and Best Fit (BF) algorithms. The names are quite descriptive of the way they work. In the FF algorithm, the brick is first attempted to be placed in the first bin. If it doesn't fit, then the second bin is tried and then the third, etc. If none of the existing bins have enough space to accommodate this brick, a new bin is 'activated'

and the brick is placed in this one. In BF, the existing aggregate brick levels inside the bins are considered. The bin that contains the most gets the first pick for the new brick. If the brick doesn't fit there, then the bin with the next highest brick level is tried. This procedure is repeated until either the brick gets placed into one of the already active bins, or a new bin has to be activated for it and it gets placed there. If they don't already sound similar to fixed priority and maximum loading, they are!

### 3.5.1 Analogy

The way the ATM switch fits the bin packing model is with the internal links viewed as bins and the call requests are bricks. The weight of the call is the random variable that needs to be stacked into these bins. There are a number of differences between this restricted problem and the general bin packing problem. These variations are:

- The number of bins are finite and fixed.
- Bricks have a lifetime; they appear and after some time disappear.
- There are two layers of bins.

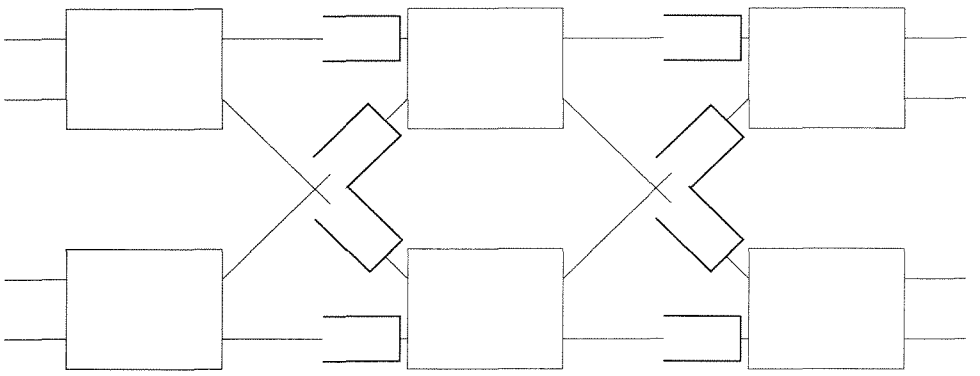


Figure 3.10: ATM Switch Links As Bins

The classical bin packing problem as described above in the general sense has exactly the opposite results in terms of algorithms. The expected waste under FF grows faster than that of BF. It is shown over and over again (tightening the bounds) that BF performs better than FF [1], [2], [3] even in bounded space [4]. These results

seem to contradict the behaviour of call routing algorithms, but thanks to the three major differences, they can be totally different problems with different outcomes. One needs to look carefully to find which one(s) of the differences create this major change.

### 3.5.2 Effects of Differences

It is essential to figure out among the three differences listed previously which, if any, of the restrictions create a change. To accomplish this, first the situation is considered with a finite number of bins. This is a case that is similar to what is referred to as $K$-bounded bin packing where there can only be $K$ active bins at a time and if a brick cannot be placed in any of these, one bin gets deactivated and a new empty one is activated (like a truck loading area with $K$ parking spaces. A truck has to leave before a new truck can begin loading). In the ATM switch, the number of bins is permanently fixed. When such a situation is looked into with a fixed number of bins and a predetermined number of bricks with sizes of random variables uniformly distributed between 0 and 1, it is observed (Figs. 3.11, 3.12, 3.13) that BF creates less blocking than FF (of course, there actually is no blocking in the original bin packing problem because there are infinite number of bins, but when the number of bins is finite, then the bricks that cannot be placed in any bin are discarded and this is again the criteria used for comparison) in total agreement with the general bin packing results.

This can also be observed in a very simple example with two bins of unit capacity and and four bricks with random heights.

**Theorem 3.3** *Given two bins of unit size and four incoming ordered bricks with $x_i$ is the height of the ith brick where $x_i$ is a uniformly distributed random variable between 0 and 1, the FF has a higher blocking probability in terms of number of bricks and also brick height.*

*Proof:* Until the third brick, the two algorithms will obviously perform the same: The first two bricks will either go into the same bin ($x_1 + x_2 \leq 1$) or into two separate bins ($x_1 + x_2 > 1$). If the former, then there will be no difference between algorithm

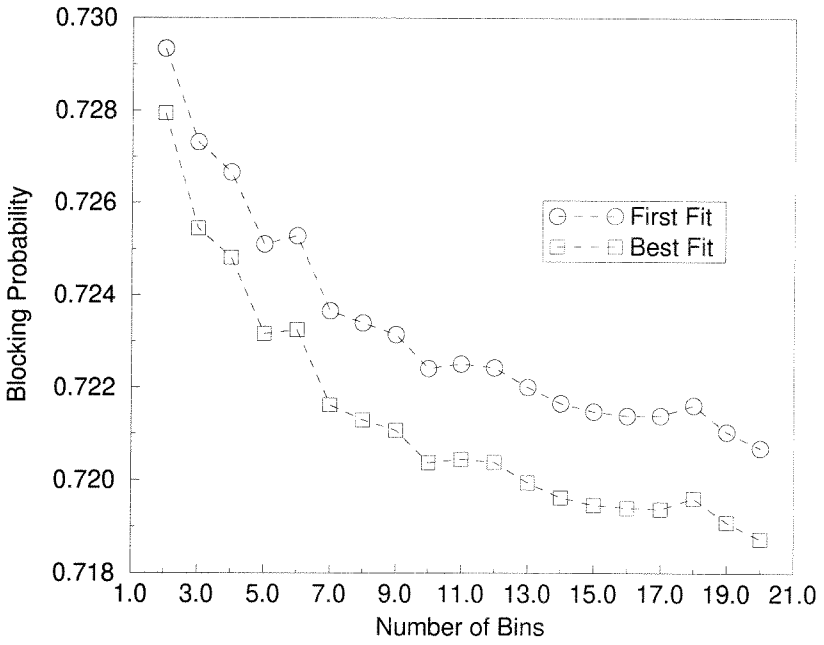



Figure 3.11: Finite Number of Bins and Bricks. (The number of bricks is seven times the number of bins.)

performances all the way through brick four because the third brick will certainly be accepted and the the fourth will either find a place in both algorithms or be rejected by both. On the other hand, the latter case can create a fork in algorithm performance if $x_1 \leq x_2$. In that situation BF will try to put the third brick in the second bin first and will place it into the first only if it doesn't fit in the second. FF, though, will try to put it in the first bin first. In the situation $x_3 + x_2 \leq 1$, BF will place the third brick in the second bin whereas FF will place it in the first bin. However, since $1 - x_1 > 1 - x_2$ placing the third brick in the second bin will leave much more space for the fourth brick. In other words, FF has a higher probability of rejecting the fourth brick than BF. Since in all other possible scenarios, both algorithms reject or accept a brick together, FF will have a higher blocking rate than BF.

*Ex:* Let $x_1 = 0.5$, $x_2 = 0.6$, $x_3 = 0.3$, and $x_4 = 0.45$. In both algorithms, $x_1$ will go into the first bin and $x_2$ into the second bin. Then FF will place $x_3$ into the first bin and BF will place it inside the second bin. Because of this placement, FF will

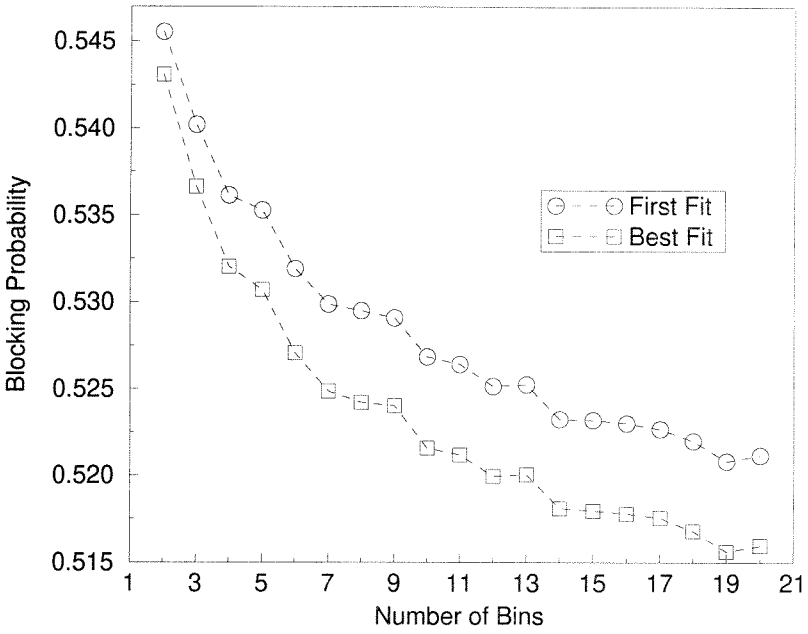



Figure 3.12: Finite Number of Bins and Bricks. (The number of bricks is four times the number of bins.)

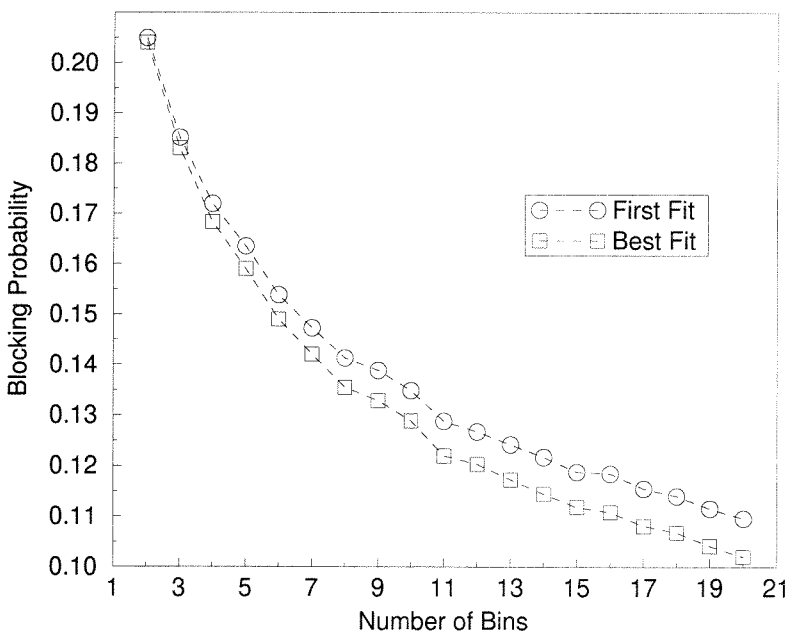



Figure 3.13: Finite Number of Bins and Bricks. (The number of bricks is twice the number of bins.)

have to reject $x_4$, but BF will be able to place it in the first bin.

Since the first difference didn't help much, the finite brick lifetimes concept is the next one to be looked into. This way the bricks not only appear randomly, but also disappear, making space for newly arriving bricks. In this model, only a single state change per time instance is allowed. A departure takes place with probability $\delta$, an arrival takes place with probability $\alpha$ and with probability $1-\alpha-\delta$ there is no state change. This model also agrees with the general bin packing in terms of BF still being better than FF (Fig. 3.14).

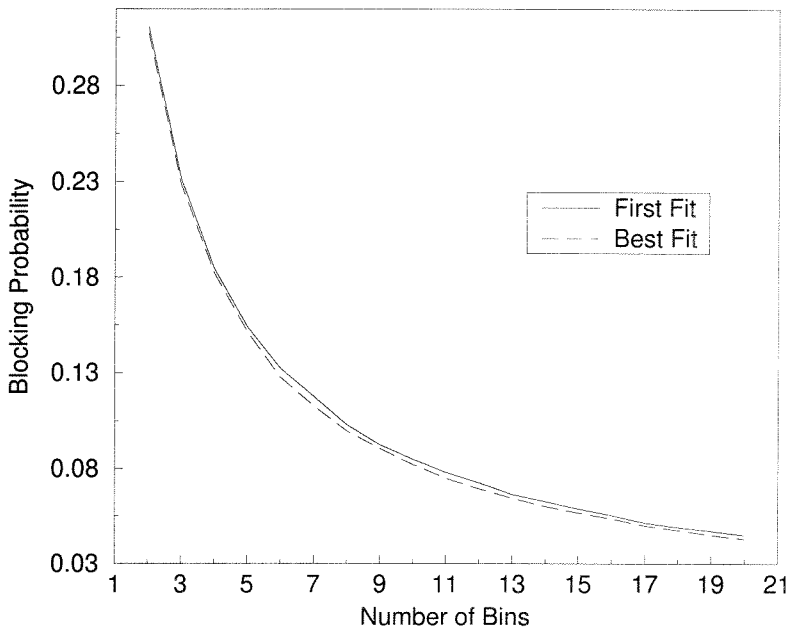



Figure 3.14: Finite Number of Bins; bricks have a lifetime. $\alpha = \delta = 0.42$

This leaves only one more avenue of approach. The layering of bins. Before looking into it in detail, there is one simple observation. Even though the occupancy of the first bin in FF is not of any special interest, it can easily be determined. For this purpose, the following problem can be considered: An ordered list of $n$ bricks of various sizes with $x_i$, $0 \leq x_i \leq 1$, being the height of the $i$th brick is introduced, starting from the lowest indexed brick, to be placed into a bin of unit size. The cumulative height of bricks accumulated inside the bin after $i$ bricks is $T_i$ and a new

$i+1$th brick is admitted if and only if $T_i + x_{i+1} \leq 1$ and it is discarded (i.e. blocked) otherwise. Under this model, what is the expected value of the height of the bricks accumulated inside the bin after $n$ bricks?

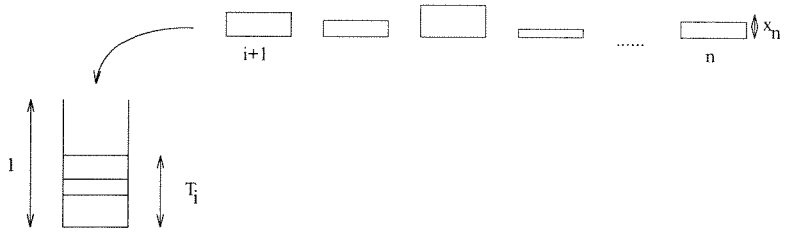



Figure 3.15: Single Bin

$$E\{T_n\} = ?$$

To answer this the contents of the bin can be defined recursively:

$$T_i = \begin{cases} T_{i-1} + x_i & \text{if } T_{i-1} + x_i \leq 1 \\ T_{i-1} & \text{otherwise} \end{cases} \tag{3.7}$$

$$T_1 = x_1 \tag{3.8}$$

where all $x_i$s are independent, identically distributed random variables between 0 and 1 according to some given pdf, $g(\cdot)$.

Let $F_i(t)$ be the cdf of $T_i$, i.e.

$$F_i(t) = Pr\{T_i \leq t\} \tag{3.9}$$

For example, $F_1(t) = t$ where $0 \leq t \leq 1$.

For the case in which $x_i$s are uniformly distributed between 0 and 1, the cdf will be given by:

**Theorem 3.4** *For the single bin situation, given that $x_i$ is a random variable uniformly distributed between 0 and 1, the cdf of the occupancy function after $i$ bricks is*

$$F_i(t) = t^i, \quad 0 \leq t \leq 1, i \geq 1 \tag{3.10}$$

*Proof:* Suppose $T_{i-1} = u$ for some $u \leq t$. Then

$$\begin{aligned} Pr\{T_i \leq t\} &= Pr\{x_i \leq t-u\} + Pr\{x_i > 1-u\} \\ &= G(t-u) + 1 - G(1-u) \end{aligned}$$

which gives for the uniform distribution:

$$\begin{aligned} Pr\{T_i \leq t\} &= \int_0^t ((t-u)+u)dF_{i-1}(u) \\ &= t\int_0^t dF_{i-1}(u) \\ &= tF_{i-1}(t) = tt^{i-1} \\ &= t^i \end{aligned}$$

In the more general sense, the cdf would take the shape:

$$F_i(t) = \int_0^t (G(t-u) + 1 - G(1-u))dF_{i-1}(u) \tag{3.11}$$

Given the cdf, the mean height of bricks inside the bin can be calculated and the answer to the initial question is:

$$E\{T_n\} = \frac{n}{n+1} \tag{3.12}$$

To explain the effects of the double layer of bins, a simpler model can be made use of. In this model, there are multiple sources and some sources have access to only some of the bins. The simplest example of this situation is with two sources and two bins (Fig. 3.16) and this example will be used to show the effects.

The two sources submit bricks in a probabilistic manner in such a way that when a brick arrives to the system, with probability $p$ it comes from source (b) and with probability $1-p$ it comes from source (a) ($0 \leq p < 1$). However, while a brick arriving from source (a) can be placed into either one of the bins – according to the algorithm

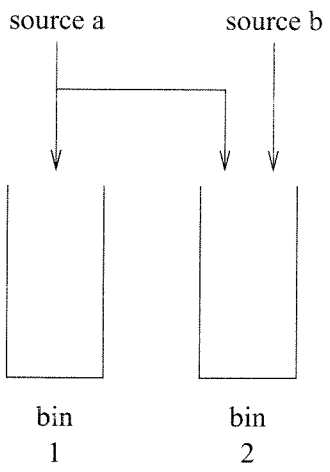



Figure 3.16: Two Bins and Two Sources. The Lopsided Model.

being used – a brick from source (b) can only be placed in the second bin. The bricks are being accepted or rejected as described before. Under these circumstances, first a simple situation is better to consider to get some insight:

In the case with two bins and two sources, one with access to both bins and the other to only one, consider a brick arriving from each source. The probability that the third brick will be blocked is higher in FF if $p$ is small and higher in BF if it is large. Here are the details of this simple situation which will be generalized: A brick with size $\epsilon_b$, $0 < \epsilon_b < 1$, arrives from source (b) and a brick of size $\epsilon_a$, $0 < \epsilon_a < 1 - \epsilon_b$, arrives from source (a). Under FF, $\epsilon_b$ will go into the second bin and $\epsilon_a$ will be placed in the first whereas in BF both will be placed in the second bin. At this point, what is the probability that the third brick will be blocked ? In BF, if the third brick arrives from source (a), it can't be blocked because the first bin is empty and if it arrives from source (b), it will be blocked if it is bigger than $1 - (\epsilon_a + \epsilon_b)$. Therefore the overall probability that the third brick will be blocked under BF is $(\epsilon_a + \epsilon_b)p$. On the other hand, for FF, if the third brick is from source (b) it will be blocked if it is larger than $\epsilon_b$ and if it is from source (a) it will be blocked if it is larger than the bigger one of $1 - \epsilon_b$ and $1 - \epsilon_a$. Therefore, the overall blocking probability for the third brick under FF is $p\epsilon_b + (1-p)\min\{\epsilon_a, \epsilon_b\}$.

$$p\,(\epsilon_a + \epsilon_b) \quad <\stackrel{?}{=}> \quad p\,\epsilon_b + (1-p)\,\min\{\epsilon_a, \epsilon_b\}$$

$$p\,\epsilon_a \quad \stackrel{?}{<=>} \quad (1-p)\,\min\{\epsilon_a, \epsilon_b\}$$

There are two equally-likely cases to consider:

- $\epsilon_a \leq \epsilon_b$. In this case, the decision simplifies to

$$p\,\epsilon_a \quad \stackrel{?}{<=>} \quad (1-p)\,\epsilon_a$$

  One can easily see that BF has higher probability to block the third brick when $p > 1/2$ and FF when $p < 1/2$.

- $\epsilon_a > \epsilon_b$. This time the relationship is:

$$p\,\epsilon_a \quad \stackrel{?}{<=>} \quad (1-p)\,\epsilon_b$$

  In this case, it is a bit more complicated. BF has higher probability to block the third brick when $p > \epsilon_b/(\epsilon_a+\epsilon_b)$ and FF when $p$ is smaller. The mean value of $\epsilon_b/(\epsilon_a+\epsilon_b)$ comes out to be 0.153426.

In this example, all this shows that when $p$ is chosen to be small, BF is better and when $p$ is large, FF is better (Fig. 3.17). Needless to say, among all three-brick combinations possible, this is the only case when the algorithms behave differently.

The situation above can be generalized to cover any number of bricks:

**Theorem 3.5** *For the lopsided two-bin model, for p sufficiently small, BF performs better and for p sufficiently large, FF performs better on the average in terms of blocking probability.*

*Proof:*

Consider the general case after $n$ bricks where the level inside the first bin is $x$ and the level inside the second bin is $y$. The $n+1$st brick can be from either one of the sources. If it arrives from source (b), then both algorithms will react the same and try to place it in the second bin and block it if this is not possible. If the brick

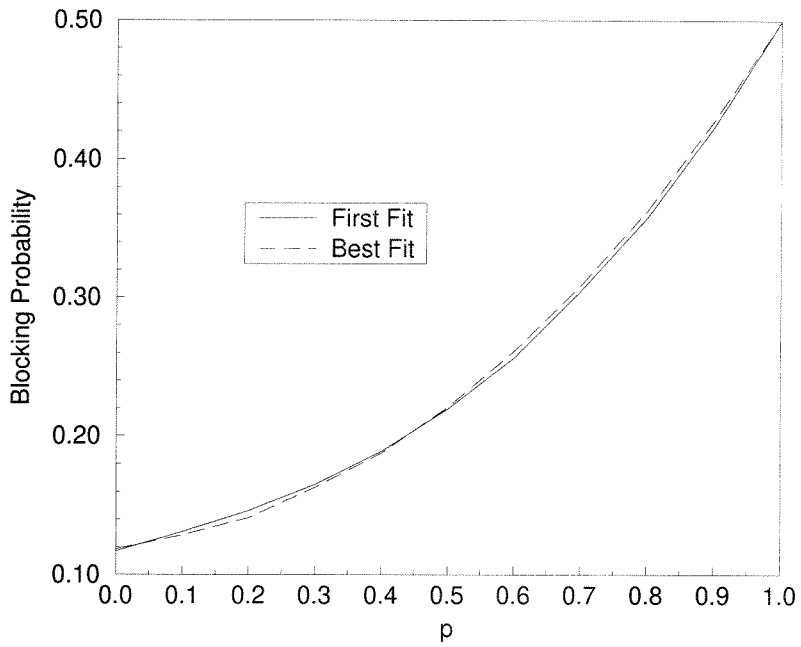



Figure 3.17: Simulation Results With Two Bins and Three Bricks in The Lopsided Model

arrives from source (a), though, things might be different. If it is blocked, it'll be blocked by both algorithms. If it is possible to admit it, though, it might be placed differently. Even in this case, if $x \geq y$ to begin with, again both algorithms will do the same thing. However, in the situation when $x < y$, BF will try to place it in the second bin first and FF will try to place it in the first bin first. If both of them are successful at their attempt, the problem reduces to the previous example for the $n+2$nd brick, with height $z$:

$$p\,(y+z) + (1-p)\,x \quad <\stackrel{?}{=}> \quad p\,y + (1-p)\,min\{x+z, y\}$$

Again the same two situations with first $x + z \leq y$ and the breakpoint $p = 0.5$ and the second $x + z > y$ with a complicated breakpoint of $p = (y-x)/(z+y-x)$ which is another random variable.

This shows that for this two-bin case, the result always holds true on the average because in the only case where the algorithms react differently, this effect holds

(Figs. 3.18, 3.19). Of course, it is true that when one algorithm rejects one brick, it might mean space for the next brick that might not find space in the other algorithm, but since the result holds true for any number of bricks, it will dominate on the average. These results with five and ten bricks can be seen in the following two graphs respectively.

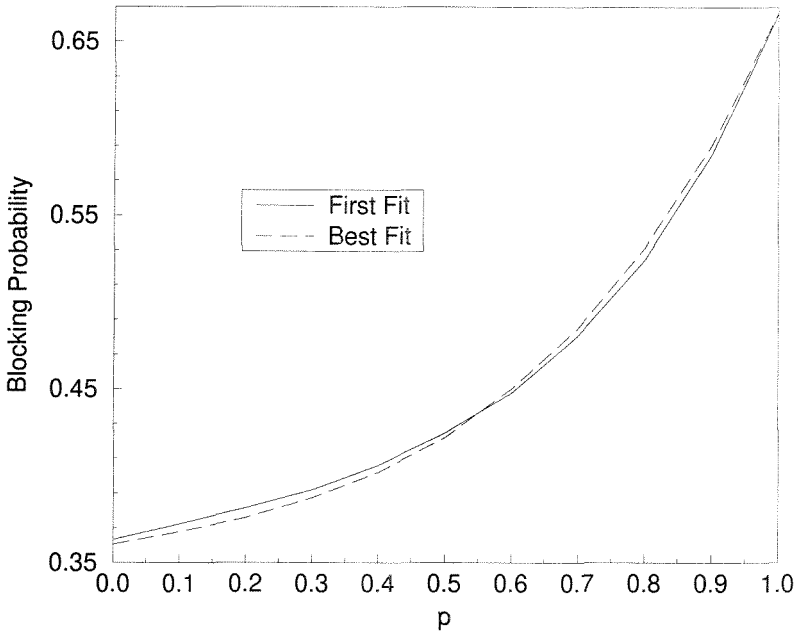



Figure 3.18: Simulation Results With Two Bins and Five Bricks in The Lopsided Model

## 3.6 Conclusion

The lopsided model with two bins actually represent the second stage of the three-stage ATM switch pretty nicely and explain what is happening. The reason why BF can't outperform FF is due to the intervention of the second source introduced. As $p$ grows larger, more bricks are being introduced by this other source and these 'alien' bricks are 'confusing' BF. On the other hand, FF has no problems with this because that algorithm is totally independent of the current status of the bins themselves.

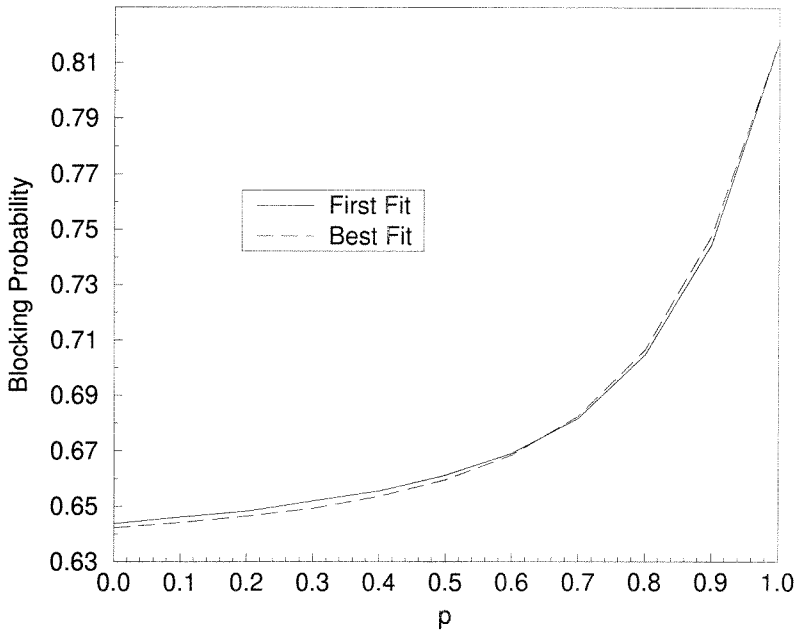



Figure 3.19: Simulation Results With Two Bins and Ten Bricks in The Lopsided Model

In the ATM switch, this is exactly what is happening between the center and third stages. The decisions made by the maximum loading algorithm is dependent on the traffic on two links on either side of the center-stage. The second layer of links, keep getting 'extra' calls to carry from the first stage based on decisions made on the first layer of links. This ruins the strategy of maximum loading algorithm and algorithms that don't depend on the traffic on the links win.

This chapter shows that even though it is much simpler than many others, fixed priority algorithm can be a very good candidate for routing decisions inside ATM switches. Using this algorithm, the blocking probability on the call level can be greatly reduced and also the speed-up factor can be kept at reasonable levels.

# Chapter 4 Buffer Management

ATM networks allow for the input traffic from the user to vary both from one call to another and also within one call. Since CBR service provides a fixed-bandwidth transmission circuit, it is fairly easy to deal with in terms of its instantaneous traffic load on the network. However, the situation is totally different for VBR services intended for bursty traffic. These applications, such as transaction processing applications and LAN interconnections, can send data at high burst rates as long as the overall transmission rates don't exceed a specified average. On top of this, the new classes being considered and defined, UBR (unspecified bit rate) with data being sent across the network with no guarantee when or if the data will arrive at its destination and ABR (available bit rate) with similarities to UBR but containing minimum bandwidth guarantees and flow control, complicate the network traffic further.

| Class of service | Bandwidth guarantee | Delay variation guarantee | Throughput guarantee | Congestion feedback |
|---|---|---|---|---|
| CBR | YES | YES | YES | NO |
| VBR | YES | YES | YES | NO |
| UBR | NO | NO | NO | NO |
| ABR | YES | NO | YES | YES |

Table 4.1: Service Class Differences

The primary role of traffic control and congestion control is to protect the network and the user in order to achieve network performance objectives. An additional role is to optimize the use of network resources. The traffic control and congestion control mechanisms should not rely on other higher layer protocols which are either application or service specific. However, protocols may make use of the information in the ATM layer to increase their efficiency.

There are two levels of congestion and control involved with ATM: the call level and the cell level. With ATM connections there is a unidirectional specifying of the

Quality of Service (QoS) parameters. These parameters are specified at connection setup and are guaranteed by the network. To guarantee the QoS, the network must be able to obtain enough information from the user about the connection and be able to ensure that no other connections that share the resources degrade the QoS. A user must enter into a contract with the network about the parameters of the call. Then the network will implement traffic control to avoid problems with degraded QoS before they occur. This includes Network Resource Management (NRM), Call Admission Control (CAC), Usage Parameter Control (UPC), and selective cell discard. The network will also control the case where congestion does occur by implementing Explicit Forward Cell Indication (EFCI), selective cell discard and reaction to UPC failure.

Within the ATM cell there are a number of bits available for congestion and priority setting. These include the Payload Type Indicator (PTI) and the Cell Loss Priority bit (CLP) both of which are contained in the header of the ATM cell. The first bit of the PTI tells that the cell is a user cell and the next bit tells if congestion has been experienced by the cell in the network. The last bit differentiates between two different types of ATM-SDU's. The CLP bit is for high and low priority setting of the cells. This can be done by the user and/or by the network. A cell entering the network with low priority is subject to being discarded by the network in times of congestion.

Traffic control is necessary to protect the network so that it can achieve the required performance objectives. UPC enforces a contract between the user and the network about the nature of the call. This prevents any one user from causing excessive traffic and hence degrading the quality of service provided to the other users. It is necessary to determine what is the worst traffic a user can inflict on the network while still abiding by UPC. The Leaky Bucket Algorithm is commonly used to implement UPC.

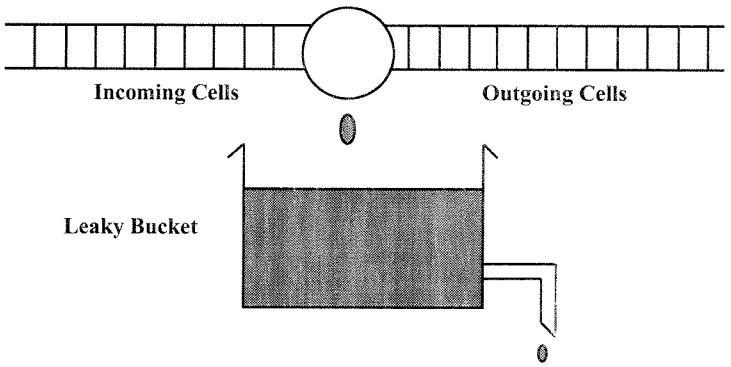



Figure 4.1: Leaky-Bucket Algorithm

## 4.1 Leaky-Bucket Algorithm

On an ATM network, at the beginning of a transmission, a 'contract' needs to be accomplished between the source and the network. The traffic contract specifies the negotiated characteristics of a connection. A connection traffic descriptor is the set of traffic parameters in the source traffic descriptor, the cell delay variation tolerance, and the conformance definition. The conformance definition is used to decide which cells are conforming in the connection. A typical conformance definition is the leaky-bucket (Fig. 4.1) [28] or Generic Cell Rate Algorithm (GCRA) although many such algorithms may be used in tandem. The CAC will use the connection traffic descriptor to allocate resources and to derive parameters for the UPC. Any connection traffic descriptor must be enforceable by the UPC. Even though a cell is found to be nonconforming, that does not mean that the connection is not conforming. The precise definition of a compliant connection is left to the network operator. However, a connection where all the cells are conforming is compliant. The traffic contract consists of the connection traffic descriptor and a requested QoS for each direction of the connection. This includes the definition of a compliant connection. The private UNI may support a different traffic to the public UNI.

The contract must contain the Peak Cell Rate (PCR) of the source traffic, the cell delay variation, and the cell delay variation tolerance. Sustainable cell rate and burst tolerance are optional parameters. For best-effort[1] traffic the only parameter specified

[1]The network will make its "best effort" to deliver the traffic, but without any guarantees.

is the PCR and the network may not reject the call because that bandwidth is not available but it may impose a different PCR. CAC is used to decide if a connection should be accepted or continue to be accepted in the network. It is required that the traffic contract be accessible to the CAC. The prime concern is to achieve the required QoS for the new or renegotiated connection as well as to ensure that the connection will maintain the QoS of all the other connections in the network. As well as deciding to accept the connection, the CAC must determine the parameters needed by the UPC and route and allocate the resources to the connection. Even if high and low priority are not set the network may set them for nonconforming cells. UPC is the set of actions the network take to monitor and control traffic. This includes the validity of the connection. The operation of the UPC shall not violate the QoS objectives of a compliant connection. However, the excessive policing actions on a compliant connection are part of the overall network performance degradation and so safety margins should be engineered to limit the effect of the UPC. The UPC can also fail to take action on a noncompliant connection. Policing actions on the nonconforming cells are not to be allocated to the network performance degradation of the UPC. At the cell level the UPC may pass a cell, change the priority of the cell or discard the cell. A low priority cell is discarded by the UPC if it is nonconforming. Following the UPC shaping may be implemented on the conforming cells to reduce cell clumping. It is optional for the network operator to allow the UPC to initiate the release of a noncompliant connection. When two levels of priority are used the UPC may discard high priority cells even though if the UPC were performed on the high priority alone the cells would be conforming.

The UPC and CAC are operator specific and should take into account the traffic contract to operate efficiently. It is specified that the signaling should take into account experimental traffic parameters that could be proprietary to either the manufacturer or network operator. It is optional to allow the the operation of these parameters across the UNI by mutual agreement. It is optional for the user to be allowed to mark cells as low and high priority. It is also optional for the network to mark cells as low priority if they are not adhering to the traffic contract. The cell

loss ratio for low priority cells must be higher than for high priority.

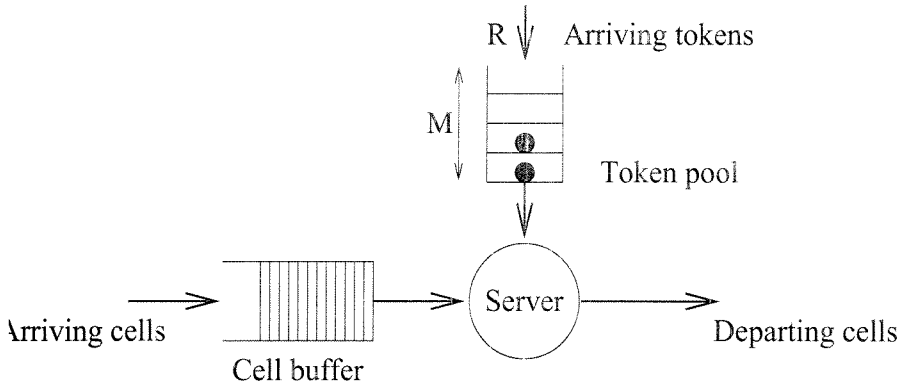



Figure 4.2: Leaky-Bucket Algorithm

The leaky-bucket algorithm (Figs. 4.2, 4.3) is a UPC standardized by the ATM Forum [29]. The operation of the leaky-bucket algorithm under the 'bucket' concept is that a splash is added to the bucket (counter increment) for each incoming cell when the bucket is not full. When the bucket is full, cells cannot pass through to the network unmarked but the bucket leaks away at a constant rate. A more formal definition should be made taking into consideration the rates and a "token pool" idea. The important parameters to be defined in this system are the token regeneration rate (leak rate of the bucket), $R$, the size of the token pool (bucket capacity), $M$, and the peak cell emission, $p$. Every time a cell arrives at the server, it needs to obtain a token to pass through into the network untagged. If there is at least one token at the token buffer at that time ($n \geq 1$), the cell passes through and the number of tokens at the buffer, $n$, get decremented by one. If there are no full tokens, then the cell either gets tagged as a nonconforming cell and is released into the network or is removed. It is also possible to have a cell buffer to buffer such cells until a token arrives at the token buffer. For the analysis in the rest of this chapter, it is assumed that if the cells cannot pass the leaky-bucket unmarked then they are lost and are not taken into account in the cell loss rate. (This is because the network will only give guarantees to the unmarked cells and the marked cells should not interfere with the unmarked ones.) The token get replenished at a constant rate $R$, but the buffer size can never exceed $M$ ($n \leq M$).

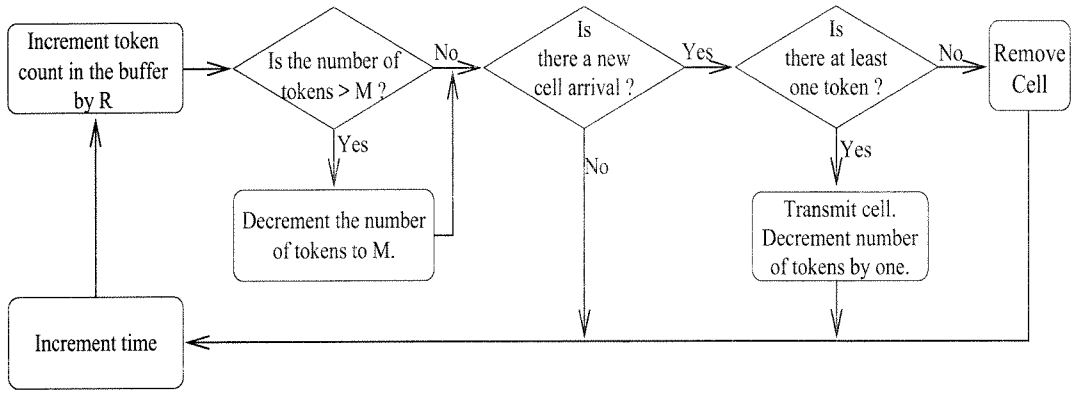



Figure 4.3: Discrete Time Leaky-Bucket Algorithm; Flowchart

The UPC standard of the ATM Forum is actually a double leaky-bucket that controls the two different cell rates $R$ and $p$, each with a separate bucket. The peak rate, $p$, is controlled by the first leaky-bucket with a token buffer of 1 tokens and a token replenishment rate, $p$, and the second bucket operates as explained above. For a cell to be a conforming cell, it needs to be conforming with both buckets at the time it is transmitted.

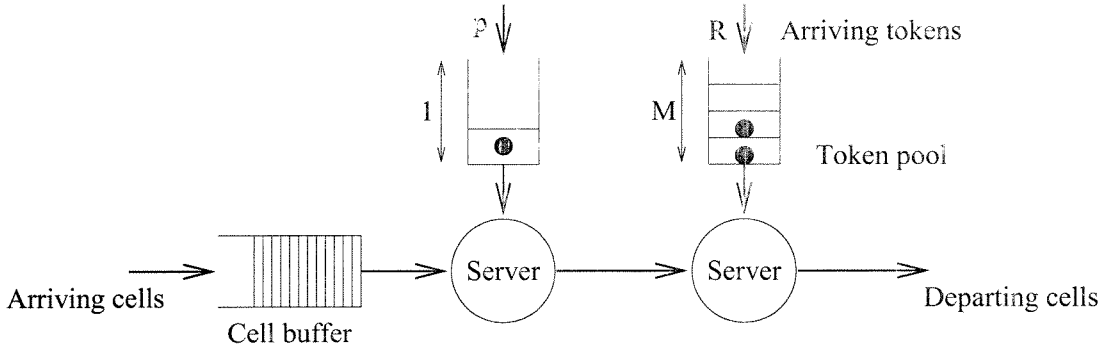



Figure 4.4: Double Leaky-Bucket Algorithm

The main utilization of the leaky-bucket algorithm as a traffic shaper is to 'smooth out' the traffic and reduce its burstiness. This allows the network to multiplex the traffic much more easily by reducing the burst collisions, the duration of the burst collisions and to reduce the cell loss rate (CLR). The effects of this will be observed in later sections.

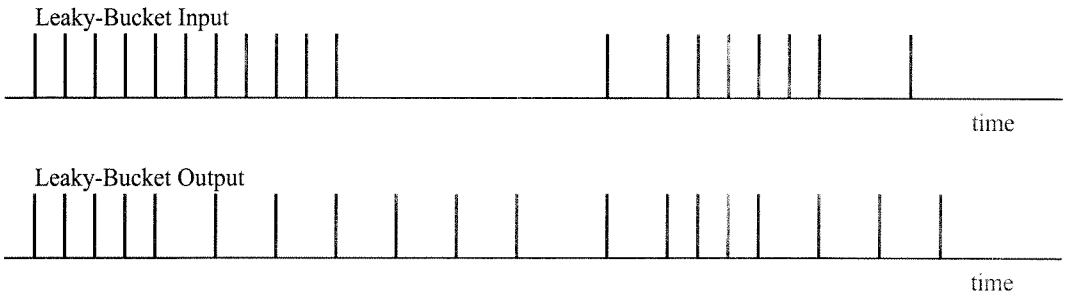



Figure 4.5: Input and Output Cell-Streams For a Leaky-Bucket with $M = 2$ and $R = 0.5$

## 4.2 Traffic Types

A simplified ATM switch (node) model consists of $N$ users feeding a finite FIFO buffer of capacity $B$ cells (Fig. 4.6).

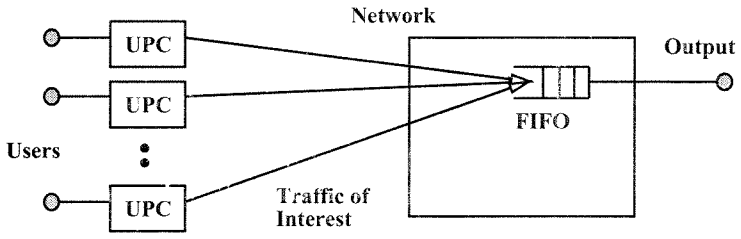



Figure 4.6: Simplified Model

The arrival process from the users is in general random, but for the worst case analysis it is assumed that the UPCs are heavily loaded and that the UPC algorithm makes the arrival process to the FIFO buffer deterministic. There are two types of arrival patterns, $x[n]$, that will initially be considered after the UPC and both of them are periodic. Due to the packetized nature of ATM and the synchronous operation of SONET, this traffic is described in a discrete fashion rather than a continuous fashion for the whole analysis, with the cell arrivals taking place at 53-byte intervals. The service process is also deterministic (at rate $r$). The cells that arrive at the FIFO are either served at the service rate, $r$, or are queued if another cell is being served at the moment. A cell is lost when a cell arrives, but cannot be served immediately and cannot be queued either due to the buffer being full. The worst case traffic is defined to be the one which creates the highest cell loss at the FIFO for a certain type of

UPC, the leaky-bucket in this analysis.

For the most part of the analysis, the homogeneous case with all the sources transmitting the same traffic type will be considered. However, the phase difference between sources will be random. In other words, even though the individual traffic patterns are deterministic and periodic, the aggregated incoming traffic at the FIFO buffer will be a sum of the individual sources with a random (and discrete) phase.

**Definition 4.1** *Let $x[n]$ be the outgoing traffic with period $T$ from one source. The incoming traffic at the FIFO buffer from $N$ sources is*

$$X[n] = \sum_{i=1}^{N} x[n - n_i] \tag{4.1}$$

*where $n_i$s are independent, uniformly distributed random variable between 0 and $T$.*

**Definition 4.2** *Let $q$ be the buffer occupancy in a FIFO buffer of size $B$. Cell loss occurs at a time $n_c$ if and only if*

$$X[n_c] + q - r > B$$

*at which time exactly $X[n_c] + q - r - B$ cells are discarded.*

When considering which types of sources will produce the worst performance in the network while still maintaining the contract, the first point to note is that the type of source will not allow the leaky-bucket to ever overflow and will always comply with the contract. In other words, the total available number of cells that are allowed to enter the network will enter to produce the lowest performance. Two types of sources that have been proposed as possible worst types are a two-state source (greedy on-off source) and a three-state source (Fig. 4.7).

The greedy on-off source emits a burst of cells at the peak cell emission rate until the bucket runs out of available tokens and then falls silent waiting for the token buffer to fill up. This occurs periodically depending on the parameters of the system. The three-state source is similar to the greedy on-off source except that it keeps emitting

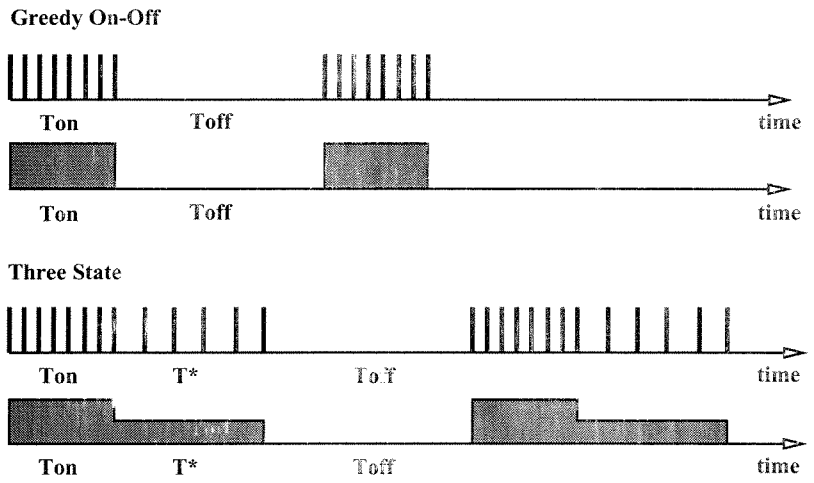



Figure 4.7: Source Types

cells at the average rate ($R$) after the burst. Therefore its operation is to emit a burst of cells at the peak cell emission rate until the bucket runs out of tokens and then emit cells at the token-replenishment rate of the leaky-bucket for some time and then fall silent to allow the tokens to fill up. This would then be repeated again. What can be seen is that the three-state source would have a longer period than the greedy on-off for the same system parameters.

The general belief is that the greedy on-off source gives rise to the worst case traffic as it would have the largest variance possible. However, the three-state source has been proposed [32] as producing longer queues and therefore larger loss.

## 4.3 Discussion of Previous Studies

An initial study of the three-state source compared its performance to that of the greedy on-off source, for a large number of sources into an infinite buffer [32]. The following parameters describe the traffic in those simulations:

Greedy On-Off Source: $N = 99$, $r = 1$, $p = 0.1$, $R = 0.01$, $B = 9.1$

Three-State Source: $N = 99$, $r = 1$, $p = 0.1$, $R = 0.01$, $B = 9.1$, $T^* = 1000$

In two different simulations, 99 sources of one type were fed into the buffer many times with their phase differences with respect to each other being randomly varied at each separate case. The parameters given above produce overall periods of 1000 cells for the greedy on-off and 2000 cells for the three-state, resulting in a number of all possible phase alignments of $1000^{98}$ and $2000^{98}$ respectively. These sources are all transmitting into an *infinite* FIFO buffer.

It was found in those simulations that the three-state sources produced higher buffer occupancy than the greedy on-off sources for the duration that the two situations were simulated. The survivor function, $P[Q > q]$, was obtained where $Q$ was the buffer occupancy. This function was then assumed to approximate the cell loss in a finite buffer. In other words, it was concluded that for an integer $B$, if $P_{Three-State}(Q > B) > P_{On-Off}(Q > B)$, then the three-state source would cause a higher cell loss at a finite buffer of size $B$.

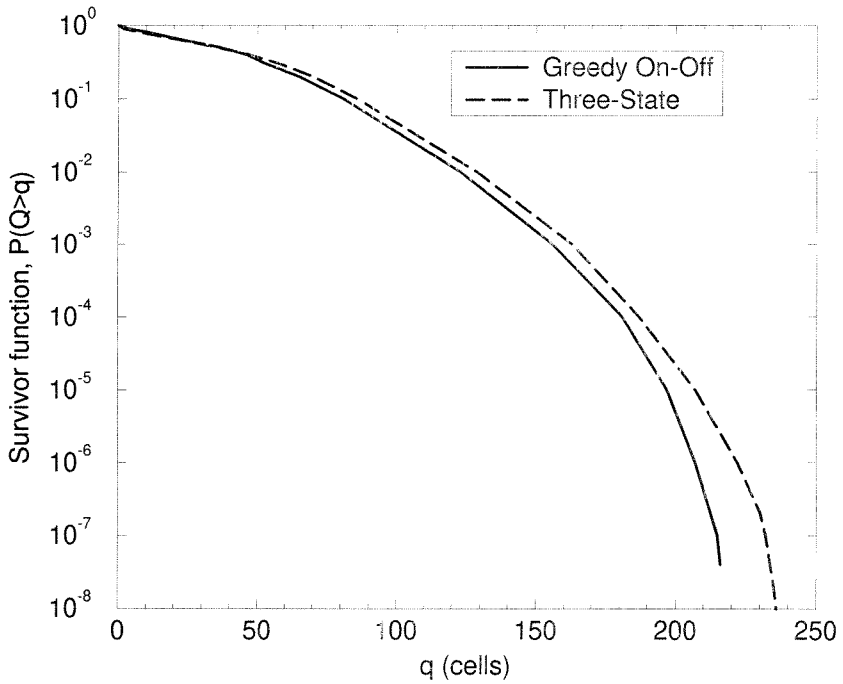



Figure 4.8: Simulation Results For Buffer Occupancy in an Infinite Buffer [32]

Even though later on in this chapter, it'll be presented that the end-result has a high level of truth in it, the assumption above about the relationship between infinite and finite buffers is not true in this situation. It would only be true for input traffic that would be statistically independent of the queue lengths, which is not the case here as the input traffic is periodic and deterministic.

The fact of higher buffer occupancy in the infinite case does not necessarily mean higher cell loss rates in the finite buffer when the arrival process is deterministic. For deterministic arrivals the loss in a finite queue is not necessarily related to the buffer occupancy in the infinite case but more on the method of arrivals, or the process of arrivals, past the finite places in the queue. This problem was simulated [31] and these conclusions help to explain the results obtained therein. Furthermore, the survivor function $P[Q > q]$ can provide more conflicting results if viewed carefully. As an example, here is a simple situation with $M = 2$, $p = 1$, $R = 0.5$, and $r = 1$. For the three-state source, $T^*$ is left as a variable and the overall period of the source is $T$ ($T = 6$ for the greedy on-off case). Then the exact frequency that each possible queue occupancy is going to occur at can be calculated:

- $P[Q = 0] = (4T - 7)/T^2$
- $P[Q = 1] = (.75T^2 - 4.5T + 12)/T^2$
- $P[Q = 2] = (.25T^2 - 3)/T^2$
- $P[Q = 3] = (.5T - 2)/T^2$

The survival function can then be obtained and looked into for different values of $T^*$. When it is plotted for three different values of $T$ – $T = 6$ for greedy-on-off, and $T = 8$ and $T = 24$ for three-state – it is clearly seen (Fig. 4.9) that up to $q = 3$ cells, the three-state sources create a higher buffer occupancy, but just before $q = 3$, there is a crossover.

In other words, the buffer occupancy for the infinite buffer and the cell loss rate in a finite buffer do not provide a one-to-one relationship in this case. In fact, almost in all cases, the traffic that creates a lower buffer occupancy in the infinite buffer causes

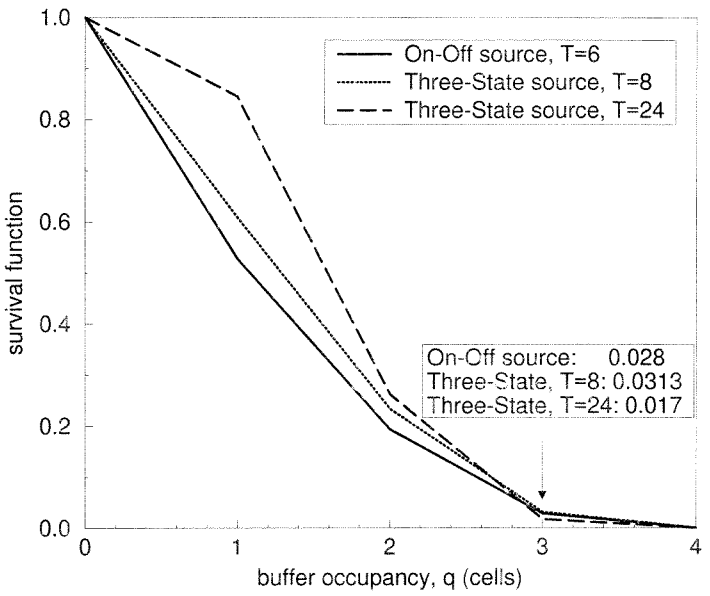



Figure 4.9: Small Counter-Example: Comparison of buffer occupancies for a greedy on-off and two three-state sources

| B | $P[Q > q]$ | | | Cell Loss Rate | | |
|---|---|---|---|---|---|---|
| | Greedy On-Off | Three-State $T = 8$ | Three-State $T = 24$ | Greedy $On - Off$ | Three-State $T = 8$ | Three-State $T = 24$ |
| 1 | 0.194 | 0.234 | 0.262 | 0.111 | 0.094 | 0.038 |
| 2 | 0.028 | 0.031 | 0.017 | 0.028 | 0.016 | 0.002 |

Table 4.2: Buffer Occupancy and Cell Loss Rate

a higher cell loss rate in the finite buffer (Table 4.2). This shows that the survival function for an infinite buffer cannot be used to make conclusions on the finite buffer situation for this deterministic traffic pattern.

### 4.3.1 Positive Slope

Why is this so ? This can be investigated using a continuous time model which afterwards can be discretized.

The source traffic can be characterized using the step function:

$$x_{on-off}(t) = p\,u(t) - p\,u(t - T_{on}),\quad 0 \le t < T \tag{4.2}$$

$$x_{three-state}(t) = p\,u(t) - (p - R)\,u(t - T_{on}) - R\,u(t - T^*),\quad 0 \le t < T \tag{4.3}$$

where

$$u(t) = \begin{cases} 1 & \text{if } t \ge 0 \\ 0 & \text{otherwise} \end{cases}$$

The buffer occupancy is the accumulation of traffic inside the buffer when the input rate, $x(t)$, is higher than the output rate, $r(t)$, which is $r$ for all values of $t$. With the two representations above being one period of the source traffic, the buffer occupancy in an infinite buffer will vary with respect to time according to:

$$q(t) = \int_0^T x(t) - r(t)\,dt \tag{4.4}$$

which will translate into:

$$q_{on-off}(t) = max\{0,\, t\,(p\,(u(t) - u(t - T_{on})) - r\,u(t))\},\quad 0 < t < T \tag{4.5}$$

$$q_{three-state}(t) = max\{0,\, t\,(p\,u(t) - (p - R)\,u(t - T_{on}) \tag{4.6}$$

$$- R\,u(t - T^*) - r\,u(t))\},\qquad 0 < t < T \tag{4.7}$$

For a suitable choice of parameters satisfying $p > r$, $p > R \ge r$, it is clear that $q_{three-state}(t)$ will have values greater than a fixed number $B$ more often than $q_{on-off}(t)$, i.e. higher buffer occupancy. However, when it comes to a finite buffer, the situation changes. The best question to provide insight would be "when are cells lost ?" That was defined in definition 4.2 for the discrete case. When applied to the continuous case, the condition will be:

$$\int_{t-\Delta t}^{t} x(t)\,dt - \int_{t-\Delta t}^{t} r(t)\,dt + q(t - \Delta t) > B \tag{4.8}$$

When each component of the equation above is inspected, one can see that cell

loss can only occur if

$$\frac{d\,q(t)}{dt} > 0 \tag{4.9}$$

which happens only during the peak rate burst duration of the source traffic. The important point here is that both traffic types have the same duration for this period. In other words, the cell loss that they will experience will be exactly equal in terms of cells, but when normalized for a per unit time basis, due to the longer period, three-state source has a lower cell loss rate (CLR). The actual reason for this is that the buffer occupancy does not take into consideration the time variable. On the other hand, CLR is on a per unit time basis resulting in the conclusion that even though two sources might be losing the same number of cells within a single period, the one with the shorter period will lose many more in the long run (Fig. 4.10).

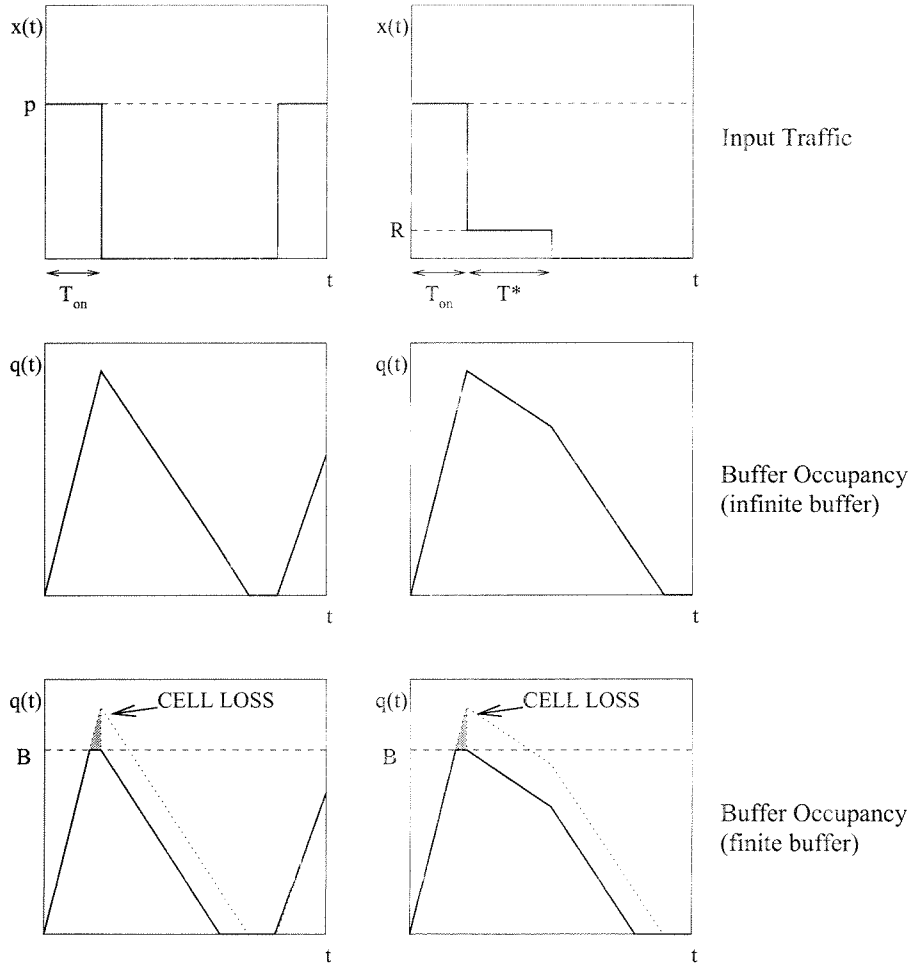



Figure 4.10: Comparison of Finite and Infinite Buffers For Continuous-Time Case

However, that is the simplistic approach of a single phase combination. Does this result still hold true for all possible random phase combinations ? The answer is both yes and no.

## 4.4 Two Sources

The problem contains both continuous and discrete variables and therefore it is important to distinguish between the two types. A simplified model similar to [30] is considered where there are two identical, independent users feeding a finite buffer. If the users were not independent then the worst case would be the greedy on-off source with all sources emitting together in phase. However, when the users or sources are independent then the phase between the sources is a random variable. This gives rise to the probability of cell loss because there is a probability of phase difference between the sources. It is assumed that the two sources are randomly phased, which means one source can be considered as a reference source and the other is then a random amount out of phase. The amount out of phase will determine the number of cells lost. The more in phase the more loss is expected. The probability of each of the possible combinations of out of phase is just the reciprocal of the number of the possible combinations.

Two possibilities are considered: Either two greedy on-off sources or two three-state type sources. The cell losses for all distinct combinations of the two traffic patterns are calculated and averaged and the cell loss rate for each scenario is compared. There are a number of constraints on the problem due to the discrete nature of the variables:

1. The sources may be out of phase only in one cell time units which means that integration cannot be used.

2. The on period of the on-off source is the time taken for the leaky-bucket to run out of tokens while emitting at the peak rate and also to emit any cells which

arrive during this time. In the discrete case for $p = 1$ this is equal to

$$T_{on} = \lfloor (M - R)/(1 - R) \rfloor \tag{4.10}$$

3. Cells are only lost as integer units – no fractional cell loss. Therefore it is necessary to ensure that in using general formulas the cell loss is truncated to an integer. The case of $M \geq 1$ is investigated.

4. The discrete nature of the cell also implies that the FIFO buffer size must be an integer: $B \in Z^+$

5. The service process can be assumed to be continuous. This means a cell can be served as it arrives. Alternatively, one can think of the server waiting until the cell has fully arrived before starting to serve it.

The comparison of cell loss from both source types in the two identical user system is revisited assuming the cell arrival and service processes are discrete. The cell loss rate is computed as the total number of cells lost in one period divided by the total number of cells emitted in one period. The total number of cells lost in any one period is the average of the cells lost by each combination of the two traffic patterns. It is assumed that the minimal phase difference between two sources is one cell time and there are $T$ ($T$ is the period) combinations of the two traffic patterns. The total number of cells emitted in one period $T$ equals $2RT$. Furthermore the following stipulations are placed on the parameters :

- The service rate is at least equal to the peak cell emission rate, $p \leq r$
- The leaky-bucket rate is less than or equal to half the peak cell rate (and hence the service rate), $R \leq p/2$
- Together the peak cell emission and the leaky-bucket rate exceed the service rate, $p + R \geq r$
- The buffer size must be small enough so that cell loss is guaranteed when both sources are in phase, $(2p - r)T_{on} \geq B$

These stipulations ensure that cell loss only occurs when at least one of the sources is emitting at the peak rate. If the service process is assumed to begin as the cell is arriving, the server is named a fast server. Then the cell loss for two greedy on-off sources, $x$ places out of phase with each other, is denoted by $CL(x)$ and is calculated in Equation 4.11

$$CL(x) = max\{0, \lfloor (2p - r)(T_{on} - x) - B \rfloor\} \tag{4.11}$$

where the symbols have the usual meanings. Similarly the cell loss for the three-state sources can be represented by Equation 4.12.

$$CL(x) = max\{0, \lfloor (2p - r)(T_{on} - x) + (p + R - r)(x) - B \rfloor\} \tag{4.12}$$

Alternatively if it is assumed that the server waits until the cell has arrived in the buffer before starting to serve the cell, i.e. the slow server, then the expression for cell loss are modified for the greedy on-off source as follows in Equation 4.13

$$CL(x) = max\{0, \lfloor (2p)(T_{on} - x) - (T_{on} - x - 1/p)r - B \rfloor\} \tag{4.13}$$

and for the three-state slow server the cell loss will be given in Equation 4.14.

$$CL(x) = max\{0, \lfloor (2p)(T_{on} - x) - (T_{on} - x - 1/p)r + (p + R - r)(x) - B \rfloor\} \tag{4.14}$$

To calculate the CLR $x$ is allowed to vary over all phase possibilities and then averaged over the number of combinations and also normalized by the period of each source. The number of cells transmitted by both sources will be $2RT$ and the number of phase combinations will be $T$, so the CLR is given by Equation 4.15.

$$CLR = \frac{\sum_{x=0}^{T-1} CL(x)}{T^2} \tag{4.15}$$

To show that there is no single worst type traffic for two identical sources, a

counter example to the traditional theory of the greedy on-off being the worst case is presented. This example shows that the three-state source can produce higher CLR for integer values of variables chosen. As mentioned previously two different types of servers are initially considered: The fast serving server and the slow serving server.

### 4.4.1 Fast Serving Server

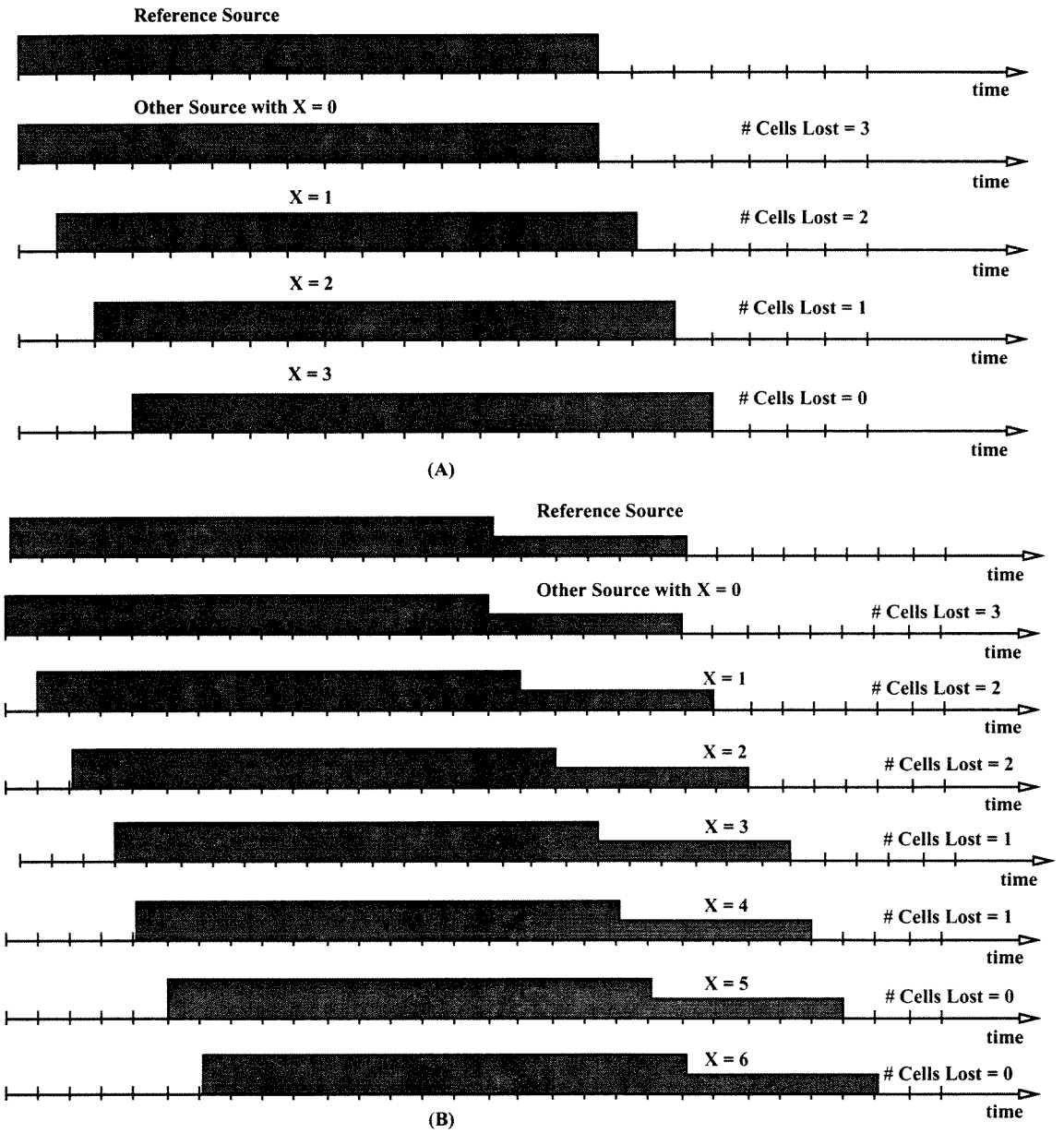



Figure 4.11: Fast Server, (A) Greedy On-Off Source, (B) Three-State Source

Assuming that the cell is served as it arrives here is an example showing the three-state source gives rise to greater cell loss in a finite buffer than the on-off source. The following system parameters are used :

- $p$, peak cell emission = 1
- $r$, service time = 1
- $R$, leaky-bucket rate = .5
- $M$, leaky-bucket capacity = 8
- $B$, buffer size = 12
- $T^*$ (for three-state source) $\geq ((2p - r)T_{on} - B)/(p - R) = 6$

It takes 8 cell times to use the tokens up but in that time 4 more cells are allowed through because of the constant replenishment rate of the tokens and while they provide service 2 more arrive and then finally one arrives and the token-buffer is empty. Therefore the amount of time that the source is on and emitting cells at the peak rate is given by $T_{on} = \lfloor (pM - R)/(p - R) \rfloor = 15$. When the bucket is full it takes $M/R$ seconds to empty normally; however, here one time period has already elapsed so $T_{off} = 15$.

There are 30 phase combinations for the greedy on-off source patterns; however, sources that are too far out of phase do not give rise to cell loss. For the greedy on-off sources by examining Equation 4.11 it is concluded that sources that are 3 or more time units out of phase do not produce any cell loss. It should be remembered that this loss can occur when the second source is a little advanced from the reference and also when it's so advanced that it is almost back in phase with the reference. This can be seen in Figure 4.11. The cell loss for the two state sources is in total 9 cells, which is calculated from 3 cells lost when in phase, 2 cells lost when either one out of phase and also 29 out of phase, and 1 cell lost when either two out of phase or 28 out of phase. For all other phasings there is no cell loss. Therefore using Equation 4.15 the cell loss rate can be calculated to be 0.01.

For the three-state sources, there are 36 possible phase combinations and by examining Equation 4.12 loss can occur up to 4 units out of phase. The total number of cells lost over all possible phasings can be seen in Figure 4.11 and is 15 cells. By using Equation 4.15, the loss is calculated to be 0.01157. Therefore the three-state source produces higher loss than the greedy on-off source for the fast server.

### 4.4.2 Slow Serving Server

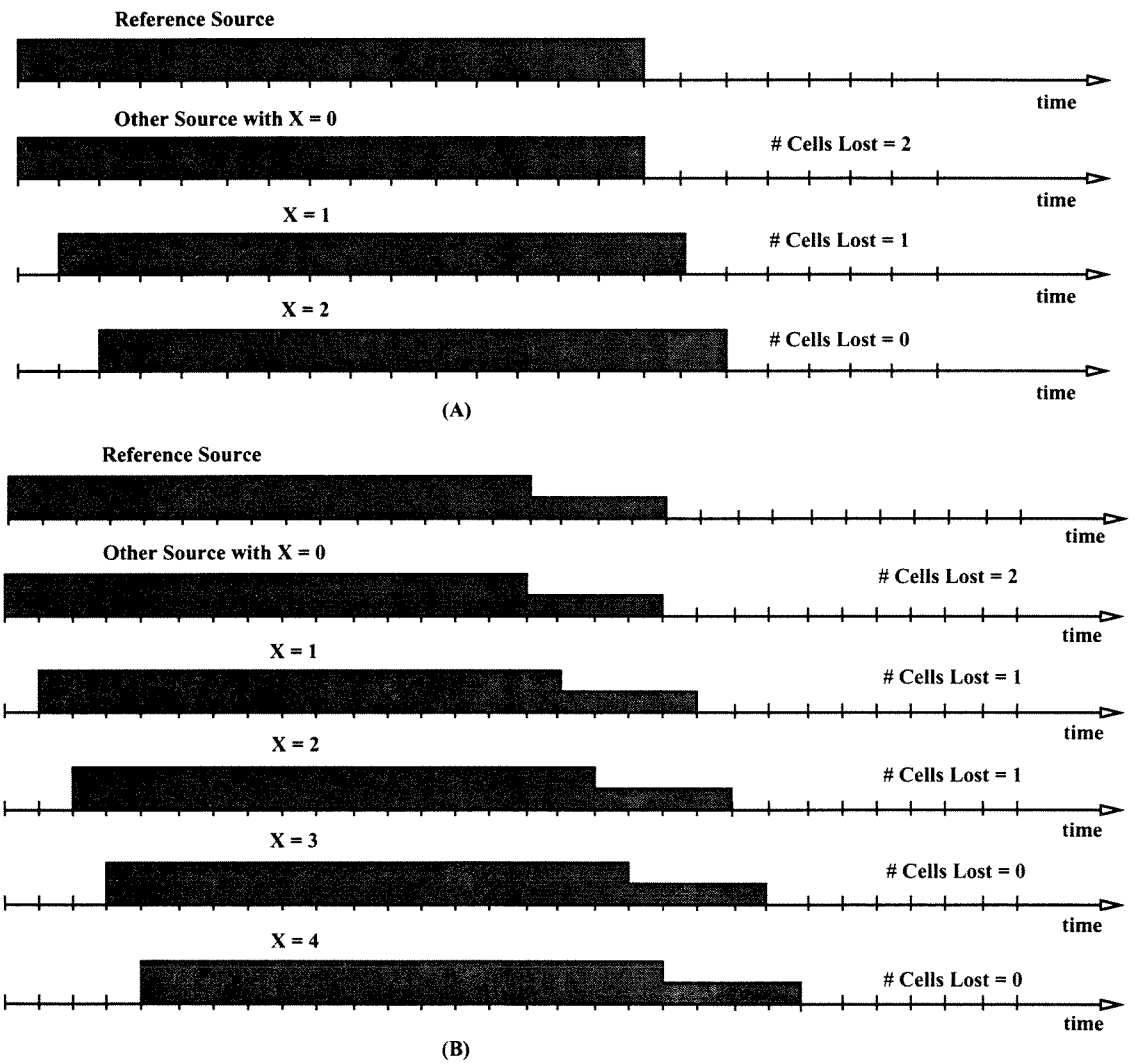



Figure 4.12: Slow Server, (A) Greedy On-Off Source, (B) Three-State Source

If the service process is assumed to begin serving a cell only after it has fully arrived into the buffer, there is a similar counter example. The following system parameters

are used :

- $p$, peak cell emission = 1
- $r$, service time = 1
- $R$, leaky-bucket rate = .5
- $M$, leaky-bucket capacity = 8
- $B$, buffer size =14
- $T^*$ (for three-state source) $\geq ((2p - r)T_{on} - B + r/p)/(p - R) = 4$

Similar to the fast serving server $T_{on} = T_{off} = 15$. For the greedy on-off sources by examining Equation 4.13 it is concluded that sources that are 2 or more time units out of phase do not produce any cell loss. This can be seen in Figure 4.12. The cell loss for the on-off sources is over all possible phase combinations equal to 4 cells and by using Equation 4.15, the cell loss rate can be calculated to be 0.00444.

For the three-state sources cell loss can occur up to 2 time units out of phase. This is concluded by examining Equation 4.14 for the cell loss for a three-state source and this is also seen in Figure 4.12. Here in total 6 cells are lost over all possible phase combinations and so again by using Equation 4.15 the cell loss rate can be calculated to be 0.00519. Therefore the three-state source produces higher loss than the greedy on-off source for the slow serving server, too.

Therefore regardless of how the service is achieved in the buffer there is an example of the three-state source producing more loss than the greedy on-off source.

## 4.5 Effect of Buffer Size

However, the example presented in the previous section doesn't mean that the three-state source is worse than the on-off source in general. In fact, if the very same example is considered for other value of $B$, an interesting situation is encountered: The answer to the question which source type creates higher CLR is dependent on

the size of the buffer (Fig. 4.13). When there is no buffer ($B = 0$) the two sources are creating the same amount of cell loss (this will be proven to hold true for $N$ sources in a later section). As $B$ increases, first the greedy on-off source achieves a higher CLR, but as $B$ further increases, the CLR for the three-state source becomes higher. At the end, they reach the zero-loss state when the buffer size becomes sufficiently high ($B = 15$ in this case).

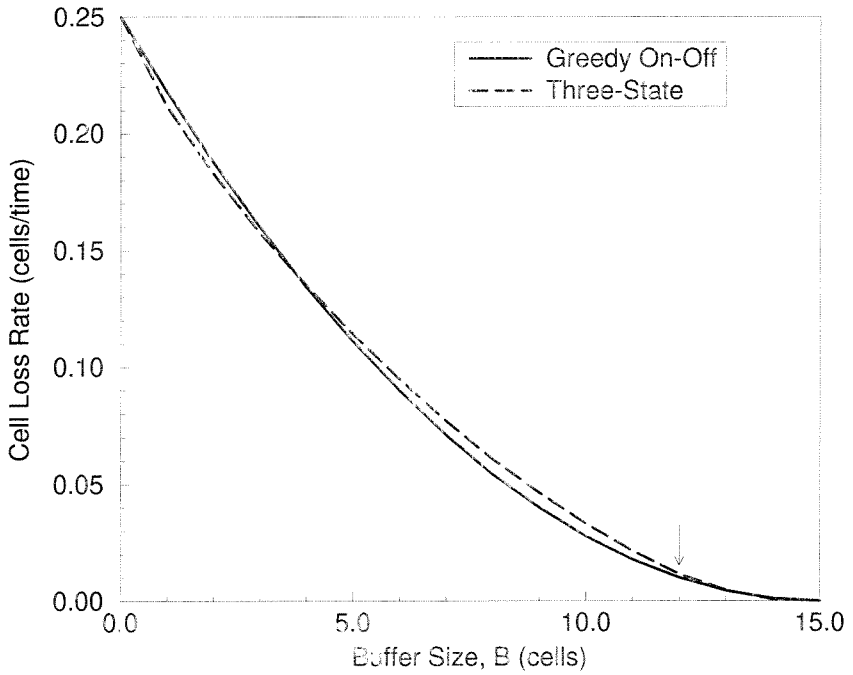



Figure 4.13: CLR vs Buffer Size For The Two Types of Traffic

From this example alone, it is apparent that:

- neither the greedy on-off source, nor the three-state source create a CLR that is uniformly higher overall.
- the characteristics of cell loss is heavily dependent on the buffer size, $B$.

These suppositions are further supported by the data in Fig. 4.14[2]. Same leaky-bucket parameters hold for all six graphs. The only variation is the duration of the

[2]All the data presented in the graphs in this chapter can be found in Appendix A

average-rate state, $T^*$, of the three-state source. The two CLR lines cross over in almost all of the plots; more than once in some. However, it's the trend that as $T^*$ increases, the region where the three-state source has a higher CLR decreases and greedy on-off has a higher CLR for all values of $B$ after a certain value of $T^*$.

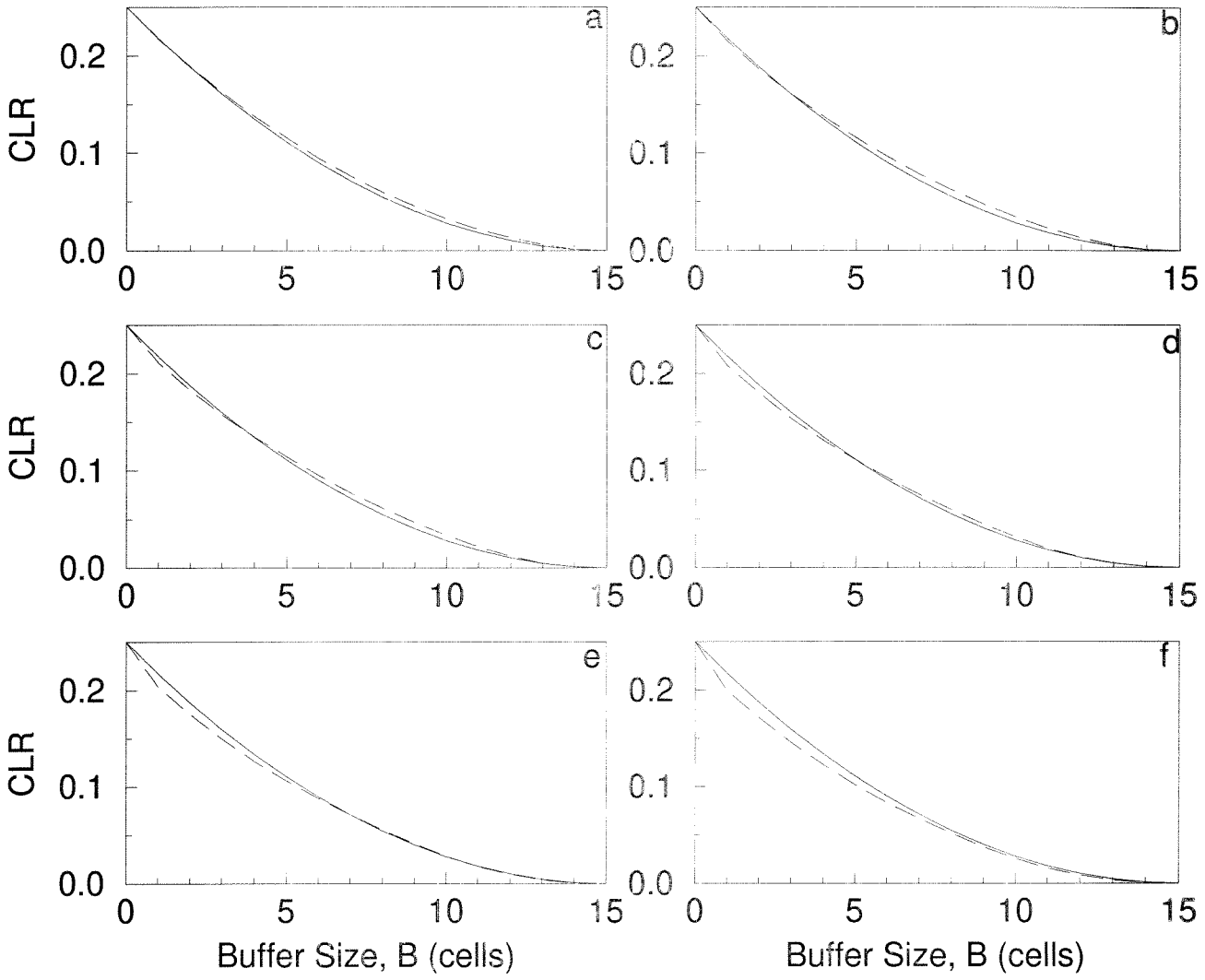



Figure 4.14: CLR vs Buffer Size For Two Types of Traffic With Varying $T^*$.
On all graphs, solid line is the CLR for the greedy on-off source and the dashed line for the three-state source. $N = 2$, $M = 8$, $R = 0.5$, $r = 1$, $p = 1$
a) $T^* = 2$, b) $T^* = 4$, c) $T^* = 6$, d) $T^* = 8$, e) $T^* = 10$, f) $T^* = 12$

The same trend can be observed in another different example (Fig. 4.15) of the same situation. This time the token buffer size is increased to $M = 10.5$ while keeping everything else the same. This increases the burst-length to 20 cells. Once again, the

average-rate state of the three-state traffic is varied between graphs.

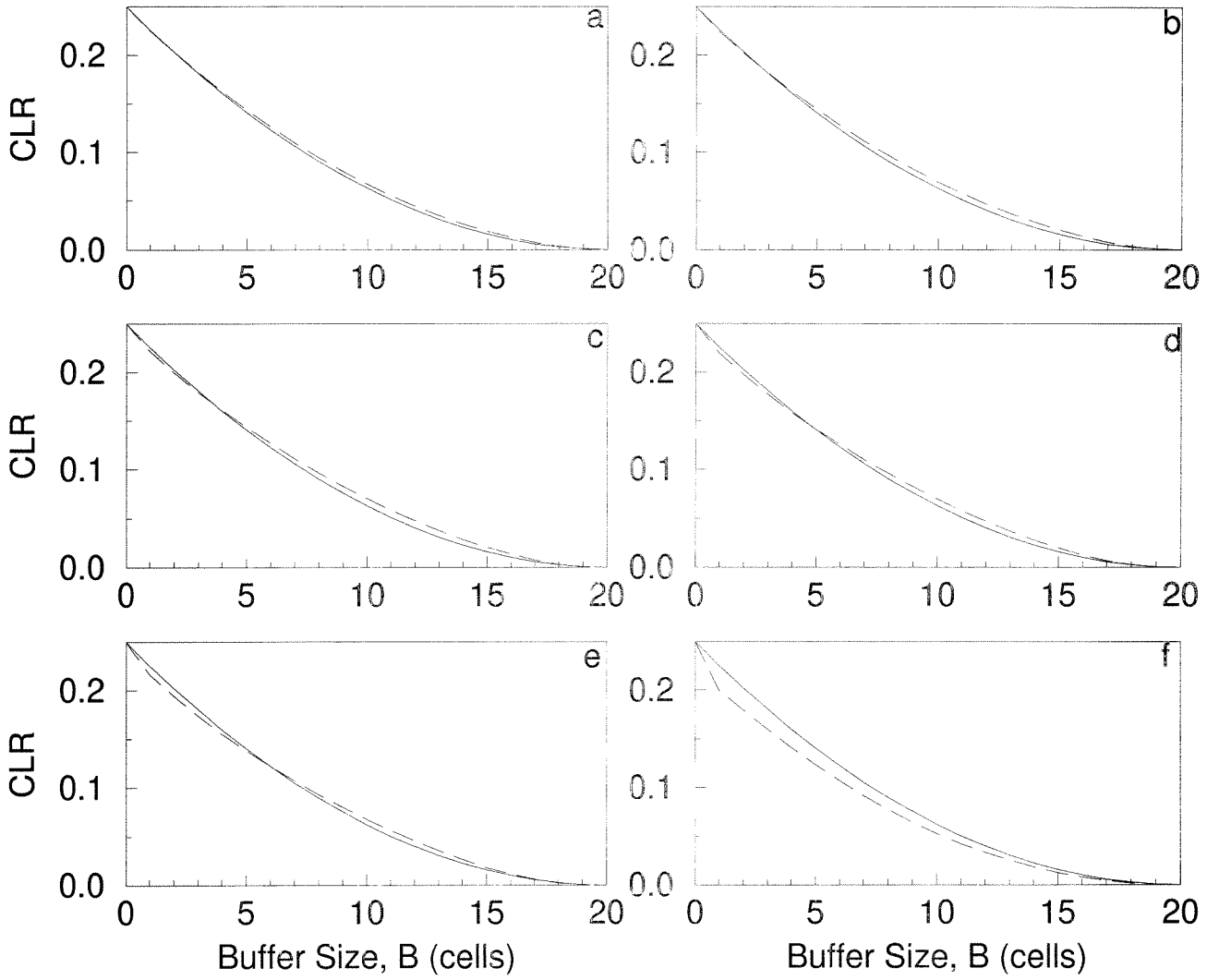



Figure 4.15: CLR vs Buffer Size For Two Types of Traffic With Varying $T^*$.
On all graphs, solid line is the CLR for the greedy on-off source and the dashed line for the three-state source. $N = 2$, $M = 10.5$, $R = 0.5$, $r = 1$, $p = 1$
a) $T^* = 2$, b) $T^* = 4$, c) $T^* = 6$, d) $T^* = 8$, e) $T^* = 10$, f) $T^* = 20$

This behaviour is observed in many other examples. The conclusion is that for the case of two leaky-bucket controlled sources transmitting into the same buffer, neither the situation where those sources produce greedy on-off type traffics with a certain set of leaky-bucket parameters, nor the situation where they produce three-state traffics with the same parameters is the worst case uniformly. The CLR difference between the two situations varies with respect to the buffer size, $B$.

## 4.6 Multiple Sources

The first question to be answered is whether the results of the previous section holds true for more than two sources. In fact, do they hold true for any $N$ ? After all, the first simulations on the topic included long periods and a large number of sources. Furthermore, the practical real-life situations will contain a large number of sources, possibly on the order of hundreds of thousands at busy nodes.

The answer to this question, unfortunately, is not easily answered for all values of $N$, even though there is strong evidence that it holds true for $N > 2$. The following examples show that this holds true in many cases where there are more sources than only two. The restrictions placed on these sources are once again that they are leaky-bucket compliant with the same parameters and that $NR \leq r$ so that the server is never overloaded.

The following example (Fig. 4.16) contains three sources with $M = 7$ and $R = 1/3$. Once again the peak rate and the service rate are unity. The results are very similar to those of the two-sources case. Once again the CLR-graphs cross over, showing that neither traffic type is uniformly worse than the other. Furthermore, the dominance of greedy on-off CLR over the CLR for three-state also appears in the same manner as $T^*$ increases.

Similar situations are encountered when four sources (Fig. 4.17 a,b) and five sources (Fig. 4.17 c,d) are considered.

In these examples – and also in general– there are two points of special interest. The first and also the trivial one is the buffer size for at which the CLR goes down to zero. This point is trivial because it's very easily obtained when the worst phase combination occurs when all the sources transmit in phase. If the buffer is large enough at this point to accept all the excess cells transmitted in a burst length, $T_{on}$ by $N$ sources and not be able to be serviced, then there will never be any cell loss for that buffer.

**Theorem 4.1** *For $N$ leaky-bucket controlled sources with parameters $p = 1$, $M$, and $R$, transmitting into the same FIFO buffer of size $B$ and service rate $r$, $CLR = 0$ if*

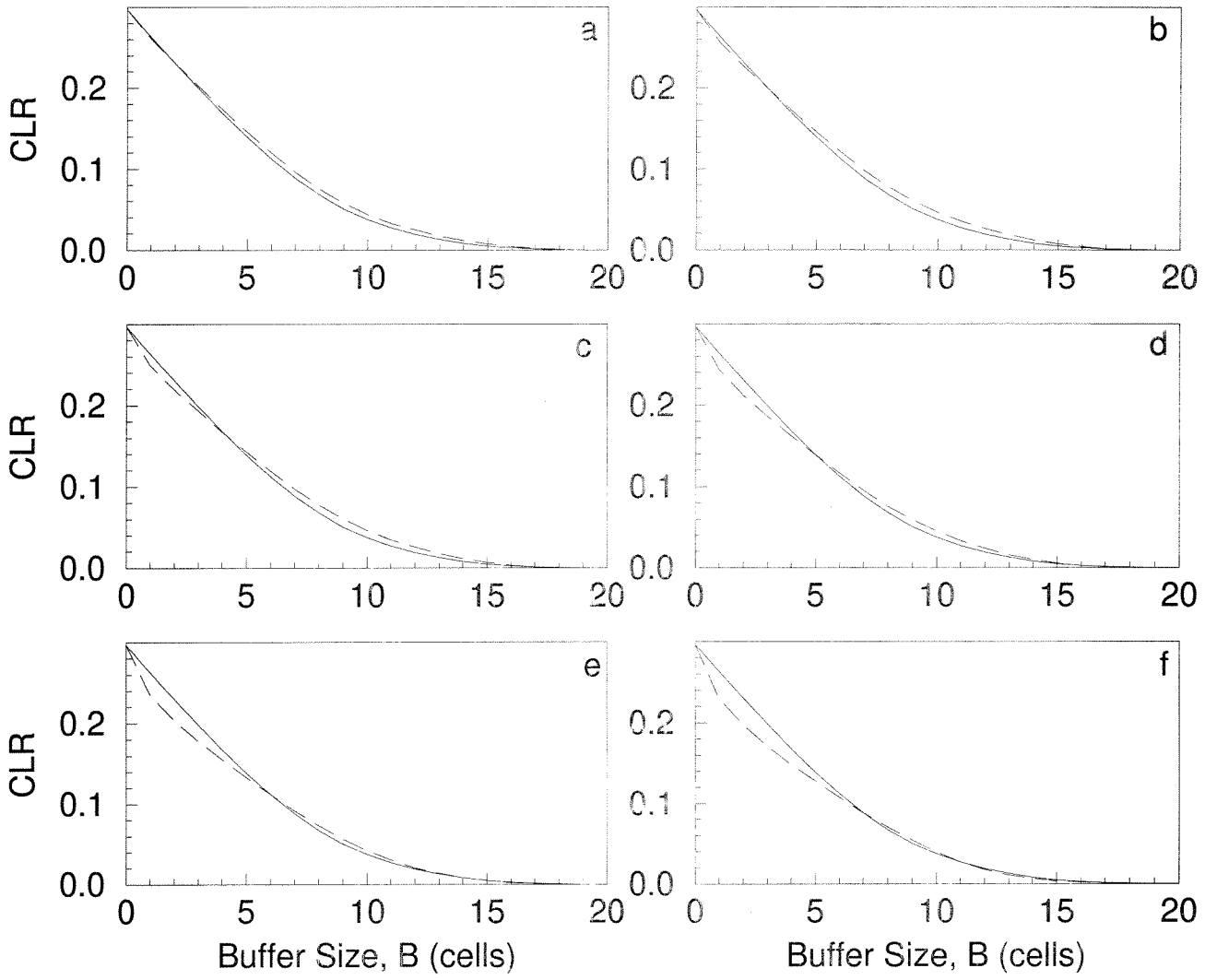



Figure 4.16: CLR vs Buffer Size For Two Types of Traffic With Varying $T^*$.
On all graphs, solid line is the CLR for the greedy on-off source and the dashed line for the three-state source. $N = 3$, $M = 7$, $R = 1/3$, $r = 1$, $p = 1$
a) $T^* = 3$, b) $T^* = 6$, c) $T^* = 9$, d) $T^* = 12$, e) $T^* = 15$, f) $T^* = 18$

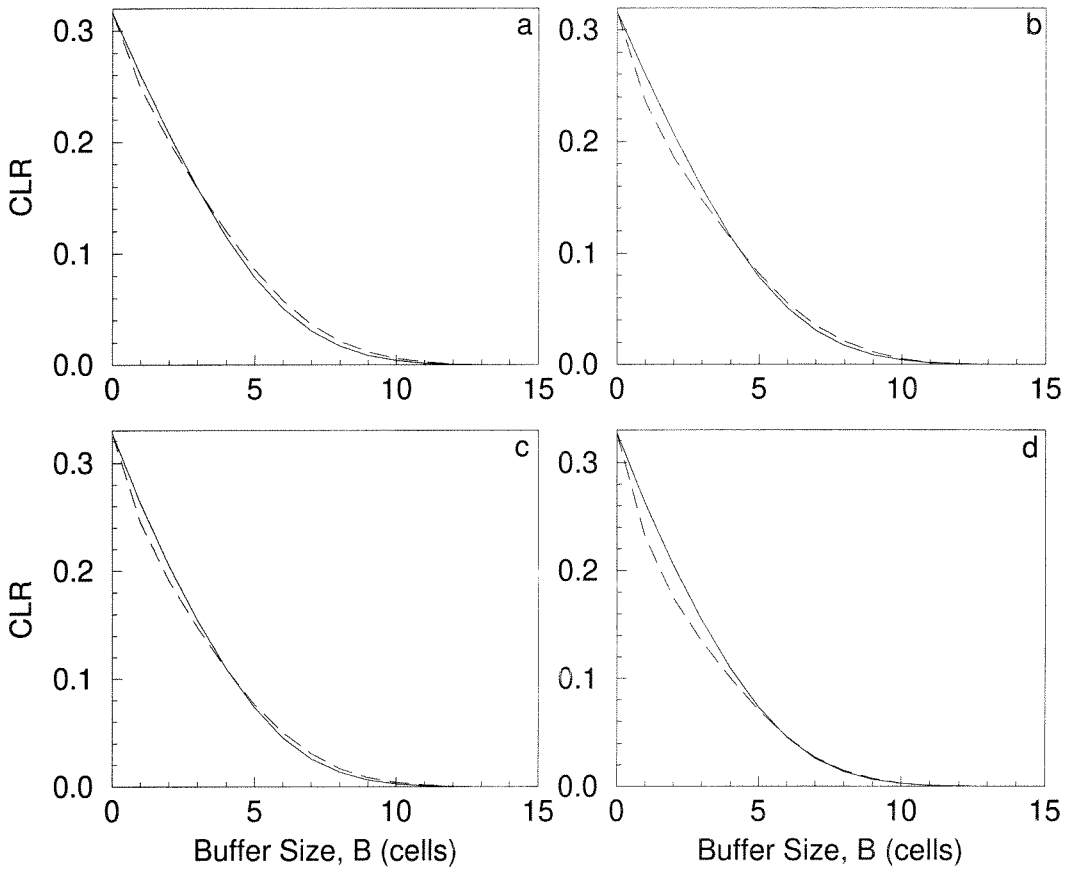



Figure 4.17: CLR vs Buffer Size For Two Types of Traffic With Varying $T^*$. On all graphs, solid line is the CLR for the greedy on-off source and the dashed line for the three-state source.
a) $N = 4$, $T^* = 4$, b) $N = 4$, $T^* = 8$ c) $N = 5$, $T^* = 5$, d) $N = 5$, $T^* = 10$

*and only if*

$$B \geq N \left\lfloor \frac{1\,M - R}{1 - R} \right\rfloor (1 - r) \tag{4.16}$$

*provided that* $r \geq NR$.

*Proof:*

When all sources are in phase then during a burst $T_{on}$ exactly $NT_{on}p$ cells arrive at the buffer. Out of these, only $T_{on}r$ of them can be serviced during this time, leaving $NT_{on}p - T_{on}r$ to be placed in the buffer. Consequently, if CLR is zero, then the buffer must be big enough to accommodate at least this many cells. Furthermore, if the buffer is this big, then there will not be any cell loss.

The other point that is also of interest is the no-buffer case, the point where $B = 0$.

At this point, all the sources with average rate, $R$, create the same amount of cell loss regardless of the shape of the traffic.

**Theorem 4.2** *When $B = 0$, for $N$ sources of average rate $R = 1/N$ transmitting onto the same server with service rate $r = 1$, the CLR is*

$$CLR(N) = \sum_{i=1}^{N}(i-1)\begin{pmatrix} N \\ i \end{pmatrix}(1/N)^i\,(1-1/N)^{N-i} \tag{4.17}$$

*Proof:*

For $B = 0$ point, the shape of the traffic is not important. The only important part is the statistics of it. Given that $r = 1$ and $B = 0$ cell loss occurs when more cells than one arrive at the server simultaneously. If $i$ cells arrive at time $n$, then $i-1$ cells are lost. These $i$ cells could have come from $\begin{pmatrix} N \\ i \end{pmatrix}$ different combinations of sources and the occurrence of this event is given by the probability that a cell occupies a time slot raised to the power $i$ for that many cells multiplied by the probability a time slot is empty raised to the power $N-i$ since that many sources didn't transmit anything at that time instant. When this is summed up the CLR is obtained.

This point is of further interest because of the following corollary that derives from this theorem:

**Corollary 4.1** *When $B = 0$, for $N$ sources of average rate $R = 1/N$ transmitting onto the same server with service rate $r = 1$,*

$$\lim_{N\to\infty} CLR(N) = e^{-1}$$

*Proof:*

$$CLR(N) = \sum_{i=1}^{N}(i-1)\begin{pmatrix} N \\ i \end{pmatrix}(1/N)^i\,(1-1/N)^{N-i}$$

$$= \sum_{i=1}^{N} i \begin{pmatrix} N \\ i \end{pmatrix} (1/N)^i (1-1/N)^{N-i} - \sum_{i=1}^{N} \begin{pmatrix} N \\ i \end{pmatrix} (1/N)^i (1-1/N)^{N-i}$$

$$= \underbrace{\sum_{i=1}^{N} i \begin{pmatrix} N \\ i \end{pmatrix} (1/N)^i (1-1/N)^{N-i}}_{f_1(N)} - \underbrace{(1-(1-1/N)^N)}_{f_2(N)}$$

$$\lim_{N\to\infty} f_2(N) = 1 - e^{-1}$$

$$\lim_{N\to\infty} f_1(N) = \lim_{N\to\infty} \sum_{i=1}^{N} i \begin{pmatrix} N \\ i \end{pmatrix} (1/N)^i (1-1/N)^{N-i}$$

To evaluate $f_1(N)$ consider the following:

$$(X+Y)^N = \sum_{i=0}^{N} \begin{pmatrix} N \\ i \end{pmatrix} (X)^i (Y)^{N-i}$$

If one takes the derivative of both sides with respect to $X$

$$N(X+Y)^{N-i} = \sum_{i=0}^{N} i \begin{pmatrix} N \\ i \end{pmatrix} (X)^{i-1} (Y)^{N-i}$$

Let $X = 1/N$ and $Y = 1 - 1/N$.

$$N = N \sum_{i=0}^{N} i \begin{pmatrix} N \\ i \end{pmatrix} (1/N)^i (1-1/N)^{N-i}$$

$$f_1(N) = \sum_{i=0}^{N} i \begin{pmatrix} N \\ i \end{pmatrix} (1/N)^i (1-1/N)^{N-i} = 1$$

$$\lim_{N\to\infty} CLR(N) = 1 - (1 - e^{-1}) = e^{-1}$$

This corollary marks the $B = 0$ graphs provided up to now. One very interesting aspect of this result is that no matter how many sources there are, the network can guarantee a known percentage of the sources by sacrificing a number of others. And this sacrifice does not grow without bounds, but is limited. Furthermore, it's the same for any shape of traffic even when it's not leaky-bucket controlled as long as the

average rate is $R = 1/N$.

The previous theorem can actually be generalized to cover all ranges of average rates and all server rates rather than the special case used in the previous examples.

**Theorem 4.3** *When $B = 0$, for $N$ sources of average rate $R$ transmitting onto the same server with service rate $r$, the CLR is*

$$CLR(N, R) = \sum_{i=r}^{N} (i - r) \begin{pmatrix} N \\ i \end{pmatrix} (R)^i (1 - R)^{N-i} \tag{4.18}$$

*Proof:*

The proof for this theorem is the same as that of Theorem 4.2. Given that $r$ and $B = 0$ cell loss occurs when more than $r$ cells arrive at the server simultaneously. If $i$ cells arrive at time $n$, then $i - r$ cells are lost. These $i$ cells could have come from $\begin{pmatrix} N \\ i \end{pmatrix}$ different combinations of sources and the occurrence of this event is given by the probability that a cell occupies a time slot, namely $R$, raised to the power $i$ for that many cells multiplied by the probability a time slot is empty, $1 - R$, raised to the power $N - i$ since that many sources didn't transmit anything at that time instant. When this is summed up the CLR is obtained.

This once again is independent of the traffic shape.

## 4.7 Worse Than Worse – Pseudo-n-State Sources

Now that it's established that on neither the greedy on-off source nor the three-state source is uniformly worse than the other for all values of $B$ in many cases, the question that comes to mind is whether there is some other source like that, other than those two considered up to this point. An affirmative answer would make the previous discussion moot whereas a negative answer would reinforce it.

Unfortunately no definite answer to this question was found. The results that are presented in this section, though, seem to point in the negative direction for existence of an absolute worst source. However, some other source types are found and they

happen to be a further extension to the two-state, three-state discussion. This general family of traffic shapes will be referred to as pseudo-n-state sources (Fig. 4.18).

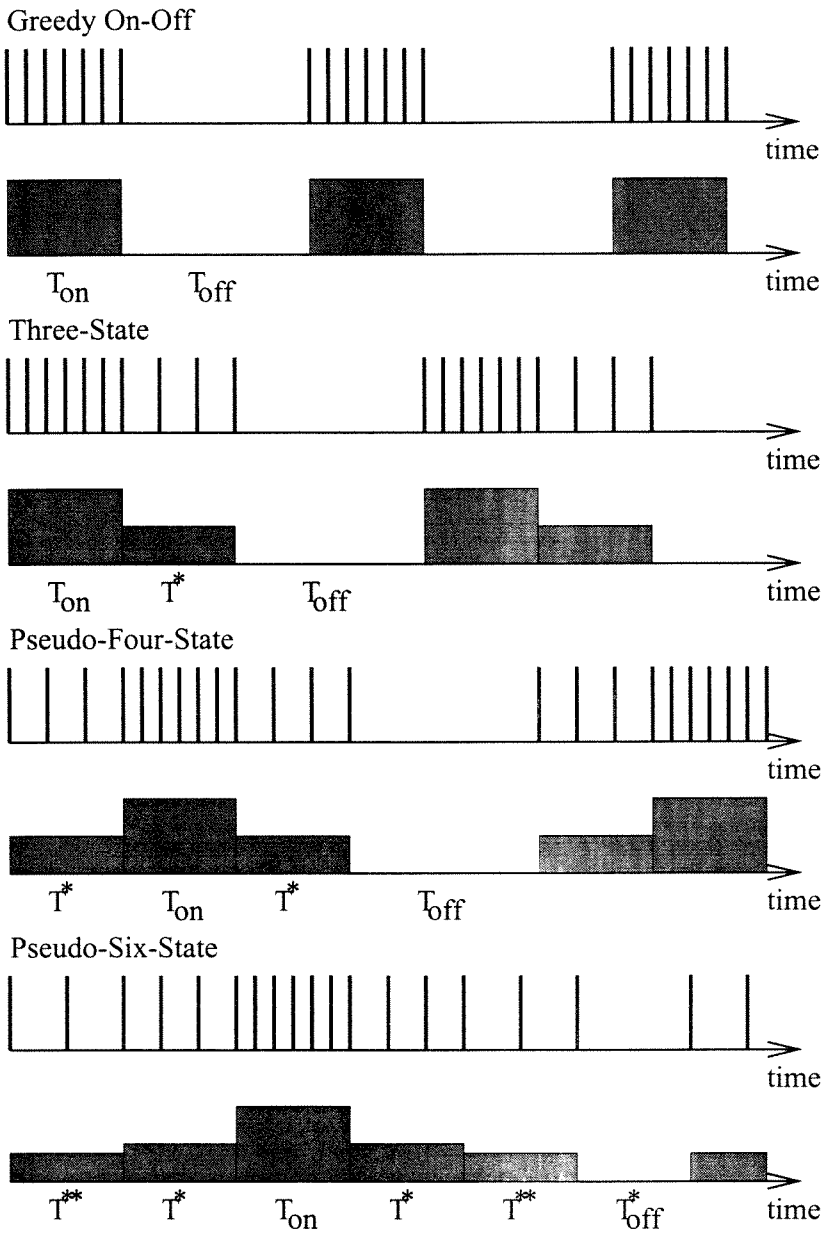



Figure 4.18: Different Types of Sources

An exhaustive search for patterns that might create a higher cell loss was launched and all possible traffic patterns compliant with a given set of leaky-bucket parameters were compared. This revealed a group of sources, namely pseudo-n-state sources, that have higher CLR than both of the other two on some occasions. The first example of

this group is the pseudo-three-state source:

$$\begin{aligned} x_{pseudo-four-state}(t) &= R\,u(t) + (p-R)\,u(t-T^*) - (p-R)\,u(t-T^*-T_{on}) \\ &\quad - R\,u(t-2\,T^*-T_{on}) \qquad 0 \le t < T \end{aligned} \tag{4.19}$$

This traffic shape is very similar to a symmetric three-state source that has an extra average-rate state. The next member of this group would be a pseudo-six-state source:

$$\begin{aligned} x_{pseudo-six-state}(t) &= R'\,u(t) + (R-R')\,u(t-T^{**}) \\ &\quad + (p-R)\,u(t-T^{**}-T^*) - (p-R)\,u(t-T^{**}-T^*-T_{on}) \\ &\quad - (R-R')\,u(t-T^{**}-2\,T^*-T_{on}) \\ &\quad - R'\,u(t-2\,u(t-T^{**}-2\,T^*-T_{on}) \qquad 0 \le t < T \end{aligned} \tag{4.20}$$

When the CLRs of the pseudo-four-state source are obtained and compared to those of the greedy on-off and the three-state sources (Fig. 4.19), it's observed that this source produces a line that is even further skewed than that of the three-state source. In other words, in the region where the three-state source has a higher CLR than the greedy on-off source, the pseudo-four state source has even higher.

The same trend can be observed on the pseudo-six state source. However, for the same leaky-bucket parameters used in the previous example, the pseudo-six-state source provides an interesting piece of information: Its CLR is lower than that of the pseudo-four-state source's for all values of $B$ even though it follows the same shape and is higher than that of the three-state source's in the same $B$-range. In fact, this is true for both values of $T^{**}$ provided (Fig. 4.20).

This is a clear indication that for a given set of leaky-bucket parameters, there is one source in that family that will provide the highest CLR values in the region being considered with having to have to take $n$ to be very large. It's also highly likely

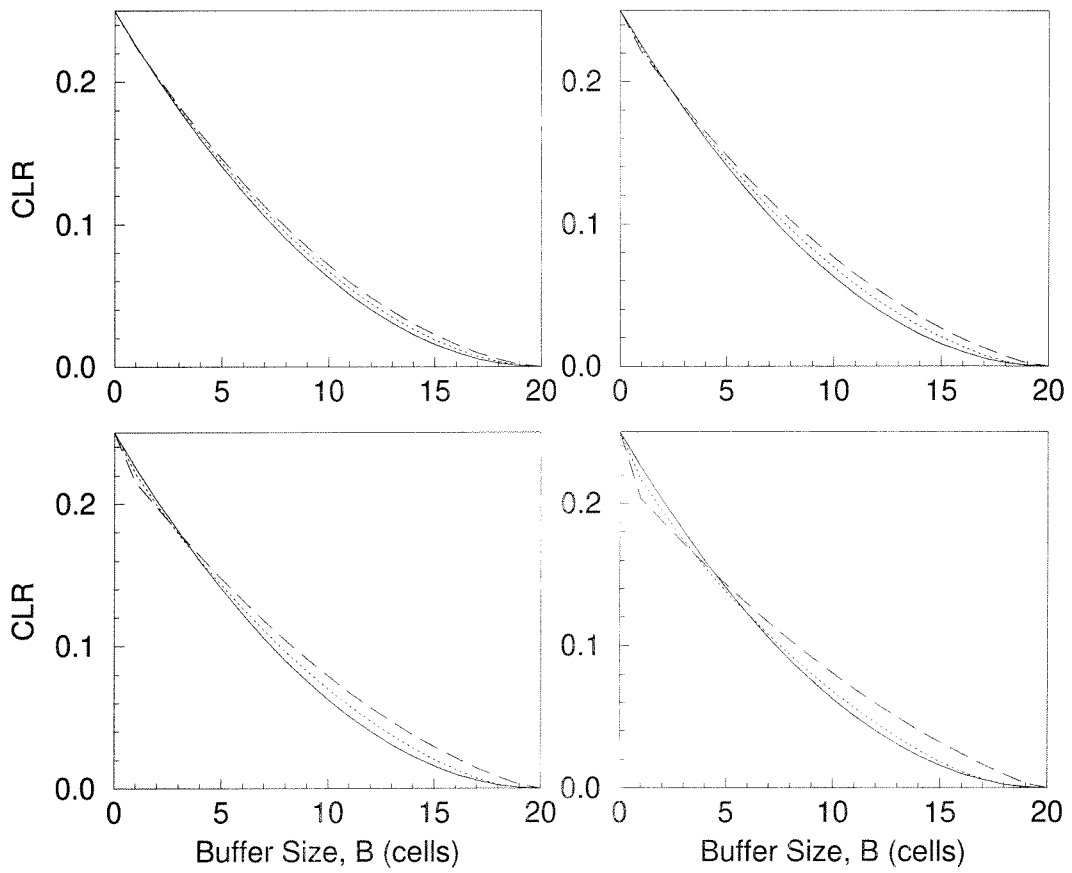



Figure 4.19: CLR vs Buffer Size For Three Types of Traffic With Varying $T^*$. On all graphs, solid line is the CLR for the greedy on-off source, the dotted line for the three-state source, and the dashed line for the pseudo-four-state source. $N = 2$, $M = 10.5$, $R = 0.5$
a) $T^* = 2$, b) $T^* = 4$ c) $T^* = 6$, d) $T^* = 10$

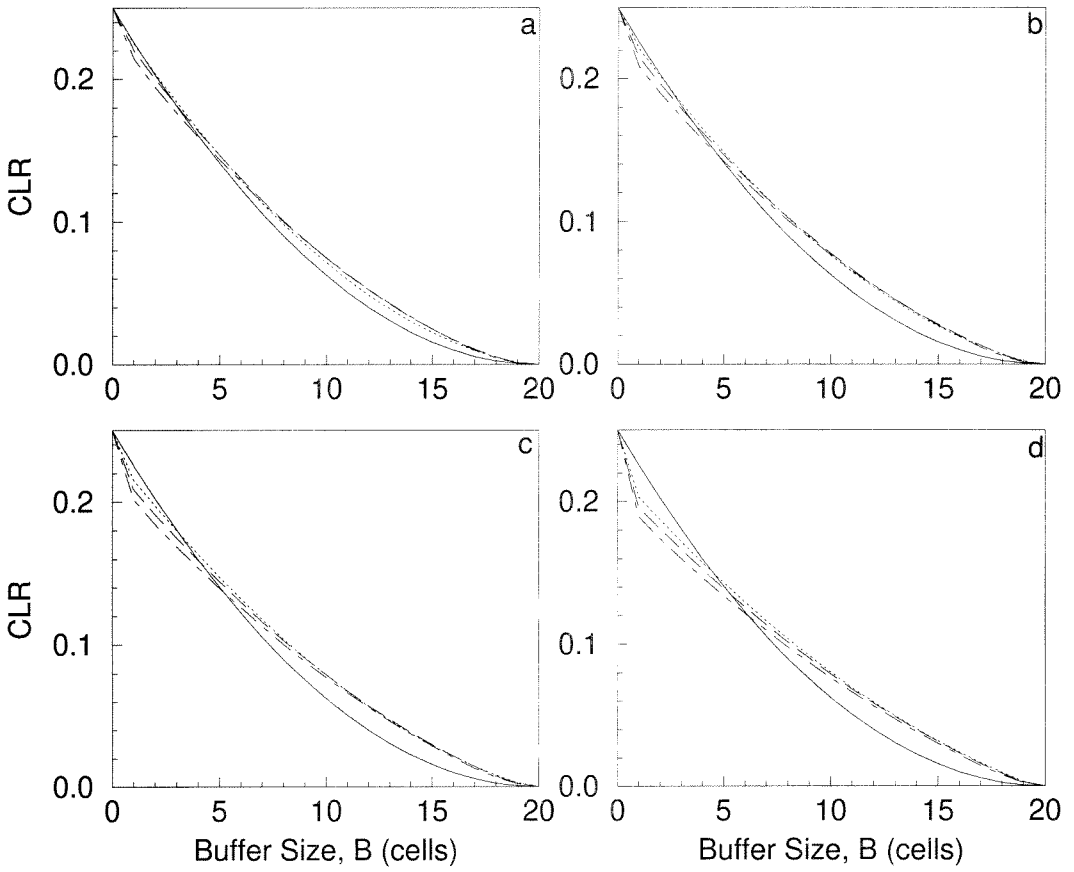



Figure 4.20: CLR vs Buffer Size For Three Types of Traffic With Varying $T^*$. On all graphs, solid line is the CLR for the greedy on-off source, the dotted line for the pseudo-four-state source, the dashed line for the pseudo-six-state source with $T^{**} = 3$, and the dot-dashed line for the pseudo-six-state source with $T^{**} = 6$. $N = 2$, $M = 10.5$, $R = 0.5$
a) $T^* = 2$, b) $T^* = 4$ c) $T^* = 6$, d) $T^* = 10$

that this source is the pseudo-four-state source when considering the presented data. However, the relative behaviour of these traffic shapes might be different enough for longer periods in a way to enable a higher-level pseudo-n-source to have a higher CLR.

The following three graphs (Figs. 4.21, 4.22, 4.23) show that the results hold true not only for two sources, but for higher values of $N$, too.

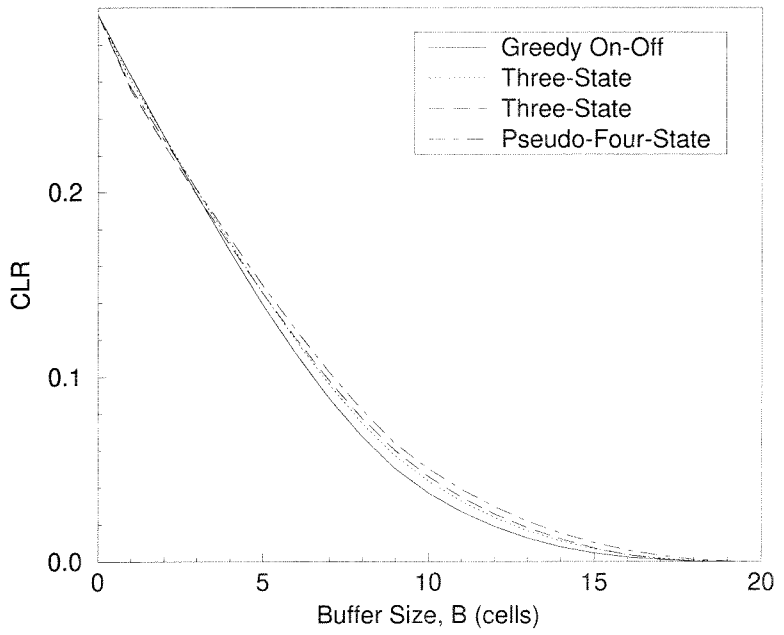



Figure 4.21: CLR vs Buffer Size For Various Types of Traffic
The first three-state source has $T^* = 3$ and the second $T^* = 6$.

## 4.8 Conclusion

It is important to know the nature of the traffic one produces and/or one receives so that the parameters can be adjusted to have the least impact on the incoming or outgoing traffic. The data displayed in this chapter leads to the conclusion that this might not be an easy process in ATM. The promise of the pseudo-four-state source in providing the highest CLRs in the region where on-off doesn't is certainly clouded

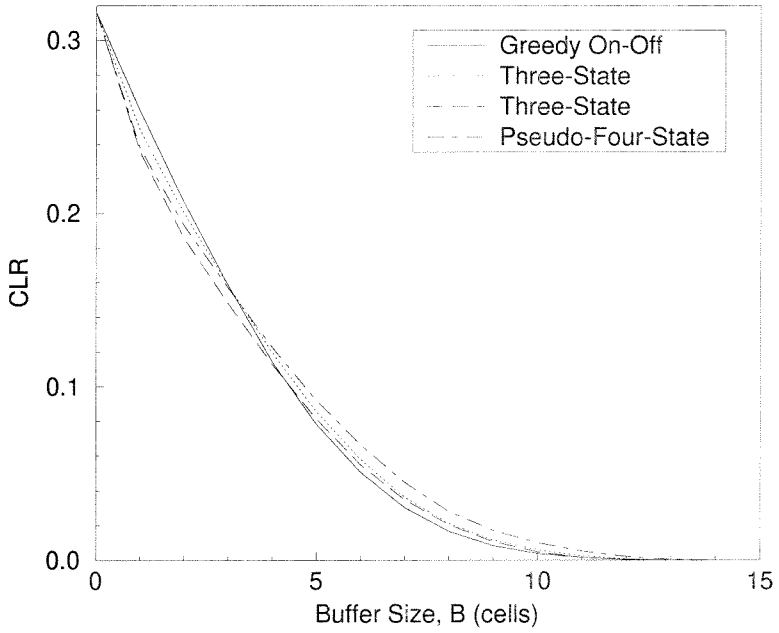



Figure 4.22: CLR vs Buffer Size For Various Types of Traffic
The first three-state source has $T^* = 4$ and the second $T^* = 8$.

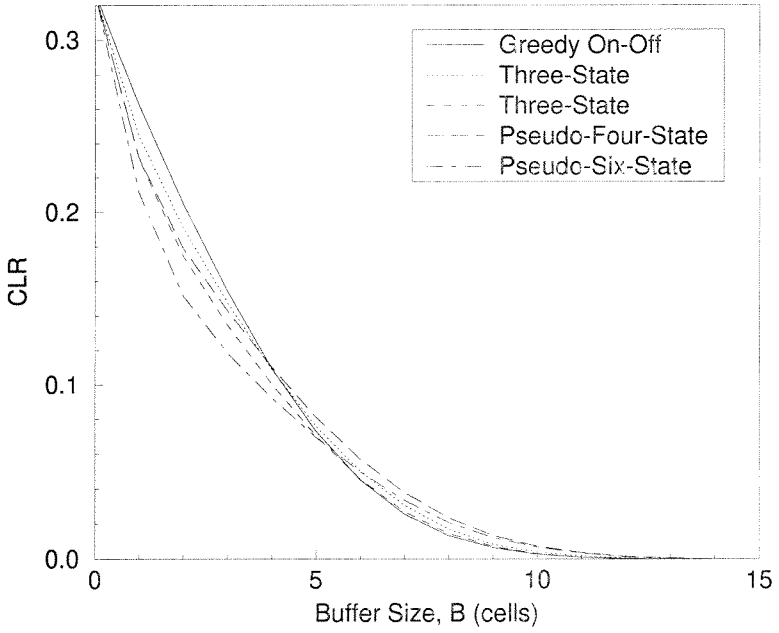



Figure 4.23: CLR vs Buffer Size For Various Types of Traffic
The first three-state source has $T^* = 5$ and the second $T^* = 10$.

by the lack of overwhelming evidence supporting it. On the other side, the fact that the end-points of the curve are bounded by tight bounds is more than an incentive to think so should the intermediate points be.

One thing is definite, though: For a given set of leaky-bucket parameters, the burstiest traffic doesn't create the highest impact on the overall network and this false belief should be watched out for. For a given situation, the parameters should be chosen taken into consideration the free memory and also the characteristics of the incoming traffic of the past connections.

# Chapter 5 Conclusions and Future Work

The results on call routing algorithms are quite interesting, but due to the direction the industry seems to be heading in the last two years, their immediate value in ATM switching applications is questionable. Especially after the success of the Fore ATM switches, the trend turned towards bus-architecture switches rather than traditional space-division architectures. Most probably, due to their parallelism and convenient and simple routing, space-division will make a come-back sometime and for this routing algorithms is one topic that should be looked into further. The results show that fixed priority and maximum switch-utilization perform the best among the algorithms considered. However, the question always remains whether there are any other that are not necessarily simpler than fixed priority but provide a lower blocking rate on the call level.

On the other hand, the cell loss problem is a current important issue. Most of the time, the average-case and random arrivals from bursty sources situations are investigated [19], [20]. However, the worst-case situation investigated here is something that can happen quite easily, mostly due to its simplicity for the user. Even though no one worst source type was found in this study, it still doesn't rule out that possibility that it might exist. Therefore, it is important to either find it or to disprove its existence. In the meantime, the pseudo-n-state sources should provide enough of a guideline for worst-case traffic.

# Bibliography

[1] P. W. Shor "The Average-Case Analysis Of Some On-Line Algorithms For Bin Packing," Combinatorica, 6 (2), pp. 179-200, 1986.

[2] J. L. Bentley, D. S. Johnson, F. T. Leighton, C. C. McGeoch, L. A. McGeoch "Some Unexpected Expected Behaviour Results For Bin Packing," Proc. 16th Ann. ACM Symp. on Theory of Computing, pp. 279-288, 1984.

[3] E. G. Coffman, C. Courcoubetis, M. R. Garey, D. S. Johnson, L. A. McGeoch, P. W. Shor, R. Weber, M. Yannakakis "Fundamental Discrepancies Between Average-Case Analyses Under Discrete And Continuous Distributions: A Bin Packing Case Study," Proc. 23rd Ann. ACM Symp. on Theory of Computing, pp. 230-241, 1991.

[4] J. Csirik, D. S. Johnson "Bounded Space On-Line Bin Packing: Best Is Better Than First," Proc. 2nd Ann. ACM-SIAM Symp. on Discrete Algorithms, pp. 309-319, 1991.

[5] C. Y. Lee, "Analysis Of Switching Networks," Bell Syst. Tech. J., vol. 34, pp. 1287-1315, 1955.

[6] V. E. Benes, "On Some Proposed Models For Traffic In Connecting Networks," Bell Syst. Tech. J., vol. 46, pp. 105-116, 1967.

[7] C. Clos, "A Study of Nonblocking Switching Networks," Bell Syst. Tech. J., vol. 32, pp. 406-424, 1953.

[8] D. G. Cantor, "On Nonblocking Switching Networks," Networks, vol. 1, pp. 367-377, 1971.

[9] N. J. Pippenger, "On Crossbar Switching Networks," IEEE Trans. Commun., vol. COMM-23, no. 6, pp. 646-659, 1975.

[10] R. Melen and J. S. Turner, "Nonblocking Networks For Fast Packet Switching," Proc. of IEEE Infocom 89, pp. 548-557, 1989.

[11] R. Melen and J. S. Turner, "Nonblocking Multirate Distribution Networks," IEEE Trans. Commun., vol. 41, no. 2, pp. 362-369, 1993.

[12] E. Valdimarsson, "Blocking In Multirate Interconnection Networks," IEEE Trans. Commun., vol. 42, no. 2-4, part 3, pp. 2028-2035, 1994.

[13] E. W. Zegura, "Evaluating Blocking Probability In Generalized Connectors," IEEE-ACM Trans. Networking, vol. 3, no. 4, pp. 387-398, 1995.

[14] E. W. Zegura, "Evaluating Blocking Probability In Distributors," Proc. of IEEE Infocom 93, pp. 1107-1116, 1993.

[15] M. Gerla, T-Y. C. Tai and G. Gallassi, "Internetting LANs And MANs To B-ISDNs For Connectionless Traffic Support," IEEE J. on Sel. Areas In Commun., vol. 11, no. 8, pp. 1145-1159, 1993.

[16] L. Kleinrock, *Queueing Systems, Vol. 1: Theory.* New York: Wiley, 1976.

[17] L. Kleinrock, *Queueing Systems, Vol. 2: Computer Applications.* New York: Wiley, 1976.

[18] R. O. Onvural, *Asynchronous Transfer Mode Networks: Performance Issues.* Artech House, 1993.

[19] Z. Yu, R. J. McEliece, "An Analysis on Statistical Multiplexing with On-Off Traffics in ATM Networks," submitted to MMT '96, Paris, France, May 1996.

[20] H. W. Lee, J. W. Mark, "ATM Network Traffic Characterization Using Two Types of On-Off Sources," CCNG Report E-216, University of Waterloo, April 1992

[21] D. Hughes, K. Hooshmand, "ABR Stretches ATM Network Resources," Data Communications, pp. 123-128, April 1995

[22] *Asynchronous Transfer Mode: Bandwidth For The Future.* Telco Systems, 1992.

[23] *The Road To ATM Networking.* Technology White Paper, Synoptics, 1994.

[24] A. Alles, "ATM Internetworking," Technology White Paper, Cisco Systems, May 1995.

[25] M. D. Prycker, *Asynchronous Transfer Mode; Solution For Broadband ISDN.* Prentice Hall, 1995.

[26] M. Karol, M. G. Hluchyj and S. P. Morgan, "Input Vs. Output Queueing On A Space-Division Packet Switch," IEEE Trans. Commun, vol. 35, no. 12, pp. 1 347-1356, 1987.

[27] T. Meisling, "Discrete-Time Queueing Theory," Oper. Res., vol. 6, pp. 96-105, 1958.

[28] S. Akhtar, "Congestion Control In A Fast Packet Switching Network," Master's thesis, Washington University, 1987.

[29] *ATM Forum*, "ATM User-Network Interface Specification," Prentice Hall, 1993.

[30] B. T. Doshi, "Deterministic Rule Based Traffic Descriptors For Broadband ISDN: Worst Case Behaviour And Connection Acceptance Control," International Teletraffic Congress, ITC-14, Vol 1a, pp. 591-600, Antibes, France, June 1994.

[31] T. Worster, "Modelling Deterministic Queues: The Leaky Bucket as an Arrival Process," International Teletraffic Congress, ITC-14, Vol 1a, pp. 581-590, Antibes, France, June 1994.

[32] N. Yamanaka, Y. Sato and K. Sato, "Performance Limitation of Leaky Bucket Algorithm for Usage Parameter Control and Bandwidth Allocation Methods," IEICE Trans. Commun., Vol E75-B, No. 2, February 1992.

[33] J. Y. Hui, *Switching and Traffic Theory For Integrated Broadband Networks*, Kluwer Academic Publishers, 1990.

[34] J. Murphy, J. Murphy and B. Erimli, "On Worst Case Traffic In ATM Networks," IEE 12th UK Teletraffic Symp., pp. 15.1-15.12, London, England, March 1995.

# Appendix A Tables of Cell Loss Rates

| B | Cell Loss Rate | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Greedy On-Off | Three-State | | | | | | |
| | | $T^* = 2$ | $T^* = 4$ | $T^* = 6$ | $T^* = 8$ | $T^* = 10$ | $T^* = 12$ | $T^* = 20$ |
| 0 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.217778 | 0.216797 | 0.214533 | 0.211420 | 0.207756 | 0.203750 | 0.199546 | 0.1824 |
| 2 | 0.187778 | 0.188477 | 0.185986 | 0.182870 | 0.179363 | 0.175625 | 0.171769 | 0.1564 |
| 3 | 0.160000 | 0.162109 | 0.160900 | 0.157407 | 0.153740 | 0.150000 | 0.146259 | 0.132 |
| 4 | 0.134444 | 0.137695 | 0.137543 | 0.135031 | 0.130886 | 0.126875 | 0.123016 | 0.1092 |
| 5 | 0.111111 | 0.115234 | 0.115917 | 0.114198 | 0.110803 | 0.106250 | 0.102041 | 0.088 |
| 6 | 0.090000 | 0.094727 | 0.096021 | 0.094907 | 0.092105 | 0.088125 | 0.083333 | 0.0684 |
| 7 | 0.071111 | 0.076172 | 0.077855 | 0.077160 | 0.074792 | 0.071250 | 0.066893 | 0.0504 |
| 8 | 0.054444 | 0.059570 | 0.061419 | 0.060957 | 0.058864 | 0.055625 | 0.051587 | 0.0364 |
| 9 | 0.040000 | 0.044922 | 0.046713 | 0.046296 | 0.044321 | 0.041250 | 0.037415 | 0.0264 |
| 10 | 0.027778 | 0.032227 | 0.033737 | 0.033179 | 0.031163 | 0.028125 | 0.025510 | 0.018 |
| 11 | 0.017778 | 0.021484 | 0.022491 | 0.021605 | 0.019391 | 0.017500 | 0.015873 | 0.0112 |
| 12 | 0.010000 | 0.012695 | 0.012976 | 0.011574 | 0.010388 | 0.009375 | 0.008503 | 0.006 |
| 13 | 0.004444 | 0.005859 | 0.005190 | 0.004630 | 0.004155 | 0.003750 | 0.003401 | 0.0024 |
| 14 | 0.001111 | 0.000977 | 0.000865 | 0.000772 | 0.000693 | 0.000625 | 0.000567 | 0.0004 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table A.1: Cell Loss Rates $N = 2$, $M = 8$, $R = 0.5$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | | | | |
|---|---|---|---|---|---|---|---|
| | Greedy On-Off | Three-State | | | | | |
| | | $T^* = 2$ | $T^* = 4$ | $T^* = 6$ | $T^* = 8$ | $T^* = 10$ | $T^* = 20$ |
| 0 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.225625 | 0.225057 | 0.223657 | 0.221645 | 0.219184 | 0.216400 | 0.200278 |
| 2 | 0.202500 | 0.202948 | 0.201446 | 0.199433 | 0.197049 | 0.194400 | 0.179444 |
| 3 | 0.180625 | 0.181973 | 0.181302 | 0.179112 | 0.176649 | 0.174000 | 0.159722 |
| 4 | 0.160000 | 0.162132 | 0.162190 | 0.160681 | 0.157986 | 0.155200 | 0.141111 |
| 5 | 0.140625 | 0.143424 | 0.144112 | 0.143195 | 0.141059 | 0.138000 | 0.123611 |
| 6 | 0.122500 | 0.125850 | 0.127066 | 0.126654 | 0.125000 | 0.122400 | 0.107222 |
| 7 | 0.105625 | 0.109410 | 0.111054 | 0.111059 | 0.109809 | 0.107600 | 0.091944 |
| 8 | 0.090000 | 0.094104 | 0.096074 | 0.096408 | 0.095486 | 0.093600 | 0.077778 |
| 9 | 0.075625 | 0.079932 | 0.082128 | 0.082703 | 0.082031 | 0.080400 | 0.064722 |
| 10 | 0.062500 | 0.066893 | 0.069215 | 0.069943 | 0.069444 | 0.068000 | 0.052778 |
| 11 | 0.050625 | 0.054989 | 0.057335 | 0.058129 | 0.057726 | 0.056400 | 0.042500 |
| 12 | 0.040000 | 0.044218 | 0.046488 | 0.047259 | 0.046875 | 0.045600 | 0.033333 |
| 13 | 0.030625 | 0.034580 | 0.036674 | 0.037335 | 0.036892 | 0.035600 | 0.025278 |
| 14 | 0.022500 | 0.026077 | 0.027893 | 0.028355 | 0.027778 | 0.026400 | 0.018333 |
| 15 | 0.015625 | 0.018707 | 0.020145 | 0.020321 | 0.019531 | 0.018000 | 0.012500 |
| 16 | 0.010000 | 0.012472 | 0.013430 | 0.013233 | 0.012153 | 0.011200 | 0.007778 |
| 17 | 0.005625 | 0.007370 | 0.007748 | 0.007089 | 0.006510 | 0.006000 | 0.004167 |
| 18 | 0.002500 | 0.003401 | 0.003099 | 0.002836 | 0.002604 | 0.002400 | 0.001667 |
| 19 | 0.000625 | 0.000567 | 0.000517 | 0.000473 | 0.000434 | 0.000400 | 0.000278 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table A.2: Cell Loss Rates $N = 2$, $M = 10.5$, $R = 0.5$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | | | | |
|---|---|---|---|---|---|---|---|
| | Greedy | Three-State | | | | | |
| | On-Off | $T^* = 3$ | $T^* = 6$ | $T^* = 9$ | $T^* = 12$ | $T^* = 15$ | $T^* = 18$ |
| 0 | 0.296296 | 0.296296 | 0.296296 | 0.296296 | 0.296296 | 0.296296 | 0.296296 |
| 1 | 0.263037 | 0.260762 | 0.255273 | 0.249144 | 0.242738 | 0.236280 | 0.229908 |
| 2 | 0.230222 | 0.230403 | 0.225823 | 0.218581 | 0.211127 | 0.203918 | 0.196994 |
| 3 | 0.198296 | 0.200824 | 0.198217 | 0.192771 | 0.185685 | 0.178030 | 0.170727 |
| 4 | 0.167704 | 0.172246 | 0.171125 | 0.167366 | 0.161862 | 0.155369 | 0.148365 |
| 5 | 0.138889 | 0.145004 | 0.145190 | 0.142568 | 0.138484 | 0.133432 | 0.127658 |
| 6 | 0.112296 | 0.119431 | 0.120542 | 0.118782 | 0.115916 | 0.112154 | 0.107603 |
| 7 | 0.088370 | 0.095862 | 0.097822 | 0.097018 | 0.094887 | 0.091995 | 0.088415 |
| 8 | 0.067556 | 0.074631 | 0.077418 | 0.077479 | 0.075640 | 0.073284 | 0.070421 |
| 9 | 0.050296 | 0.057239 | 0.060228 | 0.060470 | 0.058984 | 0.056615 | 0.054000 |
| 10 | 0.037037 | 0.043521 | 0.046125 | 0.046292 | 0.044758 | 0.042184 | 0.039424 |
| 11 | 0.027000 | 0.032724 | 0.034915 | 0.034694 | 0.033001 | 0.030288 | 0.027235 |
| 12 | 0.018963 | 0.023931 | 0.025763 | 0.025321 | 0.023351 | 0.020565 | 0.017650 |
| 13 | 0.012704 | 0.016807 | 0.018283 | 0.017667 | 0.015644 | 0.013213 | 0.010968 |
| 14 | 0.008000 | 0.011186 | 0.012217 | 0.011430 | 0.009718 | 0.007967 | 0.006565 |
| 15 | 0.004630 | 0.006901 | 0.007437 | 0.006558 | 0.005331 | 0.004335 | 0.003572 |
| 16 | 0.002370 | 0.003784 | 0.003815 | 0.003102 | 0.002484 | 0.002019 | 0.001664 |
| 17 | 0.001000 | 0.001670 | 0.001415 | 0.001113 | 0.000891 | 0.000724 | 0.000597 |
| 18 | 0.000296 | 0.000390 | 0.000300 | 0.000236 | 0.000189 | 0.000154 | 0.000127 |
| 19 | 0.000037 | 0.000028 | 0.000021 | 0.000017 | 0.000013 | 0.000011 | 0.000009 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table A.3: Cell Loss Rates $N = 3$, $M = 7$, $R = 1/3$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | |
|---|---|---|---|
| | Greedy On-Off | Three-State | |
| | | $T^* = 4$ | $T^* = 8$ |
| 0 | 0.316406 | 0.316406 | 0.316406 |
| 1 | 0.259525 | 0.248300 | 0.235703 |
| 2 | 0.206806 | 0.200346 | 0.185862 |
| 3 | 0.158025 | 0.157986 | 0.147653 |
| 4 | 0.114306 | 0.119385 | 0.112871 |
| 5 | 0.078125 | 0.085503 | 0.081301 |
| 6 | 0.050606 | 0.057825 | 0.054875 |
| 7 | 0.030325 | 0.036241 | 0.034901 |
| 8 | 0.016606 | 0.021114 | 0.020559 |
| 9 | 0.008325 | 0.011683 | 0.010835 |
| 10 | 0.003906 | 0.005923 | 0.004850 |
| 11 | 0.001600 | 0.002520 | 0.001679 |
| 12 | 0.000506 | 0.000678 | 0.000386 |
| 13 | 0.000100 | 0.000084 | 0.000046 |
| 14 | 0.000006 | 0.000003 | 0.000002 |
| 15 | 0 | 0 | 0 |

Table A.4: Cell Loss Rates $N = 4$, $M = 4$, $R = 0.25$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | |
|---|---|---|---|
| | Greedy On-Off | Three-State $T^* = 5$ | Three-State $T^* = 10$ |
| 0 | 0.327680 | 0.327680 | 0.327680 |
| 1 | 0.262494 | 0.244496 | 0.231236 |
| 2 | 0.205320 | 0.191264 | 0.174728 |
| 3 | 0.154321 | 0.147660 | 0.134573 |
| 4 | 0.109760 | 0.109123 | 0.100621 |
| 5 | 0.073221 | 0.076007 | 0.070674 |
| 6 | 0.045100 | 0.050042 | 0.045554 |
| 7 | 0.025567 | 0.030528 | 0.026840 |
| 8 | 0.013440 | 0.016864 | 0.014623 |
| 9 | 0.006391 | 0.008413 | 0.007090 |
| 10 | 0.002680 | 0.003888 | 0.002852 |
| 11 | 0.000994 | 0.001564 | 0.000900 |
| 12 | 0.000320 | 0.000447 | 0.000206 |
| 13 | 0.000076 | 0.000073 | 0.000030 |
| 14 | 0.000010 | 0.000005 | 0.000002 |
| 15 | 0 | 0 | 0 |

Table A.5: Cell Loss Rates $N = 5$, $M = 3.4$, $R = 0.2$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | | | |
|---|---|---|---|---|---|---|
| | Greedy | Three-State | | | | |
| | On-Off | $T^* = 2$ | $T^* = 4$ | $T^* = 6$ | $T^* = 8$ | $T^* = 10$ |
| 0 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.225625 | 0.225057 | 0.223657 | 0.221645 | 0.219184 | 0.216400 |
| 2 | 0.202500 | 0.202948 | 0.201446 | 0.199433 | 0.197049 | 0.194400 |
| 3 | 0.180625 | 0.181973 | 0.181302 | 0.179112 | 0.176649 | 0.174000 |
| 4 | 0.160000 | 0.162132 | 0.162190 | 0.160681 | 0.157986 | 0.155200 |
| 5 | 0.140625 | 0.143424 | 0.144112 | 0.143195 | 0.141059 | 0.138000 |
| 6 | 0.122500 | 0.125850 | 0.127066 | 0.126654 | 0.125000 | 0.122400 |
| 7 | 0.105625 | 0.109410 | 0.111054 | 0.111059 | 0.109809 | 0.107600 |
| 8 | 0.090000 | 0.094104 | 0.096074 | 0.096408 | 0.095486 | 0.093600 |
| 9 | 0.075625 | 0.079932 | 0.082128 | 0.082703 | 0.082031 | 0.080400 |
| 10 | 0.062500 | 0.066893 | 0.069215 | 0.069943 | 0.069444 | 0.068000 |
| 11 | 0.050625 | 0.054989 | 0.057335 | 0.058129 | 0.057726 | 0.056400 |
| 12 | 0.040000 | 0.044218 | 0.046488 | 0.047259 | 0.046875 | 0.045600 |
| 13 | 0.030625 | 0.034580 | 0.036674 | 0.037335 | 0.036892 | 0.035600 |
| 14 | 0.022500 | 0.026077 | 0.027893 | 0.028355 | 0.027778 | 0.026400 |
| 15 | 0.015625 | 0.018707 | 0.020145 | 0.020321 | 0.019531 | 0.018000 |
| 16 | 0.010000 | 0.012472 | 0.013430 | 0.013233 | 0.012153 | 0.011200 |
| 17 | 0.005625 | 0.007370 | 0.007748 | 0.007089 | 0.006510 | 0.006000 |
| 18 | 0.002500 | 0.003401 | 0.003099 | 0.002836 | 0.002604 | 0.002400 |
| 19 | 0.000625 | 0.000567 | 0.000517 | 0.000473 | 0.000434 | 0.000400 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 |

Table A.6: Cell Loss Rates $N = 2$, $M = 10.5$, $R = 0.5$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | | | |
|---|---|---|---|---|---|---|
| | Greedy | Pseudo-Four-State | | | | |
| | On-Off | $T^* = 2$ | $T^* = 4$ | $T^* = 6$ | $T^* = 8$ | $T^* = 10$ |
| 0 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.225625 | 0.224690 | 0.220920 | 0.215607 | 0.209503 | 0.203056 |
| 2 | 0.202500 | 0.203512 | 0.201389 | 0.197485 | 0.192602 | 0.187222 |
| 3 | 0.180625 | 0.183368 | 0.182726 | 0.180104 | 0.176339 | 0.171944 |
| 4 | 0.160000 | 0.164256 | 0.164931 | 0.163462 | 0.160714 | 0.157222 |
| 5 | 0.140625 | 0.146178 | 0.148003 | 0.147559 | 0.145727 | 0.143056 |
| 6 | 0.122500 | 0.129132 | 0.131944 | 0.132396 | 0.131378 | 0.129444 |
| 7 | 0.105625 | 0.113120 | 0.116753 | 0.117973 | 0.117666 | 0.116389 |
| 8 | 0.090000 | 0.098140 | 0.102431 | 0.104290 | 0.104592 | 0.103889 |
| 9 | 0.075625 | 0.084194 | 0.088976 | 0.091346 | 0.092156 | 0.091944 |
| 10 | 0.062500 | 0.071281 | 0.076389 | 0.079142 | 0.080357 | 0.080556 |
| 11 | 0.050625 | 0.059401 | 0.064670 | 0.067678 | 0.069196 | 0.069722 |
| 12 | 0.040000 | 0.048554 | 0.053819 | 0.056953 | 0.058673 | 0.059444 |
| 13 | 0.030625 | 0.038740 | 0.043837 | 0.046967 | 0.048788 | 0.049722 |
| 14 | 0.022500 | 0.029959 | 0.034722 | 0.037722 | 0.039541 | 0.040556 |
| 15 | 0.015625 | 0.022211 | 0.026476 | 0.029216 | 0.030931 | 0.031944 |
| 16 | 0.010000 | 0.015496 | 0.019097 | 0.021450 | 0.022959 | 0.023889 |
| 17 | 0.005625 | 0.009814 | 0.012587 | 0.014423 | 0.015625 | 0.016389 |
| 18 | 0.002500 | 0.005165 | 0.006944 | 0.008136 | 0.008929 | 0.009444 |
| 19 | 0.000625 | 0.001550 | 0.002170 | 0.002589 | 0.002870 | 0.003056 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 |

Table A.7: Cell Loss Rates $N = 2$, $M = 10.5$, $R = 0.5$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | | | |
|---|---|---|---|---|---|---|
| | Greedy | Pseudo-Six-State, $T^{**} = 3$ | | | | |
| | On-Off | $T^* = 2$ | $T^* = 4$ | $T^* = 6$ | $T^* = 8$ | $T^* = 10$ |
| 0 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.225625 | 0.220052 | 0.214867 | 0.208865 | 0.202500 | 0.196045 |
| 2 | 0.202500 | 0.199653 | 0.196006 | 0.191327 | 0.186111 | 0.180664 |
| 3 | 0.180625 | 0.180990 | 0.178624 | 0.175064 | 0.170833 | 0.166260 |
| 4 | 0.160000 | 0.163194 | 0.161982 | 0.159439 | 0.156111 | 0.152344 |
| 5 | 0.140625 | 0.146267 | 0.146080 | 0.144452 | 0.141944 | 0.138916 |
| 6 | 0.122500 | 0.130208 | 0.130917 | 0.130102 | 0.128333 | 0.125977 |
| 7 | 0.105625 | 0.115017 | 0.116494 | 0.116390 | 0.115278 | 0.113525 |
| 8 | 0.090000 | 0.100694 | 0.102811 | 0.103316 | 0.102778 | 0.101562 |
| 9 | 0.075625 | 0.087240 | 0.089867 | 0.090880 | 0.090833 | 0.090088 |
| 10 | 0.062500 | 0.074653 | 0.077663 | 0.079082 | 0.079444 | 0.079102 |
| 11 | 0.050625 | 0.062934 | 0.066198 | 0.067921 | 0.068611 | 0.068604 |
| 12 | 0.040000 | 0.052083 | 0.055473 | 0.057398 | 0.058333 | 0.058594 |
| 13 | 0.030625 | 0.042101 | 0.045488 | 0.047513 | 0.048611 | 0.049072 |
| 14 | 0.022500 | 0.032986 | 0.036243 | 0.038265 | 0.039444 | 0.040039 |
| 15 | 0.015625 | 0.024740 | 0.027737 | 0.029656 | 0.030833 | 0.031494 |
| 16 | 0.010000 | 0.017361 | 0.019970 | 0.021684 | 0.022778 | 0.023438 |
| 17 | 0.005625 | 0.010851 | 0.012944 | 0.014349 | 0.015278 | 0.015869 |
| 18 | 0.002500 | 0.005208 | 0.006657 | 0.007653 | 0.008333 | 0.008789 |
| 19 | 0.000625 | 0.001302 | 0.001849 | 0.002232 | 0.002500 | 0.002686 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 |

Table A.8: Cell Loss Rates $N = 2$, $M = 10.5$, $R = 0.5$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | | | |
|---|---|---|---|---|---|---|
| | Greedy | Pseudo-Six-State, $T^{**} = 6$ | | | | |
| | On-Off | $T^* = 2$ | $T^* = 4$ | $T^* = 6$ | $T^* = 8$ | $T^* = 10$ |
| 0 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.225625 | 0.214127 | 0.208227 | 0.201944 | 0.195557 | 0.189230 |
| 2 | 0.202500 | 0.193787 | 0.189413 | 0.184444 | 0.179199 | 0.173875 |
| 3 | 0.180625 | 0.175666 | 0.172513 | 0.168611 | 0.164307 | 0.159818 |
| 4 | 0.160000 | 0.159024 | 0.156888 | 0.153889 | 0.150391 | 0.146626 |
| 5 | 0.140625 | 0.143121 | 0.141901 | 0.139722 | 0.136963 | 0.133867 |
| 6 | 0.122500 | 0.127959 | 0.127551 | 0.126111 | 0.124023 | 0.121540 |
| 7 | 0.105625 | 0.113536 | 0.113839 | 0.113056 | 0.111572 | 0.109645 |
| 8 | 0.090000 | 0.099852 | 0.100765 | 0.100556 | 0.099609 | 0.098183 |
| 9 | 0.075625 | 0.086908 | 0.088329 | 0.088611 | 0.088135 | 0.087154 |
| 10 | 0.062500 | 0.074704 | 0.076531 | 0.077222 | 0.077148 | 0.076557 |
| 11 | 0.050625 | 0.063240 | 0.065370 | 0.066389 | 0.066650 | 0.066393 |
| 12 | 0.040000 | 0.052515 | 0.054847 | 0.056111 | 0.056641 | 0.056661 |
| 13 | 0.030625 | 0.042530 | 0.044962 | 0.046389 | 0.047119 | 0.047362 |
| 14 | 0.022500 | 0.033284 | 0.035714 | 0.037222 | 0.038086 | 0.038495 |
| 15 | 0.015625 | 0.024778 | 0.027105 | 0.028611 | 0.029541 | 0.030061 |
| 16 | 0.010000 | 0.017012 | 0.019133 | 0.020556 | 0.021484 | 0.022059 |
| 17 | 0.005625 | 0.009985 | 0.011798 | 0.013056 | 0.013916 | 0.014490 |
| 18 | 0.002500 | 0.004438 | 0.005740 | 0.006667 | 0.007324 | 0.007785 |
| 19 | 0.000625 | 0.001109 | 0.001594 | 0.001944 | 0.002197 | 0.002379 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 |

Table A.9: Cell Loss Rates $N = 2$, $M = 10.5$, $R = 0.5$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | |
|---|---|---|---|---|
| | Greedy | Three-State | | Pseudo-Four-State |
| | On-Off | $T^* = 3$ | $T^* = 6$ | $T^* = 3$ |
| 0 | 0.296296 | 0.296296 | 0.296296 | 0.296296 |
| 1 | 0.263037 | 0.260762 | 0.255273 | 0.256559 |
| 2 | 0.230222 | 0.230403 | 0.225823 | 0.228395 |
| 3 | 0.198296 | 0.200824 | 0.198217 | 0.201175 |
| 4 | 0.167704 | 0.172246 | 0.171125 | 0.174726 |
| 5 | 0.138889 | 0.145004 | 0.145190 | 0.149306 |
| 6 | 0.112296 | 0.119431 | 0.120542 | 0.125171 |
| 7 | 0.088370 | 0.095862 | 0.097822 | 0.102581 |
| 8 | 0.067556 | 0.074631 | 0.077418 | 0.081790 |
| 9 | 0.050296 | 0.057239 | 0.060228 | 0.063957 |
| 10 | 0.037037 | 0.043521 | 0.046125 | 0.050240 |
| 11 | 0.027000 | 0.032724 | 0.034915 | 0.039159 |
| 12 | 0.018963 | 0.023931 | 0.025763 | 0.029750 |
| 13 | 0.012704 | 0.016807 | 0.018283 | 0.021884 |
| 14 | 0.008000 | 0.011186 | 0.012217 | 0.015432 |
| 15 | 0.004630 | 0.006901 | 0.007437 | 0.010267 |
| 16 | 0.002370 | 0.003784 | 0.003815 | 0.006259 |
| 17 | 0.001000 | 0.001670 | 0.001415 | 0.003279 |
| 18 | 0.000296 | 0.000390 | 0.000300 | 0.001200 |
| 19 | 0.000037 | 0.000028 | 0.000021 | 0.000150 |
| 20 | 0 | 0 | 0 | 0 |

Table A.10: Cell Loss Rates $N = 3$, $M = 7$, $R = 1/3$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | |
|---|---|---|---|---|
| | Greedy | Three-State | | Pseudo-Four-State |
| | On-Off | $T^* = 4$ | $T^* = 8$ | $T^* = 4$ |
| 0 | 0.316406 | 0.316406 | 0.316406 | 0.316406 |
| 1 | 0.259525 | 0.248300 | 0.235703 | 0.237694 |
| 2 | 0.206806 | 0.200346 | 0.185862 | 0.193115 |
| 3 | 0.158025 | 0.157986 | 0.147653 | 0.156595 |
| 4 | 0.114306 | 0.119385 | 0.112871 | 0.122477 |
| 5 | 0.078125 | 0.085503 | 0.081301 | 0.092117 |
| 6 | 0.050606 | 0.057825 | 0.054875 | 0.066198 |
| 7 | 0.030325 | 0.036241 | 0.034901 | 0.044701 |
| 8 | 0.016606 | 0.021114 | 0.020559 | 0.028173 |
| 9 | 0.008325 | 0.011683 | 0.010835 | 0.017161 |
| 10 | 0.003906 | 0.005923 | 0.004850 | 0.010043 |
| 11 | 0.001600 | 0.002520 | 0.001679 | 0.005193 |
| 12 | 0.000506 | 0.000678 | 0.000386 | 0.002026 |
| 13 | 0.000100 | 0.000084 | 0.000046 | 0.000443 |
| 14 | 0.000006 | 0.000003 | 0.000002 | 0.000024 |
| 15 | 0 | 0 | 0 | 0 |

Table A.11: Cell Loss Rates $N = 4$, $M = 4$, $R = 0.25$, $p = 1$, $r = 1$

| B | Cell Loss Rate | | | | |
|---|---|---|---|---|---|
| | Greedy On-Off | Three-State $T^* = 5$ | Three-State $T^* = 10$ | Pseudo-Four-State $T^* = 5$ | Pseudo-Six-State $T^* = 5$, $T^{**} = 6$ |
| 0 | 0.327680 | 0.327680 | 0.327680 | 0.327680 | 0.327680 |
| 1 | 0.262494 | 0.244496 | 0.231236 | 0.231812 | 0.211462 |
| 2 | 0.205320 | 0.191264 | 0.174728 | 0.179286 | 0.151630 |
| 3 | 0.154321 | 0.147660 | 0.134573 | 0.142798 | 0.118274 |
| 4 | 0.109760 | 0.109123 | 0.100621 | 0.110477 | 0.092574 |
| 5 | 0.073221 | 0.076007 | 0.070674 | 0.081311 | 0.069604 |
| 6 | 0.045100 | 0.050042 | 0.045554 | 0.056928 | 0.049846 |
| 7 | 0.025567 | 0.030528 | 0.026840 | 0.037837 | 0.033491 |
| 8 | 0.013440 | 0.016864 | 0.014623 | 0.023459 | 0.020796 |
| 9 | 0.006391 | 0.008413 | 0.007090 | 0.013391 | 0.012089 |
| 10 | 0.002680 | 0.003888 | 0.002852 | 0.007099 | 0.006734 |
| 11 | 0.000994 | 0.001564 | 0.000900 | 0.003491 | 0.003267 |
| 12 | 0.000320 | 0.000447 | 0.000206 | 0.001392 | 0.001181 |
| 13 | 0.000076 | 0.000073 | 0.000030 | 0.000377 | 0.000228 |
| 14 | 0.000010 | 0.000005 | 0.000002 | 0.000047 | 0.000017 |
| 15 | 0.000000 | 0.000000 | 0.000000 | 0.000001 | 0.000000 |
| 16 | 0 | 0 | 0 | 0 | 0 |

Table A.12: Cell Loss Rates $N = 5$, $M = 3.4$, $R = 0.2$, $p = 1$, $r = 1$

# Appendix B Graphs of Blocking Probabilities

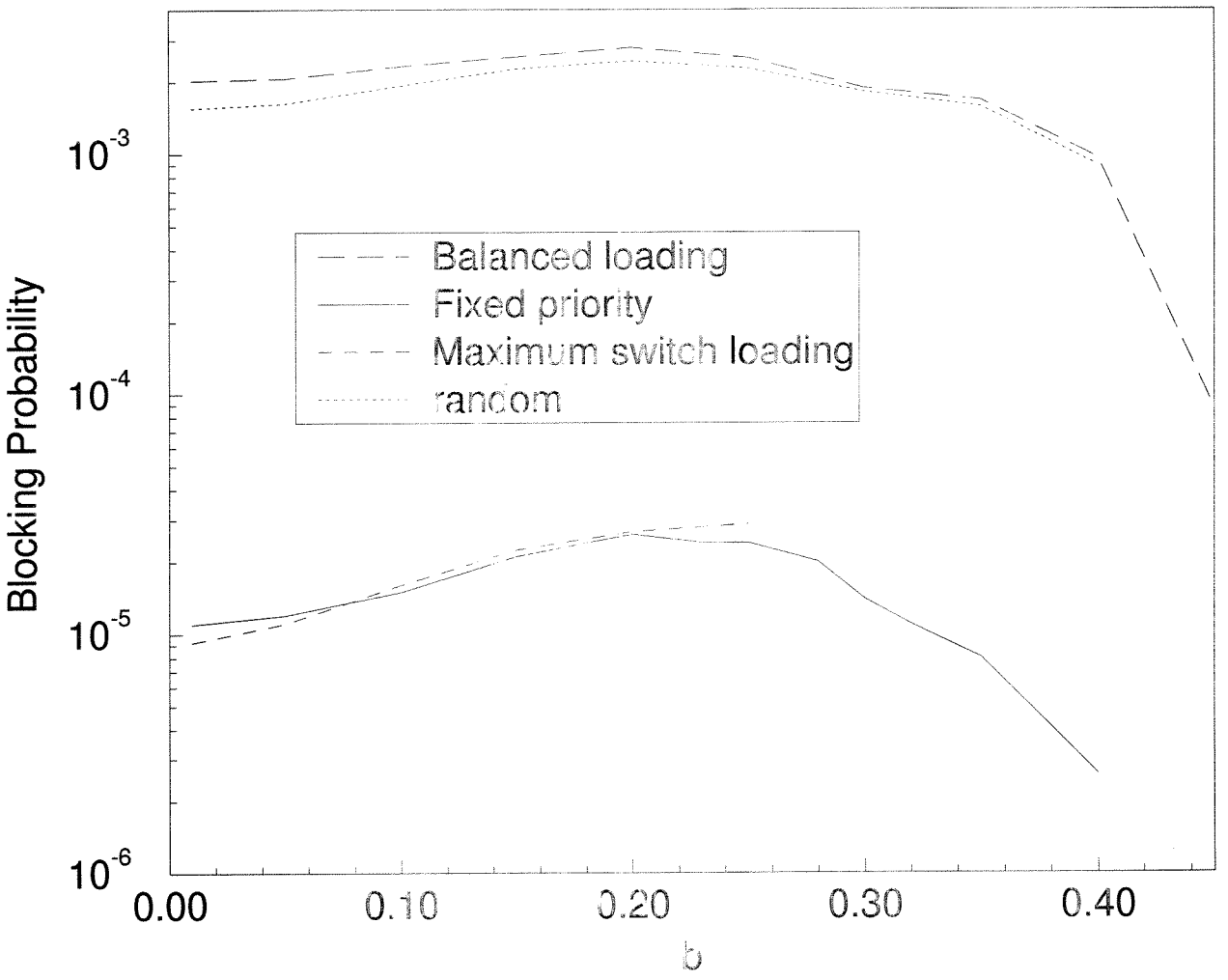



Figure B.1: Blocking Probability Varying With Respect to $b$. $B = 0.99$, $\beta = 0.99$

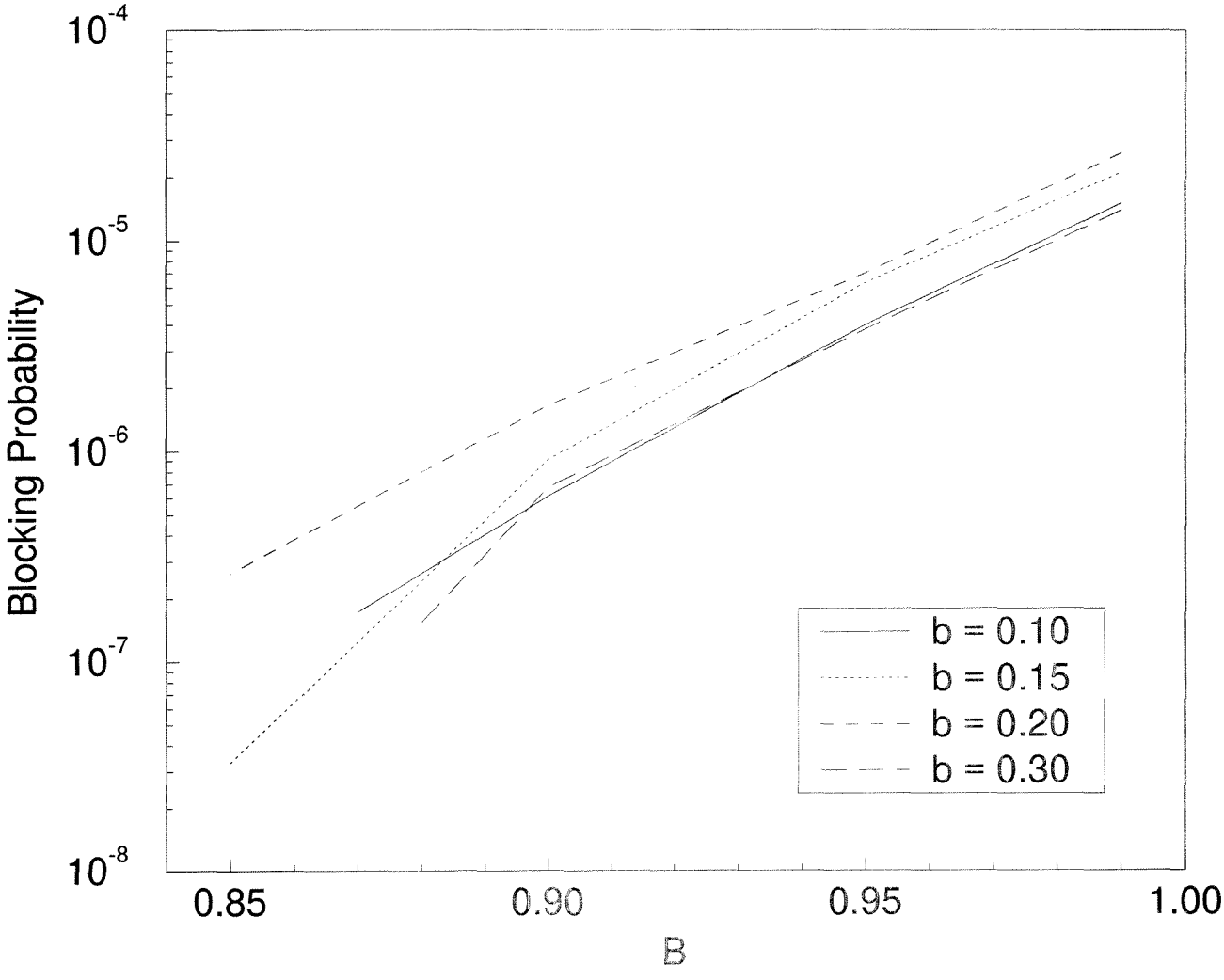



Figure B.2: Blocking Probability Varying With Respect to $B$ For Different Values of $b$. $\beta = 0.99$

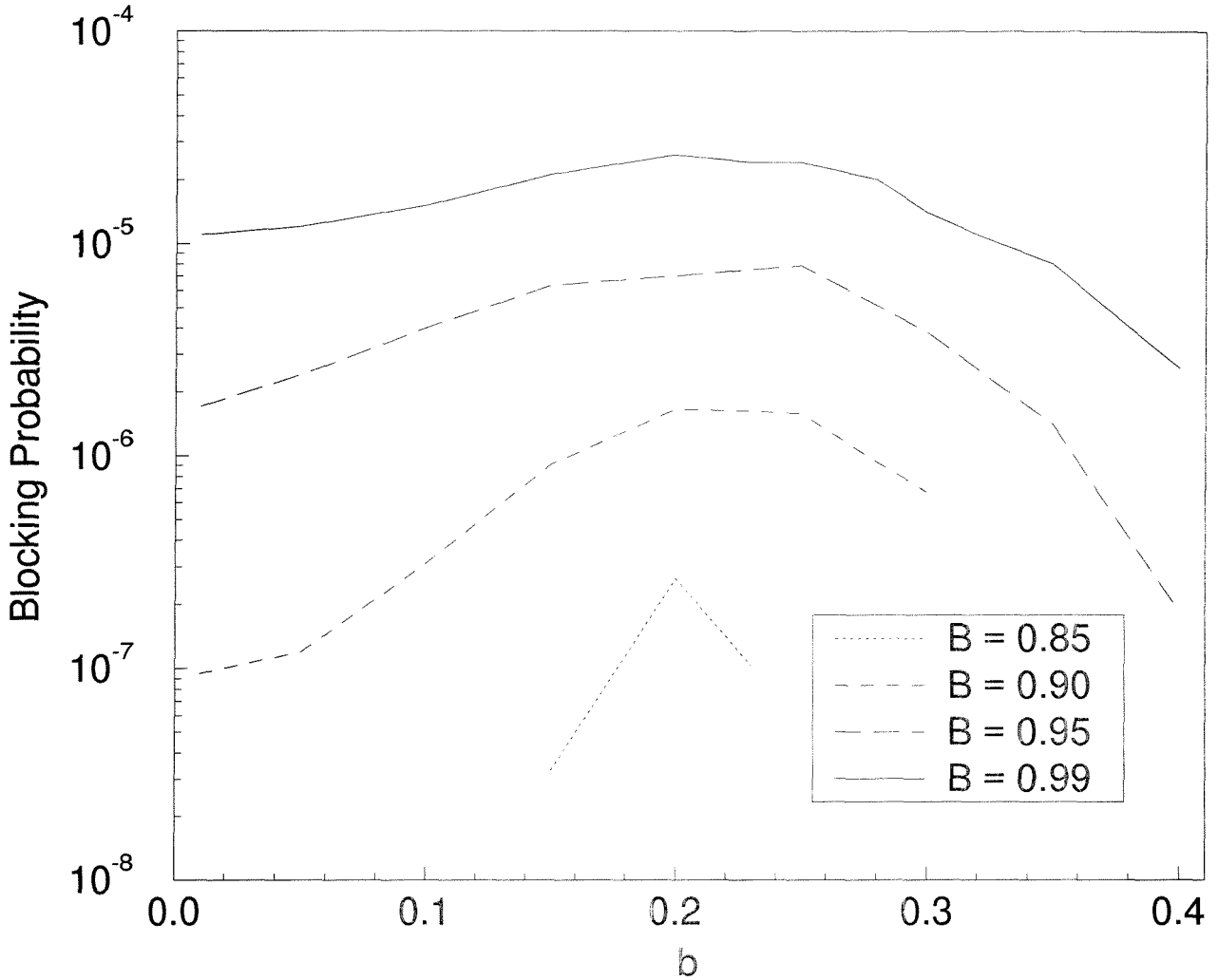



Figure B.3: Blocking Probability Varying With Respect to $b$ For Different Values of $B$. $\beta = 0.99$