PERFORMANCE OF TWO - LEVEL FORWARD ERROR CORRECTION FOR LOST CELL RECOVERY IN ATM NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

BY
Nihat Cem Oğuz
July 1995

PERFORMANCE OF TWO-LEVEL FORWARD ERROR CORRECTION FOR LOST CELL RECOVERY IN ATM NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Nihat Cem Oğuz
July 1995

Nihat Cem Oğuz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Ender Ayanoğlu, Ph. D. (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Selim Aktürk, Ph. D.

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Erdal Arıkan, Ph. D.

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Erol Sezer, Ph. D.

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Yalçın Tanık, Ph. D.

Approved for the Institute of Engineering and Science:

Mehmet Baray, Ph. D.
Director of Institute of Engineering and Science

To Bengi, Ali Atıl, and my parents

Abstract

PERFORMANCE OF TWO-LEVEL FORWARD ERROR CORRECTION FOR LOST CELL RECOVERY IN ATM NETWORKS

Nihat Cem Oğuz
Ph. D. in Electrical and Electronics Engineering
Supervisor:
Assoc. Prof. Dr. Ender Ayanoğlu
July 1995

The major source of errors in *Asynchronous Transfer Mode* (ATM) networks is expected to be buffer overflow during congestion, resulting in cell losses. The large ratio of the end-to-end propagation time for a typical connection to the cell transmission time makes lost cell recovery by means of retransmission-based error control techniques impractical especially for delay-sensitive high-speed applications. As has been shown by many authors, forward error correction is a promising alternative since it can improve end-to-end reliability without requiring retransmissions. This thesis discusses the use of a *two-level* forward error correction scheme for virtual channel and virtual path connections in ATM networks. The performance of the scheme, which exploits erasure correcting simple and interleaved block codes simultaneously, is studied via both analyses and simulations. For a single-node virtual channel connection, a novel and accurate discrete-time analytical cell loss model is developed first. Based on this model, the reduction in the cell loss rate achieved by two-level coding is then investigated extensively via iterative computational methods. For the case

of a four-node, long-distance virtual channel connection that cannot tolerate any loss, the use of the two-level coding scheme in conjunction with an automatic repeat request mechanism is considered, and detailed simulations are made to quantify the improvement achieved in the delay-throughput performance. The results obtained indicate substantial performance improvements even for very high network loads provided that an appropriate coding technique is chosen according to the traffic characteristics. Typically, bursty traffic requires code interleaving be used for effective loss recovery whereas small-latency simple block codes suffice for random traffic. Two-level coding, which is shown to effectively combine the fast and burst loss recovery capabilities of the individual coding techniques, is attractive for traffic streams of unpredictable or time-varying characteristics.

Keywords: Asynchronous transfer mode (ATM), forward error correction (FEC), automatic repeat request (ARQ), lost cell recovery, cell loss process characterization.

Özet

ATM AĞLARINDA YİTİK GÖZE KURTARIMI İÇİN İKİ SEVİYELİ İLERİ HATA DÜZELTİMİNİN BAŞARIMI

Nihat Cem Oğuz
Elektrik ve Elektronik Mühendisliği'nde Doktora
Tez Yöneticisi:
Doç. Dr. Ender Ayanoğlu
Temmuz 1995

Eşzamansız Aktarım Modu (ATM) ağlarında ana hata kaynağının, göze yitimine yol açan, sıkışmaya bağlı yastık taşımı olması beklenir. Göze gönderim zamanının tipik bir bağlantı için uçtan uca iletim gecikmesine oranla çok küçük olması nedeniyle, tekrar gönderime dayanan hata düzeltme teknikleri özellikle gecikmeye duyarlı yüksek hızlı uygulamalar için uygun değildir. Pek çok araştırmacı tarafından gösterildiği gibi, ileri hata düzeltimi, tekrar gönderime gerek duymadan uçtan uca güvenilirliği artırabilmesi nedeniyle, daha iyi bir seçenektir. Bu tez çalışmasında, *iki seviyeli* bir ileri hata düzeltim yönteminin ATM ağlarındaki sanal kanal ve sanal yol bağlantıları için kullanımı tartışılmıştır. Basit ve biniştirilmiş blok kodların eşanlı kullanımına dayanan yöntemin başarımı, hem çözümleme hem de benzetim yoluyla araştırılmıştır. Tek düğümlü bir sanal kanal bağlantısı için, önce yeni ve doğru bir ayrık zaman göze yitim çözümleme modeli geliştirilmiştir. Ardından, bu model üzerinde dürümsel hesaplama yöntemleri kullanılarak, iki seviyeli kodlama yoluyla göze yitim olasılığında elde edilen azalma geniş biçimde araştırılmıştır. Göze yitimini kaldıramayan dört düğümlü bir uzun mesafe sanal kanal bağlantısı içinse, iki seviyeli kodlama yönteminin bir

otomatik tekrar gönderim mekanizması ile birlikte kullanılacağı düşünülmüs, ve bu şekilde uçtan uca toplam gecikmede elde edilen azalmayı ölçmek üzere ayrıntılı benzetimler yapılmıştır. Elde edilen sonuçlar, trafik karakteristiğine uygun bir kodlama tekniği kullanılması kaydıyla, çok yüksek ağ yükleri için bile önemli başarım iyileşmeleri sağlanabileceğini göstermiştir. Tipik olarak, rasgele trafik için az gecikmeli basit blok kodlar yeterli olurken, çoğuşuk trafik için etkin yitik göze kurtarımı, kod biniştirmesini gerektirmektedir. Bu iki kodlama tekniğinin hızlı ve çoğuşuk göze yitim kurtarım özelliklerini etkin biçimde birleştirdiği gösterilen iki seviyeli kodlama yöntemi ise, karakteristiği önceden belirlenemeyen veya zaman içinde değişme gösteren trafik için uygun bulunmuştur.

Anahtar sözcükler: Eşzamansız Aktarım Modu (ATM), ileri hata düzeltimi (FEC), otomatik tekrar gönderim (ARQ), yitik göze kurtarımı, göze yitim süreç karakterizasyonu.

Acknowledgment

I would like to express my deepest gratitude to Dr. Ender Ayanoğlu for his invaluable guidance and suggestions during the development of this work, and also for that he encouraged me to participate in various distinguished international conferences from which I gained experience and motivation.

I am grateful to Dr. Abdullah Atalar, chairman of our department, for his continuous moral support throughout my graduate study.

I would also like to thank to my friends Nail Akar and Ogan Ocalı for their valuable technical support and discussions.

Finally, my sincere thanks are due to my parents and my wife Bengi for their love and patience. Bengi was so understanding even in my busiest moments, and undertook to look after Atıl, our lovely little son, night and day.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today, there is an ever increasing demand for new kinds of telecommunication services from both residential and business subscribers. High definition TV, video conferencing, high-speed data transfer, videophony, video library, home education, and video on demand are among the services that are expected to be in wide use in the near future. Owing to the recent advances in semiconductor and optical technologies, and the related progress in system concepts, a single high-speed communication network, often referred to as *Broadband Integrated Services Digital Network* (B-ISDN), which can support all these services efficiently in a unified manner is now becoming a reality [16], [21]. A switching technique called *Asynchronous Transfer Mode* (ATM) has been selected by ITU-T[1], the International Technological Union–Telecommunications Section, as the basis for B-ISDN [10].

Although ATM is basically a packet switching technique, the B-ISDN/ATM has several features distinguishing it from conventional packet-switched networks. In this chapter, we first describe ATM briefly as a transport discipline for B-ISDN. This thesis contributes to "packet loss recovery" in ATM networks, and we discuss this issue next. Finally, after summarizing the related work in the literature, we motivate the packet loss recovery scheme we consider for ATM networks, and

[1]ITU-T was formerly known as CCITT, the International Consultative Committee for Telegraph and Telephone.

briefly discuss our contributions.

1.1 Asynchronous Transfer Mode

In ATM, the packet size is small and fixed. These packets, called *cells* in the ATM terminology, are composed of 53 octets, 5 octets for the header and 48 octets for the information field. Although using variable-size packets in general yields higher transmission bandwidth efficiency, this choice for ATM has been made based on the following rationale [16], [21]. First, fixed-size packets enable the use of fast, hardware-based switches that can support high-speed services. Second, with small packets, it is in general easier to control end-to-end delay for real-time services. In fact, the impact of packetization delay on the quality of interactive voice services is the particular reason behind choosing an information field as small as 48 octets[2].

An ATM network conforms to a layered protocol reference model [16], [21]. At the bottom of this reference model, there is the physical layer which provides a cell pipe to the second layer, called the ATM layer. The ATM layer is simply responsible for transporting, switching, and queueing cells within the network. The third layer is the *ATM Adaptation Layer* (AAL), and its primary function is to translate the service-specific nature of user data into a cell stream at the source site, and to do the converse at the destination site.

Unlike the case in conventional data networks, there is no link-by-link flow control in ATM because small cells and high data transmission rates limit the processing time per cell at the nodes [16]. Also because optical fibers have very low error rates, there is no link-by-link error control, either. Instead, these

[2]In interactive voice communications, an overall network delay larger than 20-30 ms becomes troublesome, and requires the use of echo cancelers. Considering typical propagation and queueing delays, a packetization delay of about 5 ms is found to avoid echo problem for most voice connections. Since 64 kbps digital voice data is packetized into 32 octets in 5 ms, this suggests the use of a 32-octet information field as proposed by Europeans. On the other hand, because echo cancelers already exist in USA, Americans have proposed to use a 64-octet information field in order to improve transmission bandwidth efficiency. 48 octets has been selected as a compromise between the two.

	Class A	Class B	Class C	Class D
Timing Relation	required		not required	
Bit Rate	constant	variable		
Connection Mode	connection-oriented			connectionless
Examples	Digital Telephone Service	Compressed Audio and Video	Data Transfer, Signalling	Switched Multimegabit Data Services
AAL Type	1	2	3/4, 5	3/4

Table 1.1: Service classes in ATM and associated AAL protocol types.

functions are shifted to the network boundary, and if necessary, performed by AAL or higher layer protocols at the user-network interface.

ATM operates in a connection-oriented mode, and two types of logical connections are supported by the ATM layer [16], [21]. The basic type of connection is the *Virtual Channel* (VC) connection provided to the users for end-to-end data transport. The second one is the *Virtual Path* (VP) connection used internal to the network mainly for the purpose of reduced switching complexity. A VP can be viewed as a bundle of VCs which are switched together along a common path through the network. Each cell has a *VC* and a *VP Identifier* (VCI and VPI) in its header which respectively indicate the VC and the VP that the cell belongs to. At each node along a VP, cells are routed based only on their VPIs (VP switching). This reduces the complexity of routing decisions as the network deals with fewer, aggregated entities, i.e., VPs instead of individual VCs. Yet, a VC can consist of multiple VPs concatenated to form the end-to-end connection, and the individual VCs have to be switched similarly at the VP terminating and originating nodes (VC switching).

In ATM, services are divided into four classes, and different AAL protocols are employed to support each class successfully [16], [21]. This classification is made according to three basic parameters: timing relation between the source and the destination, bit rate characteristics, and connection mode. Classical digital telephone service is an example for constant bit rate services that require timing

relation between the source and the destination, and the AAL type 1 protocol absorbs the delay jitter due to variable queueing delays for this service. Table 1.1 summarizes other service classes and examples, and indicates the associated AAL protocol types. Note that, although ATM is tailored to connection-oriented services, transportation of connectionless traffic by the AAL type 3/4 protocols is also considered.

Like conventional packet-switched networks, an ATM network suffers from two sources of errors: imperfections of the physical transmission media, resulting in bit errors; and, buffer overflow during congestion, resulting in cell losses. Some precautions against bit errors have already been taken in ATM [16], [21]. For example, in the AAL type 2 and 3/4 protocols, 10 bits of the information field is reserved for a *Cyclic Redundancy Check* (CRC) code which can detect errors and correct up to two correlated bit errors in the information payload. Similarly, one octet of the cell header is an 8-bit CRC code which can correct single-bit errors and detect double-bit errors in the header. This code is used by the physical layer protocols to minimize switching errors, i.e., cell loss and insertion due to misrouting of cells with corrupted headers.

On the other hand, no dynamic actions are defined against buffer overflow in ATM [16]. Instead, preventive congestion control methods are employed to minimize the probability of this event. In the network design phase, physical allocation of transmission resources and dimensioning of buffers are made according to the expected traffic flow characteristics. In the network operation phase, sophisticated connection admission control protocols reject connection requests that would increase the probability of congestion too much if admitted. Once a connection is established, traffic policing mechanisms are used to ensure that the user does not overwhelm the allocated resources. Despite all these efforts, there will be congestion mainly because there is no link-by-link flow control in ATM [16]. Furthermore, due to very low bit error rates of optical fibers, congestion losses are expected to be the dominant source of errors in ATM networks [14]. Recovery of lost cells for services that cannot tolerate loss is the responsibility of the AAL protocols.

1.2 Lost Cell Recovery Techniques for ATM

There are two alternative error control techniques for packet-switched networks [20]. The first one is based on using error detecting codes and packet retransmissions by the source upon destination's requests, and hence, requires cooperation of the source and the destination. This technique is called automatic repeat request, or ARQ. There are various ARQ protocols, some tailored to simplicity (e.g., stop-and-wait ARQ) and some tailored to efficient link utilization (e.g., go-back-N and selective-repeat ARQ). Unlike ARQ, the second alternative is an open-loop technique which does not require cooperation of the end points. This technique, called *Forward Error Correction* (FEC), is based on using error correcting codes so that the destination itself can recover from errors without requiring packet retransmissions. The cost of this recovery capability is the additional transmission overhead associated with using error correcting codes instead of error detecting codes. Typically, the transmission overhead has to be doubled if errors are to be corrected rather than just being detected [7].

When ARQ and FEC are considered as candidate techniques for lost cell recovery in a wide area ATM network, two perspectives that favor the use of FEC arise [1], [14]. These are the perspectives of loss-sensitive services with and without real-time constraints. Here, ARQ suffers from the same problems as those encountered in satellite and deep-space communications, where FEC is widely used [4]. These problems are the large end-to-end propagation delay and the large ratio of propagation delay to cell transmission time.

The propagation speed through a guided medium such as the optical fiber or the coaxial cable is approximately 2×10^8 m/s [20]. Thus, the end-to-end propagation delay for a transcontinental or intercontinental connection of length, say, 10000 km is about 50 ms, and each retransmission increases the cell delay by more than the round-trip propagation delay, 100 ms. Many high-speed services such as real-time video, interactive computing, and distributed processing cannot tolerate that large retransmission delays. Therefore, FEC is definitely more appropriate than ARQ for services with stringent delay requirements.

One of the standardized data transmission rates in ATM is 155.52 Mbps [16], [21]. It takes about 2.7 μs to transmit a 53-octet cell at this rate, and this time is equivalent to the end-to-end propagation delay on a link of length 550 meters only. Therefore, even if the source utilizes only 10% of the 155.52 Mbps ATM transmission capacity, it can transmit more than 200 cells over a 550 km-long connection before it receives a signal from the destination about loss or success of the first cell it has transmitted. For longer connections, the number of cells in round-trip propagation may reach several thousands. This results in too high an ARQ protocol complexity as the source and the destination have to cooperate intensively in successful and in-sequence delivery of that many outstanding cells. FEC, not requiring source-destination cooperation, is preferable also for delay-tolerant services.

On the other hand, there are services which tolerate no loss at all. An example is file transfer. Since FEC is not a complete solution against cell losses in itself, retransmissions are inevitable for such services. In this case, ARQ can be used alone, or accompanied by FEC. The latter approach, i.e., hybrid FEC and ARQ, is advantageous because, by requiring fewer number of retransmissions as compared to ARQ alone, it not only reduces delay, thus improving the quality of service, but also can make more efficient use of the network resources.

1.3 FEC for ATM in the Literature

In 1975, Maxemchuk suggested the use of FEC to reduce the end-to-end delay for datagram-based services [13]. In a similar manner, FEC has been studied to improve reliability without increasing packet delay in high-speed networks, in particular ATM networks [1], [2], [5], [6], [11], [14], [15], [18], [19], [23], [24].

Since cell loss due to buffer overflow during congestion is expected to be the major source of errors in an ATM network as stated before, the common objective of all these studies is to recover lost cells. The basic idea in achieving this objective is to transmit separate parity cells along with the information-bearing ones. The parity cells are generated by using linear block coding techniques, and

the destination can recover all the lost cells in a block provided that sufficiently many cells of that block arrive successfully at the destination. In return for this recovery capability, the parity cells increase the network load, and in turn, the cell loss rate before decoding. Therefore, the code should introduce a reasonable parity overhead, and be still sufficiently powerful to overcome the opposite effect of load increase so that a significant net gain is achieved. Since lost cells can be identified at the destination by some means, usually by means of sequence numbers, the codes employed are commonly erasure correcting codes. These codes are more efficient than error correcting codes as they require typically half the parity overhead to recover the same number of losses per block [7].

The FEC schemes studied in the literature fall into two categories with respect to the particular block coding techniques employed. Some of the previous work focused on using codes over consecutive cells (simple block codes) [5], [6], [14], [18], [19], [23], [24], and some on using codes over equidistant cells (interleaved block codes) [1], [2], [11], [15].

Biersack carried out extensive simulations for the case of using simple block codes for a cell stream derived from a real, compressed motion picture interfering at a node with its time-shifted replicas as well as bursty streams [5]. In [6], he extended this study by using a simple analytical cell loss process characterization obtained via simulations. The results of [5] and [6] reveal that simple block coding can be quite effective for compressed video traffic, whereas it is inadequate for bursty traffic.

McAuley discussed various aspects of using FEC in broadband communications, and described an erasure correcting Reed-Solomon coder based system for ATM networks [14].

Shacham analyzed single-parity and some suboptimal multiple-parity block codes under the independent cell loss assumption [18]. He also considered the use of buffer management techniques to disperse cell losses, and hence, to improve the coding performance. Later, Shacham and McKenney extended this study, and showed by simulations that the independent cell loss assumption in the analysis may yield overly optimistic results [19]. They also considered but did not analyze

a simple form of the FEC scheme we study in this thesis.

Zhang and Sarkies analyzed the performance of simple block coding for a VP connection [23], [24]. They modeled the VP as a tandem queueing network, and assumed interrupted and Markov modulated Bernoulli cell sources. As a basis for the analysis in this context, they developed an approximate *Markov Chain* (MC) model estimating the end-to-end cell loss behavior.

Ayanoğlu *et al.* discussed incorporation of FEC into high-speed protocols [1]. In particular, they considered a hybrid FEC and ARQ scheme for minimizing the AAL *Protocol Data Unit* (PDU) retransmissions. The AAL PDUs are variable length user data units which are segmented into cells at the source site and reassembled at the destination site. The PDUs with missing cells are assumed to be lost, and retransmitted by the source. The FEC scheme they considered is based on transmitting separate parity PDUs along with the information-bearing ones, and lost PDUs can be recovered without requiring retransmissions if sufficiently many PDUs in the associated block arrive without any missing cells. Later, this scheme was shown to provide significant reductions in PDU delays by an approximate analysis for a multi-node VC connection [2].

Kitami and Tokizawa analyzed interleaved block codes under the independent cell loss assumption [11]. Later, Ohta and Kitami improved this analysis by using the Gilbert loss model, and dealt mainly with the practical issues of end-to-end erasure channel realization for a VP connection [15].

1.4 Objectives and Outline of the Thesis

Our principal motivation in this work is the fact that the cell loss process characteristics should be taken into account in designing an effective FEC scheme for ATM networks. Simple block coding is expected to be effective when losses are dispersed evenly over the cell stream (e.g., are random) [7]. However, due to the memory of the buffer overflow process, cell losses will be correlated in time, in other words, will occur in bursts. Code interleaving then appears preferable to simple block coding since it is a simple and effective way to recover from

burst losses [7]. However, the burstiness of cell losses strongly depends on the traffic characteristics. It is in general known that the more bursty the traffic is, the more bursty the cell loss process will be [5], [6]. Consequently, there may be situations where simple block coding can perform quite well with small to moderate block lengths, thus enabling fast recoveries for services with stringent delay requirements. For example, this is the case for compressed video traffic as the results of [5] and [6] indicate. This thesis considers the use of a *two-level FEC* scheme which exploits simple and interleaved block codes simultaneously to take the combined advantage of fast and burst loss recovery capabilities of the individual coding techniques.

Our main objective is to study the effect of traffic characteristics on the coding performance, and to demonstrate the usefulness of FEC, in particular two-level FEC. For this purpose, we carry out an extensive performance study comparing two-level FEC with the individual coding techniques for various traffic scenarios and network models.

For services with stringent delay requirements, we focus on a single-node VC connection. Modeling the node by an output-buffered ATM multiplexer, we quantify the reduction in the cell loss rate achieved by using FEC. This quantification is based on a novel and accurate discrete-time analytical cell loss model. We develop the model in two stages, and associated with each stage, carry out detailed performance analyses for diversely random and bursty traffic scenarios.

In the first stage, we construct an *exact MC* model for the output buffer occupancy with the traffic burstiness information *explicitly* incorporated into the model. Although the exact MC can be very large for realistic multiplexer parameters, it captures the bursty nature of cell losses precisely. Furthermore, by using simple source models, it can be made sparse. Therefore, by using iterative methods, we are able to compute the cell loss rate accurately for the uncoded and coded cases especially under bursty traffic conditions. Comparisons of the computational results with those obtained by simulations indicate that the exact MC model is quite accurate.

Because of the large size of the exact MC, the complexity of the iterative cell loss rate computations is too high with respect to both time and memory requirements. Consequently, the first stage analysis suffers from some practical limitations. That is, simple block codes are restricted to small block lengths, and more importantly, interleaved block and two-level codes are restricted to be decoded suboptimally. In the second stage, we aim to overcome this computational bottleneck. For this purpose, we introduce a novel MC cell loss model which is constructed based on the exact MC with reasonable computational burden. Unlike the exact MC, the new one is by definition highly sparse as there are only two next states reachable from any given state in just one transition. This reduces the complexity of iterative cell loss rate computations significantly. Therefore, in the second stage, we are able to study various codes without any practical restriction. Similar comparisons with simulation results indicate that the second stage analysis, and hence the new MC cell loss model, which we refer to as *reduced-complexity MC* model, are quite accurate under arbitrary traffic conditions.

The results of these performance analyses can be summarized as follows. First of all, FEC can effectively reduce the cell loss rate by several orders of magnitude over a wide range of network load and traffic characteristics. However, this requires the use of an appropriate coding technique chosen according to the traffic characteristics. The traffic stream for which FEC is desired is found to have a more important role in this regard as compared to interfering traffic. Typically, simple block coding can be quite fast and effective for a traffic stream of random nature. For a bursty traffic stream, however, the cell loss process is more bursty, and it becomes inadequate unless the block length is very large. In this case, code interleaving is a better FEC technique to use. Two-level FEC, combining the fast and burst loss recovery capabilities of the individual coding techniques, relaxes the need for a priori knowledge of the traffic stream. It allows fast loss recoveries if the traffic is random, and burst loss recoveries if the traffic is bursty. Therefore, it is attractive especially for traffic streams of unpredictable characteristics. Since a VP connection aggregates several individual VC traffic

Literature	Thesis
Simple block and interleaved block codes studied separately.	Simultaneous use of the two studied in detail, shown to be useful.
Two-level FEC based on single-parity codes mentioned [19].	Two-level FEC based on optimal multiple-parity codes discussed in detail (Ch. 2).
Approximate treatment of traffic burstiness in cell loss process characterization [23], [24].	Exact treatment: explicit incorporation of the traffic burstiness information into the exact MC model (Ch. 3).
Idea of the exact MC model exploited [8]; but not completely described for discrete-time systems with multiple sources.	Idea applied to a discrete-time system with multiple sources, the output-buffered ATM multiplexer (Ch. 3); used extensively (Ch. 4).
The Gilbert loss model used [15]; reflects cell loss behavior correctly, cell success behavior incorrectly.	Reduced-complexity MC model; reflects cell loss and success behaviors correctly (Ch. 3).
Effect of traffic characteristics on the performance of simple block coding technique studied [5], [6].	Simple block, interleaved block, and two-level coding techniques studied and compared as a complement in this regard (Ch. 4).
Hybrid FEC and ARQ studied by an approximate analysis for a multi-node VC connection; coding over PDUs considered [2].	Detailed simulation study carried out as a complement; coding within PDUs and coding over PDUs considered to be used individually as well as simultaneously (Ch. 5).

Table 1.2: Correspondence between the thesis and the literature.

streams of diverse characteristics and delay requirements, the use of two-level FEC is promising for a VP connection.

Motivated by services with stringent loss requirements, we also consider the use of two-level FEC in conjunction with ARQ. In this context, focusing on a four-node VC connection over a wide geographical area, we carry out a detailed simulation study to quantify the improvement in the delay-throughput performance achieved by using FEC under rather bursty traffic conditions. The results obtained indicate that significant savings in retransmissions are achievable even for very high network loads. In particular, two-level FEC is found to combine, or add, the gains of individual coding techniques.

In Table 1.2, the correspondence between the thesis and the previous work in the literature is summarized.

The rest of the thesis is organized as follows. Chapter 2 describes the two-level

FEC scheme in detail. We also address some practical issues related with erasure channel realization for VC and VP connections in Chapter 2. In Chapter 3, we describe the analytical performance model, i.e., the exact and reduced-complexity MC models, and the iterative methods we use to compute the cell loss rate. The results of the performance analyses are introduced and discussed thoroughly in Chapter 4. Chapter 5 introduces the simulation model for the hybrid scheme, and discusses the results obtained. We present concluding remarks in Chapter 6.

Chapter 2

Two-Level FEC for ATM

The two-level code we consider is a *product code* which has two independent dimensions or levels of coding [7]. A simple form of these codes is used in ASCII transmission to be able to detect burst as well as scattered bit errors [20]. Each 7-bit ASCII character is appended a parity check bit so that a character with odd number of bit errors can be detected (first level of coding). In addition to this, an 8-bit parity check character is appended to every group of $m - 1$ characters so that eight consecutive bit errors and many other bit error patterns in an m-character block can be detected (second level of coding). Here, we exploit this idea to recover from burst as well as random cell losses in an ATM network.

2.1 Two-Level FEC Scheme

The two-level coding scheme is shown in Figure 2.1. The information-bearing cells (i-cells) generated at the source are assumed to fill a $k' \times k$ matrix row-wise which is encoded as follows. First, each row of k i-cells is appended h parity cells (p-cells), which are generated by erasure correcting linear block coding techniques. Then, each column of k' cells of the resulting $k' \times n$ matrix is similarly appended h' parity cells (p$'$- and p$''$-cells respectively for i- and p-cell columns) to form an overall $n' \times n$ *coding matrix*, where $n = k + h$ and $n' = k' + h'$. Since both row and column codes are linear, the p$''$-cells can be generated according to either row

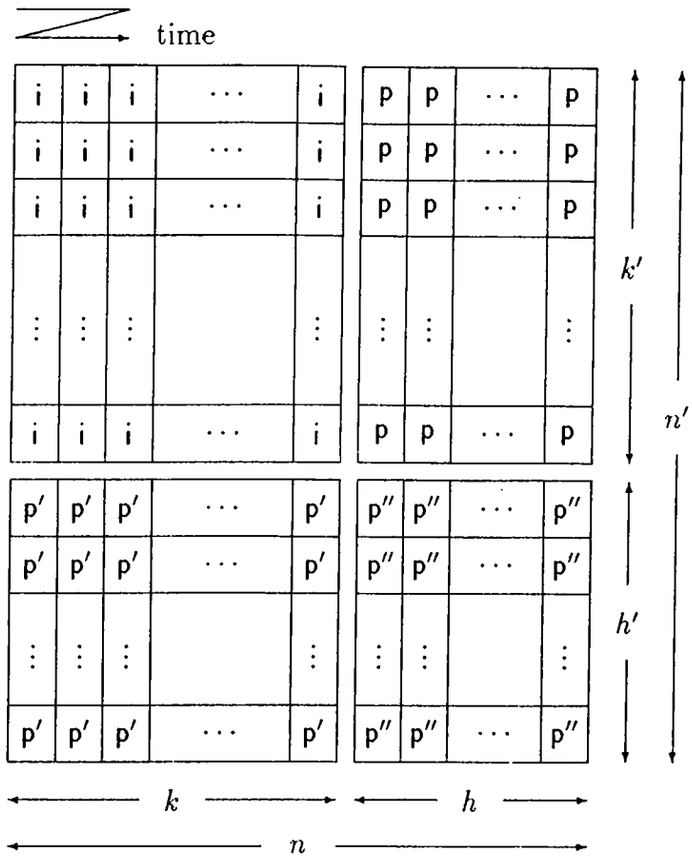

Figure 2.1: Two-level coding scheme.

or column coding rules, and be used accordingly. Observe that, by filling in and sending out the coding matrix row-wise as indicated in Figure 2.1, row coding corresponds to simple block coding, and column coding to code interleaving.

Similar to the common approach in the literature, we assume that the lost cells can be identified at the destination. This end-to-end erasure channel assumption enables the use of optimal, maximum-distance separable erasure correcting codes that can recover as many erased symbols as there are parity symbols in a block [7]. Such codes can be found, for example, in the class of well-known BCH codes, in particular Reed-Solomon codes. Some aspects of using these codes for the purpose of lost cell recovery are as follows. The encoding and decoding operations both

involve manipulation of the payloads of cells m bits taken at a time where m depends on the block length[1]. If the payload length in bits is not divisible by m, sufficiently many zeros are assumed to be padded to make it divisible. The encoding operation has a multiply-and-add nature which means that all the parity cells can be constructed in parallel without any encoding delay as the associated information-bearing cells are being transmitted. Typically, there will be some post-processing in decoding when there are losses to recover. The complexity associated with this post-processing, i.e., decoding complexity increases with the block length[2].

In this thesis, we assume that both rows and columns of the coding matrix are encoded optimally as just described so that up to h losses per row and h' per column can be recovered. With regard to decoding, we consider three particular decoders which are described below.

The simplest and perhaps the most intuitive one of the three first decodes rows one-by-one as they arrive, and then columns similarly for subsequent possible recoveries. Such a decoder, which we refer to as *single-trace decoder,* can recover up to $h'n + k'h$ losses out of nn' provided that they are distributed over the coding matrix appropriately. In particular, the loss of $h'n$ consecutive cells, or the complete loss of h' rows can be compensated for by column decoding if the number of losses in other rows does not exceed h per row. Since column recoveries may enable new row recoveries, and vice versa, still better performance is achievable. The *optimal decoder* is the one that keeps tracing rows and columns successively until no more possible recoveries are left. Unlike these two decoders, the third one performs column decoding in a rather constrained manner. That is, it treats unrecoverable rows with more than h losses as being "lost," and ignores the successful cells of such lost rows. If the number of lost rows in a coding matrix does not exceed h', it reconstructs the whole matrix. Otherwise, it does not act

[1] Here, an m-bit partition of the payload represents a symbol in $GF(2^m)$, and $2^m > n$ is required where n is the block length [7].

[2] In [7], various fast decoding algorithms are described for BCH and Reed-Solomon codes. With these algorithms, the asymptotic complexity of decoding is $O(n \log_2 n)$ for BCH codes, and is at most $O(n \log^2 n)$ (very nearly $O(n \log n)$) for Reed-Solomon codes, where n is the block length. These complexity figures indicate the number of multiplications in $GF(2^m)$.

at all, and all the recoveries will be those achieved by decoding rows only. Due to this on-off action, this decoder, which we refer to as *constrained decoder,* should be expected to have poorer performance as compared even to the single-trace decoder. Nevertheless, our results indicate that it can be appreciably effective as will be discussed in Chapters 4 and 5.

An important issue with regard to implementation of the three decoders described above is as follows. Observe that the destination has to have a buffer of capacity nn' cells to store incoming cells of a coding matrix since they will be used during post-processing, or decoding, in the case of losses. Now, while the decoder, any one of the three, is busy with post-processing of a coding matrix, a cell of the next coding matrix may arrive. Depending on the post-processing time and the instantaneous cell arrival rate, the arrival of multiple cells is even possible. Obviously, these cells must be stored until the decoder finishes with the current coding matrix. This means that an additional buffer space is required at the destination. Since the optimal decoder may go through more number of row and column decoding processes depending on the loss pattern, it requires a larger additional buffer as compared to the constrained and single-trace decoders. A simple solution here may be to employ two buffers, each of capacity nn' cells, at the destination. Then, while the decoder is post-processing a coding matrix that is stored in one of the buffers (foreground buffer), the cells of the next coding matrix can be stored in the other buffer (background buffer). Upon finishing with the current coding matrix, it can simply dump the contents of the background buffer to the foreground buffer, or can toggle between the two buffers treating them as foreground and background appropriately for successive coding matrices. However, if nn' is large, the additional buffer cost in this solution may be too high. Also, it seems quite pessimistic to expect that this much additional memory will really be required. Therefore, a compromise can be made. For example, one can content with an additional buffer of capacity n cells in which all the cells in the first row of the next coding matrix can be stored. In this case, instead of the single-trace and optimal decoders, we can consider a *best-effort decoder* which performs as much row and column traces as possible until the additional

buffer overflows. The performance of this decoder will then vary in time between those of the single-trace and optimal decoders depending on how fast this buffer overflows. In this thesis, we particularly focus on studying the performances of the constrained and optimal decoders, and assume that such a buffer overflow problem never arises.

So far, we have discussed "optimal two-level encoding and decoding" for a given set of parameters n, k, n', and k'. Obviously, this does not mean that the two-level code itself is optimal for the given overall block length nn' and code rate kk'/nn'. The optimal code here is a simple block code with kk' information-bearing cells and $nn' - kk'$ parity cells which can recover any pattern of $nn' - kk'$ lost cells out of nn'. However, this code suffers from higher decoding complexity as compared to the two-level code since its block length is larger. Using the two-level code instead of the optimal one is equivalent to trading decoding flexibility for reduced decoding complexity. Nevertheless, note that the optimal decoder described above provides a partial compensation for the loss of decoding flexibility as it can recover many patterns of more than $nn' - kk'$ lost cells. Also, the average decoding latency, i.e., the average time that a lost cell waits for a possible recovery, is smaller for the the two-level code than for the optimal one. This is justified by the fact that a row can be recovered immediately upon accumulation of sufficiently many successful cells of that row without waiting for the end of the coding matrix. Although this advantage is blurred by the requirement of in-sequence delivery of cells to the destination, it is still quite useful as will become clear at the end of Chapter 4.

2.2 Use of Two-Level FEC for VC and VP

FEC can be used for both VC and VP connections. The two approaches have their own advantages and disadvantages. The end-to-end nature of FEC for VC approach enables selective employment based on user demand. However, this requires encoding and decoding to be performed at the end terminals. Furthermore, if an individual VC connection is dedicated for a low bit rate service,

the average decoding latency may be too large unless short and ineffective codes are used. Nevertheless, FEC for VC approach is attractive especially for high bit rate services such as high quality video, or for the case of multiplexing several low bit rate services on a single VC.

FEC for VP approach, on the other hand, allows use of longer and much more effective codes as several low or high bit rate VC connections are aggregated on a VP, thus leading to a significant total traffic load. In this case, encoding and decoding are performed respectively at the originating and terminating nodes of the VP, and the associated hardware or software are shared by all the coexisting VC connections. However, since the cell sequence integrity has to be preserved within the network, the coded VP traffic will experience a decoding latency at the VP terminating node. Consequently, this may increase the end-to-end delay for a VC connection that consists of multiple concatenated VPs if FEC is performed for each VP. Nevertheless, the use of FEC is still practical and useful for long-distance and/or heavily loaded VPs since then the decoding latency becomes negligible as compared to the propagation delay.

In the following, we discuss two practical problems with regard to erasure channel realization, i.e., lost cell identification, for the cases of using two-level FEC for VC and VP connections.

2.2.1 Two-Level FEC for a VC Connection

In the AAL type 1, 2, and 3/4 protocols, cells belonging to a VC connection are numbered sequentially as they are being transmitted. There is a 4-bit *Sequence Number* (SN) subfield reserved for this purpose in the 48-octet information field [16], [21]. Then, by transmitting all the parity cells over the same VC as the information-bearing ones, the destination can determine the locations of lost cells in the coding matrix by detecting the gaps in the modulo-16 SN sequence. However, the consecutive loss of more than 15 cells, since it cannot be detected in this manner, is a potential source of decoding errors. Here, we describe a method which makes use of the correlation that already exists in the cell stream due to

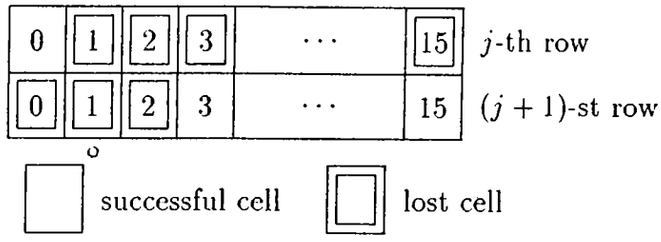

Figure 2.2: Consecutive loss of 18 cells.

coding to detect and mitigate the effects of such ill-conditioned losses.

Suppose that, for example, $n = 16$ and 18 cells are lost consecutively as shown in Figure 2.2. Then, the row decoder will misinterpret the received SN sequence, and regard the successful cells 0 of the j-th row and $3, 4, \ldots, 15$ of the $(j+1)$-st row as belonging to a single row the second and third cells of which are lost. Consequently, it will attempt to recover them if $h \geq 2$. These are obviously unintended recoveries. Since each valid row has to be a consistent block according to the row coding rules, such unsuccessful attempted recoveries can be detected in certain cases as discussed below.

Let l be the number of lost cells detected in a row. Then if $l > h$, lost cell recovery is not possible, and the destination contents with the successful cells it has already received. Otherwise, all the lost cells can be recovered. In particular if $l < h$, any set of $h - l$ successful cells can be regarded as being lost, and recovered together with the l actually lost ones by using the remaining k successful cells. Then, a comparison of these recovered cells with the respective successful ones indicates whether the row is valid or not. If they all match, the row and the recoveries are validated. A mismatch, on the other hand, indicates that the row is not a valid one, and the recoveries of l actually lost cells are discarded. The probability of a coincidental match in this comparison is $2^{-(h-l)i}$, where i is the number of bits in the payload of a cell. Since i is 352 in the worst case[3], this probability is negligible even when $l = h - 1$. Therefore, row validation can be

[3]The payload lengths are 47, 45, and 44 octets respectively for the AAL type 1, 2, and 3/4 protocols [16], [21].

done safely by regarding only one of the successful cells as being lost.

Now observe that, when $n > 16$, the decoding error due to the consecutive loss of more than 15 cells propagates since the subsequent rows will similarly be misinterpreted. In other words, row-level synchronization between the encoder and the decoder will be lost. The validation procedure described above can also avoid decoding error propagation. When $n > 16$ is desired, it is reasonable in this regard to choose n as an integer multiple of 16. With this choice, each row is known to start and end with the same SNs, 0 and 15 without loss of generality, and it is easier for the decoder to maintain row-level synchronization with the encoder. That is, in the case of detecting an invalid row, the procedure can be repeated successively each time shifting the potential row by 16 on the cell stream until a subsequent valid row is reached.

Finally note that the complete loss of one or more rows in a coding matrix disrupts proper operation of the column decoder similarly. The same procedure can also be performed over columns to validate the whole coding matrix, and hence, to achieve matrix-level synchronization between the encoder and the decoder.

2.2.2 Two-Level FEC for a VP Connection

Because the major purpose of using SNs is to enable in-sequence delivery of cells to the end user, no SNs are provided for cells belonging to a VP connection, which is used internal to the network. Therefore, a mechanism has to be devised to identify the lost cells in a coding matrix when two-level FEC is used for a VP connection. Ohta and Kitami have previously proposed transmission of auxiliary *Cell Loss Detection* cells (CLD-cells) to be used for this purpose in the case of coding over columns only. Here, we adopt this idea to the case of two-level coding, and briefly describe the mechanism in the following. For a more detailed description of the original mechanism, we refer the reader to [15].

The mechanism is illustrated in Figure 2.3. The encoding operation is performed in the same manner as described at the beginning of Section 2. The only

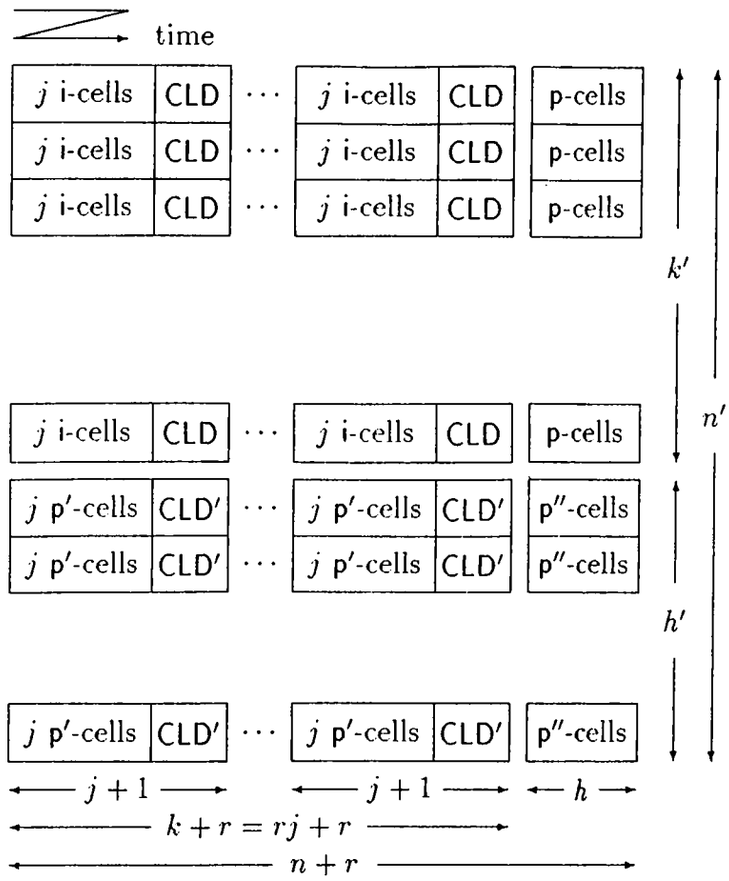

Figure 2.3: Cell loss detection mechanism for the case of two-level FEC for VP.

difference is that one CLD-cell is transmitted after every group of j i-cells. In the information field of each CLD-cell, there are 22-bit *Cell Recognition Patterns* (CRPs) copied from the preceding j i-cells. A CRP consists of the 16-bit VCI of the header and the first 6 bits of the information field of the corresponding i-cell. The VCIs remain unchanged as cells travel along the VP, and the first 6 bits of the information field include the 4-bit VC-SN [16], [21]. Therefore, the loss pattern in a block of j i-cells can be uniquely determined at the VP terminating node via a CRP matching procedure, as described in [15], provided that j is chosen properly and the corresponding CLD-cell itself is not lost.

When a CLD-cell is lost, the loss pattern in the associated block of j i-cells

cannot be determined uniquely, and lost cell recovery within that row is usually not possible. Therefore, the CLD-cells are excluded from row coding, and the h p-cells at the end of each row are generated by using the preceding rj i-cells only.

By transmitting all the parity cells (p-, p′-, and p″-cells) over a dedicated VC, lost parity cells can be identified based on the 4-bit VC-SNs. Therefore, they need not be appended CLD-cells. The CLD′-cells are parity cells generated according to the column coding rules by using the CLD-cells in the corresponding column. This provides a protection against loss of CLD-cells. One octet of the information fields of the CLD- and CLD′-cells is used for an 8-bit SN. Then, by transmitting all the CLD- and CLD′-cells over a dedicated VC, lost CLD- and CLD′-cells can be identified based on these 8-bit CLD-SNs.

Similar to the case for information-bearing rows (first k' rows), the CLD′-cells are excluded from coding within parity rows (last h' rows). Then, the p″-cells are consistent parity cells according to both row and column coding rules. Consequently, having received or recovered all the CLD-cells, the decoder can identify all the lost cells in the coding matrix, and recover up to h losses per row and h' losses per column as desired.

Observe that this mechanism introduces a CLD- and CLD′-cell overhead in addition to the original parity cell overhead. Obviously, the larger the number j is, the more efficient the mechanism will be. As one octet of the information field of a CLD-cell is an 8-bit CLD-SN, the number of 22-bit CRPs that can fit in the remaining 47 octets is 17. However, a smaller value of j is required for proper operation of the mechanism. Ohta and Kitami showed that j has to be at most 15 so that the CRP matching procedure does not fail even in the worst case that all the i-cells in a row belong to the same VC and more than 15 cells, including a CLD-cell, are lost consecutively [15].

Chapter 3

Analytical Framework

In this chapter, we describe a discrete-time analytical framework to evaluate the performance of two-level FEC for a particular VC connection. The *tagged* traffic belonging to this connection is assumed to traverse a single node at which it interferes with $N-1$ independent *untagged* traffic streams. We assume that there is an $N \times M$ ATM switch with dedicated output queueing at the node. Concentrating on the tagged output port, the node is modeled by an N-to-1 ATM multiplexer with a FIFO output buffer of capacity B cells, as shown in Figure 3.1.

Our main objective here is to quantify the *Cell Loss Rate* (CLR) accurately for both constrained and optimal decoders. This requires the development of a tagged cell loss process model into which the traffic burstiness information is incorporated. In the following, after describing the input traffic model in Section 3.1, we introduce two such models. The first model is straightforward and exact, but has very high complexity. We discuss this model and its use in the analysis for the constrained decoder in Section 3.2. Since the exact model cannot be used when the columns of the coding matrix are decoded independently, we develop a simpler model which has much less complexity, and discuss its use for the case of optimal decoding in Section 3.3. In all these analyses, we assume that the end-to-end system recovers from losing more than 15 cells consecutively.

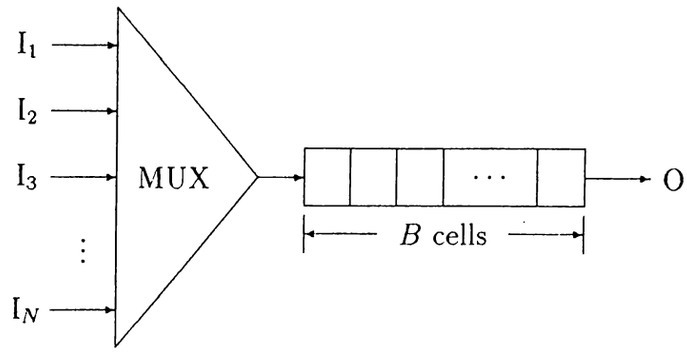

Figure 3.1: Output-buffered N-to-1 ATM multiplexer.

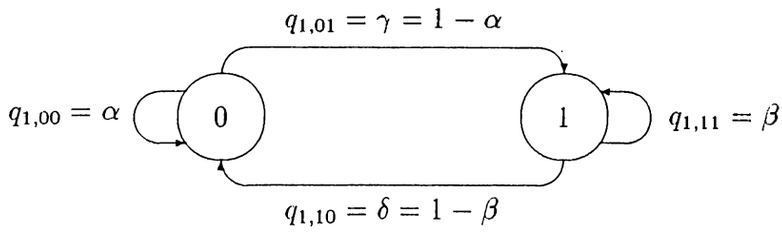

Figure 3.2: MC model of an individual source.

3.1 Input Traffic Model

We consider N discrete-time Markovian cell sources, one tagged and $N - 1$ independent untagged sources, at the input of the multiplexer. The MC model of one source is shown in Figure 3.2, where the states 0 and 1 correspond respectively to the "idle" and "active" states of the source. The state transitions occur once per *slot*, which is the unit time required to transmit a cell over the output link. A source generates one cell per slot when active, and no cell at all when idle. Then, the stationary probability of the active state, given by

$$\rho_1 = (1 - \alpha)/(2 - \alpha - \beta), \tag{3.1}$$

is the normalized load offered by one source, where α and β are respectively the idle-to-idle and active-to-active state transition probabilities as shown in Figure 3.2.

Observe that periodic insertion of parity cells during encoding, as described

in Section 2.1, destroys the Markovian behavior of the tagged source. However, we assume that it is preserved even after encoding, and reflect the effect of parity overhead in the traffic parameter β only so that the tagged load is increased by the factor $1/R$, where R is the code rate kk'/nn'. That is, we modify β as

$$\beta_e = (1 - R)(2 - \alpha) + R\beta \tag{3.2}$$

so that the normalized effective tagged load becomes

$$\rho_{et} = (1 - \alpha)/(2 - \alpha - \beta_e) = \rho_t/R, \tag{3.3}$$

where ρ_t is the normalized tagged load before encoding. The assumption that the traffic parameter α remains the same as before parity cell insertion is reasonable due to the memoryless property of the geometric idle duration distribution [12].

We assume that $N - 1$ untagged sources are independent and identical. The aggregate untagged source can then be modeled by an N-state MC with the state variable being the number of active untagged sources, or equivalently, the number of simultaneous untagged cells generated in a slot. The state transition probabilities for this MC follow from those for the individual source MC. The subscript 1 in $q_{1,ij}$ of Figure 3.2 is attributed to the single source. With similar notation, let $q_{N-1,ij}$, $i, j = 0, 1, 2, \ldots, N - 1$, be the state transition probabilities for this MC. Observe that a transition from state i to state j occurs when l of i active sources become idle, and $j - (i - l)$ of $N - 1 - i$ idle sources become active simultaneously in a slot. Then, we have

$$q_{N-1,ij} = \sum_{l=\underline{l}}^{\bar{l}} \binom{i}{l} \delta^l \beta^{i-l} \binom{N-1-i}{j-i+l} \gamma^{j-i+l} \alpha^{N-1-j-l}, \tag{3.4}$$

where $\underline{l} = \max(0, i - j)$ and $\bar{l} = \min(i, N - 1 - j)$. It can be shown that (3.4) gives the transition probabilities of Figure 3.2 when $N = 2$, i.e., when there is only one untagged source.

The individual cell source described above is much simpler than the interrupted or Markov modulated Bernoulli sources, which are used widely in the literature, e.g., in [23] and [24]. Yet, this simple source is capable of generating

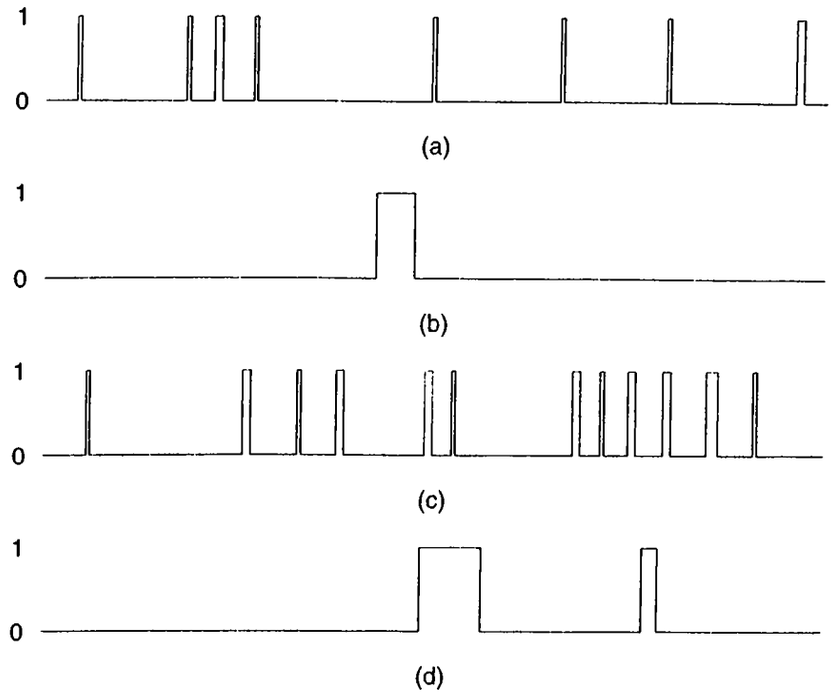

Figure 3.3: Some typical cell stream realizations over 200-slot periods for the following offered loads and clustering coefficients: (a) $\rho_1 = 0.05$, $c = 1.2$; (b) $\rho_1 = 0.05$, $c = 1.9$; (c) $\rho_1 = 0.1$, $c = 1.2$; and (d) $\rho_1 = 0.1$, $c = 1.9$. Cells generated consecutively are represented by a single pulse of height 1 and width proportional to the number of cells in the batch.

traffic streams of diverse characteristics. It generates correlated, or bursty, traffic in general. But when $\alpha + \beta = 1$, the probability of having a cell generated in a slot is the same regardless of the state of the source in the previous slot. We then have an independent Bernoulli source, and a completely uncorrelated traffic is generated. The *clustering coefficient,* defined as

$$c \triangleq \alpha + \beta \in (0, 2) \tag{3.5}$$

serves as a measure of the burstiness of a source when the offered load is small. In general, the closer c to 2 is, the more bursty the source becomes. In Figure 3.3, the typical cell stream realizations obtained over 200-slot periods for the offered loads

$\rho_1 = 0.05$ and $\rho_1 = 0.1$ illustrate the diversely random and bursty characteristics of the sources with $c = 1.2$ and $c = 1.9$, respectively. Also note that the offered load and the clustering coefficient pair completely characterize a source. That is, given ρ_1 and c, the traffic parameters α and β follow from (3.1) and (3.5) as

$$\alpha = 1 - (2 - c)\rho_1 \text{ and } \beta = c - \alpha = 1 - (2 - c)(1 - \rho_1). \tag{3.6}$$

3.2 Analysis for the Constrained Decoder

Given the multiplexer and traffic parameters, to analyze the performance of two-level FEC with the constrained decoder, we first construct an MC for the output buffer occupancy with the states of the tagged and aggregate untagged sources augmented *explicitly* into the state variable. Recently, Cidon *et al.* used the same idea to analyze the packet loss process for continuous- and discrete-time systems, but did not completely describe the particular case of a discrete-time system with multiple sources [8]. Although explicit source state augmentation results in a very large MC for realistic multiplexer parameters N and B, the augmented MC model captures the bursty nature of cell losses precisely, and hence, constitutes a basis for accurate CLR computations. For the particular cases of coding neither rows nor columns and coding only rows, the CLR can be computed based directly on this model, which we refer to as *exact cell loss model*. However, when there is column coding, one also needs a model for the row loss process since the constrained decoder manipulates each row as a whole during column decoding. For this purpose, we use a simple, two-state MC model, i.e., the *Gilbert Loss Model* (GLM), the parameters of which can be found based on the exact model. In all these CLR and GLM parameter computations, an iterative method similar to those of [8], [23], and [24] is used in different forms.

3.2.1 Exact Cell Loss Model

To characterize the tagged cell loss process without ignoring the traffic burstiness, we construct a 3-*Dimensional* (3-D) MC for the output buffer, which is driven

by one tagged and $N-1$ untagged cell sources. The state variable is the (b,t,u) triplet, where $t \in \{0,1\}$ and $u \in \{0,1,2,\ldots,N-1\}$ are respectively the numbers of tagged and untagged cell arrivals, and $b \in \{0,1,2,\ldots,B\}$ is the buffer occupancy after $t+u$ new cells arrive in a slot. We consider an ideal multiplexer capable of transporting all the simultaneous input cells to the output buffer in zero time without internal blocking. If the buffer is empty, a randomly chosen one of the new cells is transmitted immediately on the output link, and the rest is stored in random order. Otherwise, the leading cell in the buffer is transmitted first, and the new cells are stored in random order again, provided that there is enough room for them. Therefore, the buffer occupancy is characterized by the relation

$$b_i = \min(B, \max(0, b_{i-1} + t_i + u_i - 1)) \tag{3.7}$$

where i is the slot index.

Observe that there are $2N(B+1)$ distinct (b,t,u) triplets in this construction. However, some of them are inconsistent with (3.7), or correspond to transient states due to the boundary conditions at $b=0$ and B. It follows from (3.7) that

(i) if $b_i < B$, then $t_i + u_i > b_i + 1$ implies that $b_{i-1} < 0$, or we have more than one cell transmitted in slot i; and,

(ii) if $b_i = B$, then $t_i + u_i = 0$ implies that $b_{i-1} > B$, or the buffer remains full although no new cells arrive in slot i.

Discarding these transient states, we have the irreducible state space

$$\mathcal{S} = \{(b,t,u) : t+u \leq b+1 \text{ if } b < B, \text{ and } t+u > 0 \text{ if } b = B\} \tag{3.8}$$

the state transition probabilities for which follow from those for t and u given by (3.4). That is, a transition from state $(b,t,u) \in \mathcal{S}$ to $(b',t',u') \in \mathcal{S}$ occurs with probability $q_{1,tt'}q_{N-1,uu'}$ if $b' = \min(B, \max(0, b+t'+u'-1))$, and with probability 0 otherwise.

Now observe that the state space $\mathcal{S}$ can be partitioned into two subspaces

$$\mathcal{S}_D = \{(b,0,u) \in \mathcal{S}\} \tag{3.9}$$

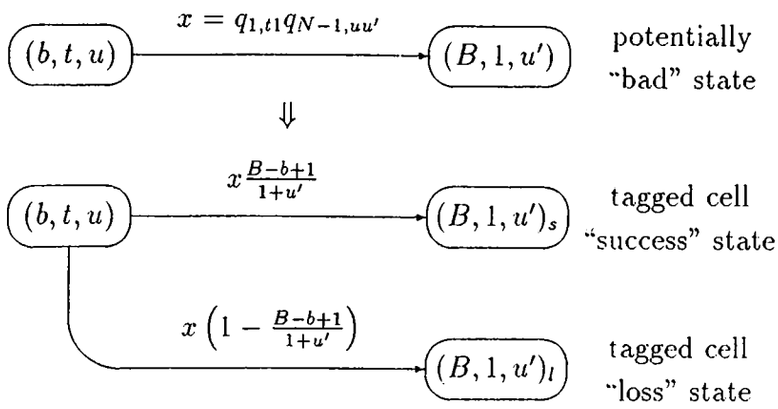

Figure 3.4: Modification of the state transition probabilities during the decomposition of a potentially "bad" state into tagged cell "success" and "loss" states, where it is assumed that $u' > B - b$.

and

$$\mathcal{S}_A = \{(b,1,u) \in \mathcal{S}\}, \tag{3.10}$$

where $\mathcal{S}_D$ is the set of "don't care" states in which there is no tagged cell arrival, and $\mathcal{S}_A$ is the set of tagged cell "arrival" states. The "arrival" states $(B,1,u) \in \mathcal{S}_A$ with $u \geq 1$ are distinguished from the others since we may have a tagged cell loss while making transition to such a state. Suppose that we are in state $(b,t,u) \in \mathcal{S}$ at the end of a slot. Then, the maximum number of new cells that can be served in the next slot is $B - b + 1$. If there are $t' + u' > B - b + 1$ new cells, randomly chosen $t' + u' - (B - b + 1)$ of them are lost. In particular, if $t' = 1$ and $u' > B - b$, the tagged cell is lost with probability $1 - (B - b + 1)/(1 + u')$, and the new state becomes $(B,1,u')$. Since pure "success" and "loss" states are preferable to such potentially "bad" states, we decompose each state $(B,1,u) \in \mathcal{S}_A$ with $u \geq 1$ into two states $(B,1,u)_s$ and $(B,1,u)_l$, which respectively indicate tagged cell "success" and "loss." After this decomposition, illustrated in Figure 3.4, we have an *extended 3-D MC* with state space $\mathcal{S}_E = \mathcal{S}_D \cup \mathcal{S}_S \cup \mathcal{S}_L$, where

$$\mathcal{S}_S = (\mathcal{S}_A \setminus \{(B,1,u) : u \geq 1\}) \cup \{(B,1,u)_s : u \geq 1\} \tag{3.11}$$

and

$$\mathcal{S}_L = \{(B,1,u)_l : u \geq 1\} \tag{3.12}$$

are the sets of pure tagged cell "success" and "loss" states, respectively.

Note that the extended 3-D MC can be very large for realistic N and B. The number of states in $\mathcal{S}_E$ can be shown to be

$$|\mathcal{S}_E| = 2N(B+1) - (N^2 - 3N + 3) \tag{3.13}$$

if $B \geq N - 2$. For example, when $N = 16$ and $B = 100$, we have $|\mathcal{S}_E| = 3021$, and the solution for stationary state probabilities may seem infeasible. However, owing to the simplicity of the input traffic model, the number of states reachable from any state in $\mathcal{S}_E$ in just one transition is at most $3N - 1$ in the worst case. Therefore, the state transition probability matrix of the extended 3-D MC is sparse if B is sufficiently larger than N.

3.2.2 Computation of the CLR: No Column Coding

Now we are ready to compute the CLR if there is no column coding. In particular, when there is no coding at all, the CLR follows immediately from the stationary state probability distribution for the extended 3-D MC. That is, defining p_s as the stationary probability of state $s \in \mathcal{S}_E$, the CLR for the no coding case is given by

$$P_{l,nc} = \frac{1}{\rho_t} \sum_{s \in \mathcal{S}_L} p_s, \tag{3.14}$$

where ρ_t is the normalized tagged load equal to the sum of all p_s for $s \in \mathcal{S}_S \cup \mathcal{S}_L$.

For the case of coding only rows, which we refer to as *row-only coding,* the CLR is defined as the ratio of the expected number of unrecoverable losses in a row to the row size n. Therefore, if $\{F(v,n) : 0 \leq v \leq n\}$ is the *Cell Loss Pattern Distribution* (CLPD) over a row, where $F(v,n)$ is the probability of having v losses in a row, the CLR for this case is given by

$$P_{l,roc} = \frac{1}{n} \sum_{v=h+1}^{n} vF(v,n) \tag{3.15}$$

since up to h losses per row can be recovered.

The CLPD over a row can be found based on the extended 3-D MC model as follows. Let $f_s(v,w)$, $v = 0, 1, 2, \ldots, w$, be the conditional probability that v

Algorithm 3.1: Iterative computation of $f_s(v, n-1)$.

(1) Initial condition: $f_s(0,0) = 1$ for all $s \in \mathcal{S}_E$.

(2) Boundary condition: $f_s(v,w) = 0$ for all $s \in \mathcal{S}_E$ if $v < 0$ or $v > w$.

(3) Set $w := 1$.

(4) For $v = 0, 1, 2, \ldots, w$, find $f_s(v,w)$ for $s \in \mathcal{S}_D$ by using (3.16) and the results of the previous iteration cycle.

(5) For $v = 0, 1, 2, \ldots, w$, compute $f_s(v,w)$ for $s \in \mathcal{S}_S \cup \mathcal{S}_L$ by (3.16) using the results of step *(4)* and the previous iteration cycle.

(6) If $w < n-1$, set $w := w+1$, and go to step *(4)*; otherwise, stop.

of next w tagged cells are lost given that the system is initially in state $s \in \mathcal{S}_E$. Also let $q_{ss'}$ be the transition probability from state $s \in \mathcal{S}_E$ to $s' \in \mathcal{S}_E$. Then, we have the recursive relation

$$f_s(v,w) = \sum_{s' \in \mathcal{S}_D} q_{ss'} f_{s'}(v,w) + \sum_{s' \in \mathcal{S}_S} q_{ss'} f_{s'}(v,w-1) + \sum_{s' \in \mathcal{S}_L} q_{ss'} f_{s'}(v-1,w-1) \tag{3.16}$$

for all $s \in \mathcal{S}_E$. That is, when the system is currently in state s, and we are looking forward for v losses out of w, we will be doing so for $v-1$ losses out of $w-1$ if the next state is a "loss" state, and still for v losses out of $w-1$ if it is a "success" state. On the other hand, if the next state is a "don't care" state, we do not have a tagged cell arrival, and hence, keep looking forward for v losses out of w. The recursive relation (3.16) enables $f_s(v, n-1)$, $0 \le v < n$, be computed iteratively for all $s \in \mathcal{S}_S \cup \mathcal{S}_L$, as illustrated in Algorithm 3.1. Then, $F(v,n)$, $0 \le v \le n$, follows by the weighted average[1]

$$F(v,n) = \frac{1}{\rho_{et}} \left(\sum_{s \in \mathcal{S}_S} p_s f_s(v, n-1) + \sum_{s \in \mathcal{S}_L} p_s f_s(v-1, n-1) \right), \tag{3.17}$$

where ρ_{et} is the normalized effective tagged load given by (3.3). Similar to the case in (3.14), ρ_{et} is equal to the sum of all p_s for $s \in \mathcal{S}_S \cup \mathcal{S}_L$.

[1]Observe that, while finding $F(0,n)$ and $F(n,n)$ by (3.17), we have $f_s(-1, n-1) = 0$ for $s \in \mathcal{S}_L$ and $f_s(n, n-1) = 0$ for $s \in \mathcal{S}_S$ from the boundary condition, i.e., step *(2)* of Algorithm 3.1.

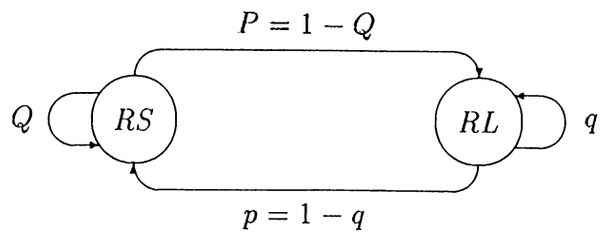

Figure 3.5: The Gilbert loss model for the row loss process.

Note that, in step *(4)* of Algorithm 3.1, we have a set of $|\mathcal{S}_D|$ linear equations with $|\mathcal{S}_D|$ unknowns, where $|\mathcal{S}_D|$ is the number of states in $\mathcal{S}_D$. Solving for these unknowns is equivalent to solving a matrix equation of the form $A\overline{x} = \overline{y}$ for vector $\overline{x}$. It can be shown that

$$|\mathcal{S}_D| = (|\mathcal{S}_E| - 1)/2 = N(B+1) - (N^2 - 3N + 4)/2 \tag{3.18}$$

if $B \geq N - 2$, and hence, $|\mathcal{S}_D|$ can be very large for realistic N and B. But, the matrix A is sparse, and we can solve for the unknowns. This follows from the fact that $A = I - A_{DD}$ where I is the $|\mathcal{S}_D| \times |\mathcal{S}_D|$ identity matrix, and A_{DD} is the transition probability matrix among "don't care" states, which has only N non-zero entries per row. However, Algorithm 3.1 requires $n(n+1)/2 - 1$ such solutions, and its complexity is determined mainly by step *(4)*. Therefore, a row-only code should not have too large a row size n so that the CLR can be computed in this manner within a reasonable time.

3.2.3 Computation of the CLR: With Column Coding

Recall that the constrained decoder treats unrecoverable rows with more than h losses as being "lost," and ignores them during column decoding. Due to the correlation between the adjacent cells of consecutive rows, row losses will be correlated in time. This has to be taken into account in order to obtain accurate results when there is column coding. For this purpose, we approximate the row loss process by the GLM, as shown in Figure 3.5. The states RS and RL respectively stand for "row success" and "row loss," where "row success" means

that a row has no more than h losses, and hence, can be completely recovered. The GLM parameters Q and q of Figure 3.5 are computed based on the extended 3-D MC model as discussed in Appendix 3.A.

For the case of two-level coding, which we also refer to as *joint row-and-column coding,* or simply as *joint coding,* the CLR is defined as the ratio of the expected number of unrecoverable losses in a coding matrix to the coding matrix size nn'. Therefore, the CLR after constrained decoding of a joint code is given by

$$P_{l,jc} = \frac{1}{nn'} \mathrm{E}\left[N_{rl}\right] \mathrm{E}\left[N_{cl} \mid RL\right], \tag{3.19}$$

where $\mathrm{E}\left[N_{rl}\right]$ is the expected number of unrecoverable row losses in a coding matrix, and $\mathrm{E}\left[N_{cl} \mid RL\right]$ is the expected number of cell losses in a row that is given to be lost. $\mathrm{E}\left[N_{cl} \mid RL\right]$ follows from the CLPD over a row:

$$\mathrm{E}\left[N_{cl} \mid RL\right] = \left(\sum_{v=h+1}^{n} F(v,n) \right)^{-1} \sum_{v=h+1}^{n} vF(v,n). \tag{3.20}$$

To compute $\mathrm{E}\left[N_{rl}\right]$, we first need to find the *Row Loss Pattern Distribution* (RLPD) $\{F'(v,n') : v = 0,1,2,\ldots,n'\}$, where $F'(v,n')$ is the probability of having v rows lost in a coding matrix of n' rows. Then, we will have

$$\mathrm{E}\left[N_{rl}\right] = \sum_{v=h'+1}^{n'} vF'(v,n'). \tag{3.21}$$

For the particular case of coding only columns, which we refer to as *column-only coding,* the CLR, $P_{l,coc}$, will follow similarly from (3.19), (3.20), and (3.21) by inserting $h = 0$ in (3.20).

The RLPD is found based on the GLM of Figure 3.5 as follows. Let $f'_{rs}(v,w)$ and $f'_{rl}(v,w)$, $v = 0,1,2,\ldots,w$, be the conditional probabilities of losing v of next w rows given that the current row is successful and lost, respectively. Similar to (3.16), we have the following two interrelated recursive relations:

$$f'_{rs}(v,w) = Qf'_{rs}(v,w-1) + (1-Q)f'_{rl}(v-1,w-1) \tag{3.22}$$

and

$$f'_{rl}(v,w) = (1-q)f'_{rs}(v,w-1) + qf'_{rl}(v-1,w-1). \tag{3.23}$$

Algorithm 3.2: Iterative computation of $f'_{rs}(v, n'-1)$ and $f'_{rl}(v, n'-1)$.

(1) Initial condition: $f'_{rs}(0,0) = f'_{rl}(0,0) = 1$.

(2) Boundary condition: $f'_{rs}(v,w) = f'_{rl}(v,w) = 0$ if $v < 0$ or $v > w$.

(3) Set $w := 1$.

(4) For $v = 0, 1, 2, \ldots, w$, compute $f'_{rs}(v,w)$ and $f'_{rl}(v,w)$ respectively by (3.22) and (3.23) using the results of the previous iteration cycle.

(5) If $w < n'-1$, set $w := w+1$, and go to step *(4)*; otherwise, stop.

By using (3.22) and (3.23), $f'_{rs}(v, n'-1)$ and $f'_{rl}(v, n'-1)$, $0 \leq v < n'$, are computed iteratively as illustrated in Algorithm 3.2. Then, $F'(v, n')$, $0 \leq v \leq n'$, follows by the weighted average[2]

$$F'(v, n') = P_{rs} f'_{rs}(v, n'-1) + P_{rl} f'_{rl}(v-1, n'-1), \tag{3.24}$$

where P_{rs} and P_{rl} are the stationary probabilities of states RS and RL, respectively. Similar to (3.1), these probabilities are given by

$$P_{rl} = 1 - P_{rs} = (1-Q)/(2-Q-q). \tag{3.25}$$

Observe that, given the GLM parameters Q and q, Algorithm 3.2 is of negligible complexity as compared to Algorithm 3.1. However, the computation of Q and q requires an additional execution of Algorithm 3.1 (see Appendix 3.A). Therefore, the complexity of CLR computations for the column-only and joint coding cases are determined by step *(4)* of Algorithm 3.1. In other words, the row size n is the major coding parameter that is constrained so that the CLPD and RLPD computations based on the extended 3-D MC model are both feasible, whereas there is not any practical limit on the column size n'.

3.3 Analysis for the Optimal Decoder

Observe that the extended 3-D MC contains too much redundancy. The "don't care" states do not provide direct information about success or loss of tagged cells,

[2]Similar to the case in (3.17), we have $f'_{rs}(n', n'-1) = f'_{rl}(-1, n'-1) = 0$ in (3.24).

but they constitute roughly half of the state space (compare $|\mathcal{S}_E|$ of (3.13) and $|\mathcal{S}_D|$ of (3.18)). Also, "success" and "loss" states themselves contain appreciable redundancy as they provide precise information about the buffer and aggregate untagged source states. So much redundancy makes it impractical to analyze the performance of the optimal decoder based on the exact cell loss model as briefly discussed below.

Recall that, in optimal decoding, the columns of the coding matrix are decoded independently. The CLR computations then require the cell loss process over a column be characterized. Raising the state transition probability matrix of the extended 3-D MC to power n serves this purpose. But, due to fill-ins, the resulting matrix will not be sparse unless the row size n is very small. On the other hand, although it may be computationally demanding, it is theoretically possible to eliminate the "don't care" states so that the number of states we have to deal with is roughly halved. But, since transition from any "success" or "loss" state to any other is possible through the "don't care" states, we then have a full state transition probability matrix. Therefore, computation of the CLR for the optimal decoder based on the exact cell loss model is not feasible for realistic multiplexer parameters N and B.

Here we introduce a simpler MC cell loss model that provides minimal necessary information without any redundancy. This MC, which we refer to as *reduced-complexity cell loss model,* is constructed based on the extended 3-D MC as accurately as desired. Although it can again be large depending on the traffic parameters and the desired accuracy, it is highly sparse with only two next states reachable from any given state. Therefore, it enables the cell loss process over a column be characterized unless the row size n is impractically large, and we can compute the CLR for most column-only codes of practical interest even when the columns are decoded independently. Also, it does not impose any practical limit on n when the CLR for a row-only code is to be computed. Finally, providing direct information about consecutive cell success and loss lengths, it allows fast CLR simulations for joint coding with the optimal decoder, which is difficult to analyze.

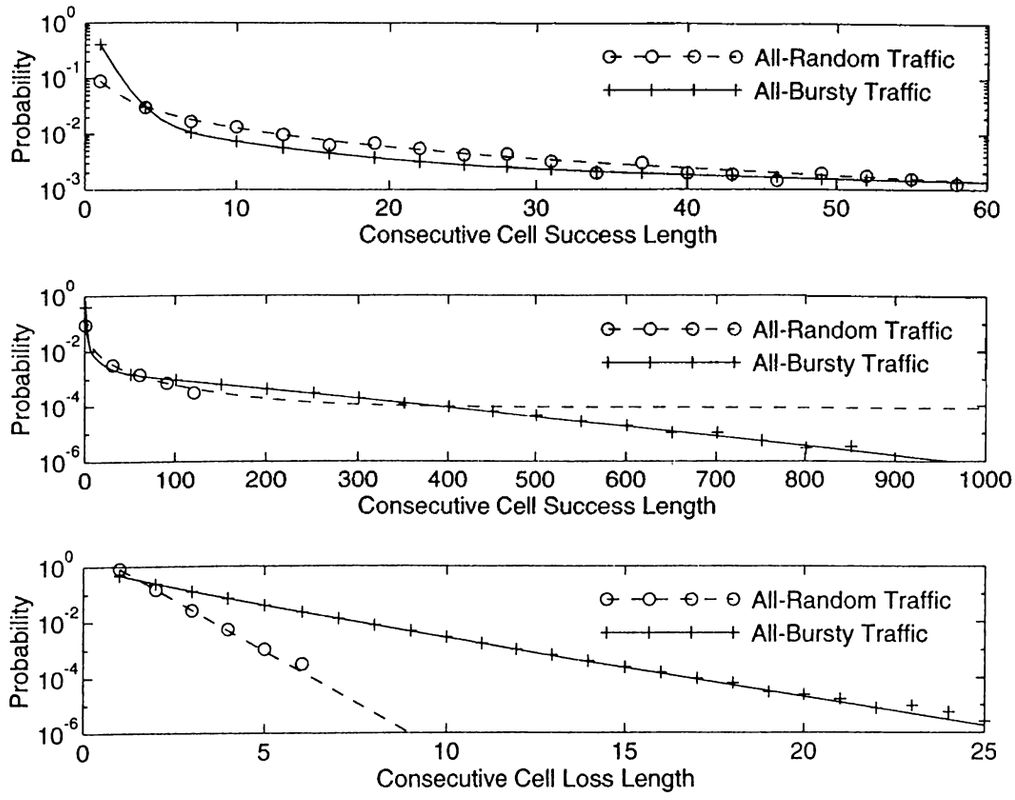

Figure 3.6: The consecutive cell success and loss length distributions for diversely random and bursty traffic scenarios.

3.3.1 Reduced-Complexity Cell Loss Model

The simplest MC model for a bursty cell loss process is the GLM, which has only one "cell success" and one "cell loss" state. Let the *Success Length Distribution* (SLD) be defined as the probability distribution for the number, l, of consecutive successful cells between two lost ones, where $l \geq 1$. Also let the *Loss Length Distribution* (LLD) be defined analogously. Then, the GLM would be an accurate cell loss model only if the SLD and LLD were both geometric.

Figure 3.6 illustrates the typical SLDs and LLDs for two different traffic scenarios. While obtaining these distributions, we set the multiplexer parameters as $N = 16$ and $B = 100$, and assume that there is no coding. For both scenarios,

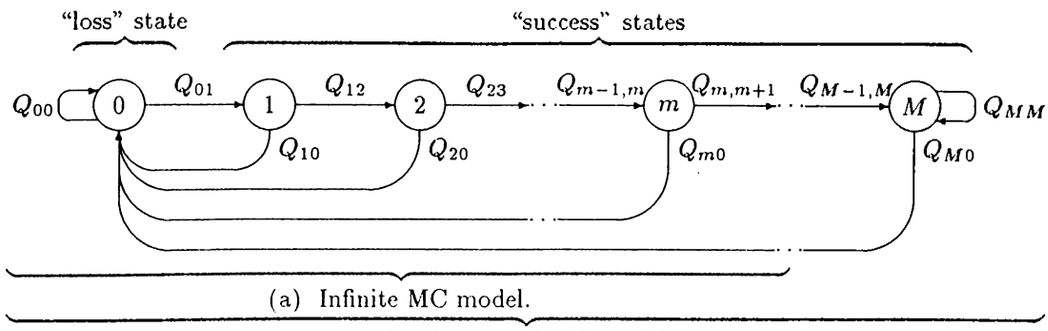

Figure 3.7: An alternative MC model for the tagged cell loss process.

the normalized loads offered by the tagged and untagged sources are $\rho_t = 0.075$ and $\rho_u = 0.06$, respectively. The clustering coefficients of all the sources are equal to $c = 1.2$ for the all-random scenario, and $c = 1.9$ for the all-bursty scenario. We refer to these scenarios as "all-random" and "all-bursty" based on the respective random and bursty characteristics of the sources with $c = 1.2$ and $c = 1.9$, as illustrated in Figure 3.3. The discrete points marked with 'o' and '+' are obtained by event-driven simulations that last for 2×10^8 aggregate tagged and untagged source state transitions. The continuous dashed and solid curves are obtained by computations based on the extended 3-D MC model as will be discussed in the sequel.

Observe that the LLDs of Figure 3.6 can be said to be geometric since they can be approximated well by straight lines in the semilogarithmic scale. Therefore, one "loss" state of the GLM suffices to characterize the consecutive cell loss behavior. However, the same is not true for the SLDs, and one definitely needs multiple "success" states to better reflect the non-geometric behavior of the SLDs.

As an alternative to the GLM, consider the infinite MC of Figure 3.7(a), where state 0 is the unique "loss" state denoting an individual tagged cell loss, and states $m \geq 1$ are the "success" states denoting m consecutive tagged cell successes following a loss. Then, by definition, there are only two possible transitions out from any state, occuring at the arrival of a new tagged cell. When we are

currently in state $m \geq 0$, the next state can be either $m+1$ or 0: $m+1$ if the new tagged cell is successful (move to the next "success" state in the sequence), and 0 otherwise (go back to or stay in "loss" state). That is, defining $Q_{mm'}$ as the probability of transition from state m to state m', we have $Q_{mm'} > 0$ only if $m' = m+1$ or $m' = 0$. Consequently, the SLD and LLD respectively follow from the infinite MC model as

$$P_S(l) = \left(\prod_{m=1}^{l-1} Q_{m,m+1} \right) Q_{l0}\,; \; l \geq 1, \tag{3.26}$$

and

$$P_L(l) = Q_{00}^{l-1} Q_{01}\,; \; l \geq 1. \tag{3.27}$$

Observe that the LLD of (3.27) is geometric as desired. On the other hand, the SLD expression (3.26) yields a general non-increasing probability distribution such as the typical SLDs of Figure 3.6. Therefore, the non-zero state transition probabilities of the infinite MC can be adjusted so that the expressions (3.26) and (3.27) match the characteristics of given SLD and LLD. In fact, the infinite MC is constructed based on the extended 3-D MC as described below, and the match between these expressions and the actual SLD and LLD is achieved by construction.

Let LS^m denote state $m \geq 0$. Then, for any $m \geq 1$, the state transition probability $Q_{m-1,m}$ can be expressed as

$$Q_{m-1,m} = \Pr\left[S \mid LS^{m-1}\right] = \frac{\Pr\left[LS^{m-1}S\right]}{\Pr\left[LS^{m-1}\right]} = \frac{\Pr\left[LS^{m}\right]}{\Pr\left[LS^{m-1}\right]} = \frac{\Pr\left[S^{m} \mid L\right]}{\Pr\left[S^{m-1} \mid L\right]}, \tag{3.28}$$

where $\Pr\left[S^m \mid L\right]$ is the probability of m consecutive successes conditioned to be preceded by a loss. If $g_s(m)$ is the conditional probability that the next m tagged cells are all successful given that the extended 3-D MC is currently in state $s \in \mathcal{S}_E$, $\Pr\left[S^m \mid L\right]$ is given by the weighted average

$$\Pr\left[S^m \mid L\right] = \left(\sum_{s \in \mathcal{S}_L} p_s \right)^{-1} \sum_{s \in \mathcal{S}_L} p_s g_s(m), \tag{3.29}$$

where p_s is the stationary probability of state $s \in \mathcal{S}_E$. Observe that, similar to (3.16), we have the recursive relation

$$g_s(m) = \sum_{s' \in \mathcal{S}_D} q_{ss'} g_{s'}(m) + \sum_{s' \in \mathcal{S}_S} q_{ss'} g_{s'}(m-1) \tag{3.30}$$

for all $s \in \mathcal{S}_E$, where $q_{ss'}$ is the transition probability from state $s \in \mathcal{S}_E$ to $s' \in \mathcal{S}_E$. Then, initializing $g_s(0)$ as unity for all $s \in \mathcal{S}_E$, (3.30) enables iterative computation of $g_s(m)$, and hence, of $Q_{m-1,m}$ via (3.28) and (3.29) for all successive $m \geq 1$.

Now let P_m denote the stationary probability of state $m \geq 0$. Because state 0 represents an individual tagged cell loss, it can be regarded as a merger of all $s \in \mathcal{S}_L$. Therefore, P_0 is given simply by

$$P_0 = \left(\sum_{s \in \mathcal{S}_S \cup \mathcal{S}_L} p_s \right)^{-1} \sum_{s \in \mathcal{S}_L} p_s, \tag{3.31}$$

and P_m for $m \geq 1$ follow immediately in successive iteration cycles by

$$P_m = P_{m-1} Q_{m-1,m} = P_0 \prod_{m'=1}^{m} Q_{m'-1,m'}. \tag{3.32}$$

Obviously, a finite MC cell loss model is preferable to an infinite one since it better suits performance calculations. Observe that the state transition probability $Q_{m,m+1}$ saturates with increasing m, as the geometric tails of the SLDs of Figure 3.6 indicate. This is because the farther we go away from the "loss" state, the less relevant the information given by the exact count of preceding successful cells is. Consequently, the "success" states $m \geq 1$ tend to denote an individual success for large m, just as the "loss" state 0 does; and, the infinite MC can be truncated at a sufficiently deep "success" state M, as shown in Figure 3.7(b).

If the infinite MC is to be truncated at the "success" state M, then P_M must be modified as

$$P_M = 1 - \sum_{m'=0}^{M-1} P_{m'} \tag{3.33}$$

so that state M properly represents all the "success" states $m' \geq M$. In addition, the transition probability from state M to itself must be found from the balance

Algorithm 3.3: Iterative construction of the truncated MC.

(1) Initialization: Set $m := 1$ and $g_s(0) := 1$ for all $s \in \mathcal{S}_E$.

(2) Find $g_s(m)$ for $s \in \mathcal{S}_D$ by using (3.30) and the results of the previous iteration cycle.

(3) Compute $g_s(m)$ for $s \in \mathcal{S}_S \cup \mathcal{S}_L$ by (3.30) using the results of step *(2)* and the previous iteration cycle.

(4) Compute $\Pr[S^m \mid L]$ by (3.29).

(5) Compute $Q_{m-1,m}$ by (3.28).

(6) Compute P_m by (3.32).

(7) Check if state m is a proper truncation state: Let $M = m$.

(7a) Compute P_M by (3.33).

(7b) Find Q_{MM} from (3.34).

(7c) If (3.35) is satisfied, stop.

(8) Set $m := m + 1$, and go to step *(2)*.

equation

$$P_{M-1}Q_{M-1,M} + P_M Q_{MM} = P_M \tag{3.34}$$

so that the state probabilities that have already been found constitute the stationary state probability distribution for the truncated MC. The truncation criterion we use is

$$|Q_{MM} - Q_{M-1,M}| < \epsilon Q_{M-1,M}, \tag{3.35}$$

where ϵ is a small number. Obviously, the smaller ϵ is, the larger but the more accurate the truncated MC model becomes.

Algorithm 3.3 summarizes iterative construction of the truncated MC. Similar to the case for Algorithm 3.1, its complexity is determined by step *(2)*, in which a matrix equation of the form $A\overline{x} = \overline{y}$ is solved for the vector $\overline{x}$ of $|\mathcal{S}_D|$ unknowns. Observe that construction of an $(M+1)$-state truncated MC requires M such solutions. Given ϵ, M depends on the traffic characteristics. For example, with $\epsilon = 10^{-8}$, we have $M = 826$ and $M = 222$ for the all-random and all-bursty traffic scenarios of Figure 3.6, respectively.

Finally note that, after the infinite MC is truncated, (3.26) is modified as

$$P_S(l) = \begin{cases} \left(\prod_{m=1}^{l-1} Q_{m,m+1}\right) Q_{l0} & \text{if } 1 \leq l \leq M, \\ \left(\prod_{m=1}^{M-1} Q_{m,m+1}\right) \left(\prod_{m=M}^{l-1} Q_{MM}\right) Q_{M0} & \text{if } l > M. \end{cases} \tag{3.36}$$

The LLDs and SLDs of Figure 3.6 are found respectively by (3.27) and (3.36). The highly close match between these distributions and those obtained by event-driven simulations indicate that the extended 3-D and the truncated MC cell loss models are both very accurate.

3.3.2 Quantification of the CLR

Now we are ready to compute the CLR for the case of column-only coding with independent column decoding. The CLR in this case is defined as the ratio of the expected number of unrecoverable losses in a column to the column size n'. Similar to (3.15), if $\left\{F^{(n)}(v,n') \,:\, 0 \leq v \leq n'\right\}$ is the CLPD over a column, where $F^{(n)}(v,n')$ is the probability of having v losses in a column, we have

$$P_{l,coc} = \frac{1}{n'} \sum_{v=h'+1}^{n'} v F^{(n)}(v,n') \tag{3.37}$$

since up to h' losses per column can be recovered.

The CLPD over a column can be found based on the truncated MC model as follows. Let $f_m^{(r)}(v,w)$ be the conditional probability that there are v losses in a block of w equidistant cells obtained by taking one of every r cells, given that the system is initially in state $0 \leq m \leq M$. The cell loss process over such a block is characterized by a MC whose state transition probability matrix is the r-th power of that for the truncated MC. Let $Q_{mm'}^{(r)}$, $0 \leq m, m' \leq M$, be the elements of this new matrix, or equivalently, the r-step state transition probabilities of the truncated MC. Then, similar to (3.16), we have the recursive relation

$$f_m^{(r)}(v,w) = Q_{m0}^{(r)} f_0^{(r)}(v-1,w-1) + \sum_{m'=1}^{M} Q_{mm'}^{(r)} f_{m'}^{(r)}(v,w-1) \tag{3.38}$$

for all $0 \leq m \leq M$, which enables $f_m^{(r)}(v,n'-1)$, $0 \leq v < n'$, be computed iteratively as illustrated in Algorithm 3.4. By running this algorithm with $r = n$,

Algorithm 3.4: Iterative computation of $f_m^{(r)}(v, n'-1)$.

(1) Initial condition: $f_m^{(r)}(0,0) = 1$ for all $0 \leq m \leq M$.

(2) Boundary condition: $f_m^{(r)}(v,w) = 0$ for all $0 \leq m \leq M$ if $v < 0$ or $v > w$.

(3) Set $w := 1$.

(4) For $v = 0, 1, 2, \ldots, w$, compute $f_m^{(r)}(v,w)$ by (3.38) using the results of the previous iteration cycle.

(5) If $w < n'-1$, set $w := w+1$, and go to step *(4)*; otherwise, stop.

the CLPD over a column follows by the weighted average[3]

$$F^{(n)}(v,n') = P_0 f_0^{(n)}(v-1, n'-1) + \sum_{m=1}^{M} P_m f_m^{(n)}(v, n'-1). \tag{3.39}$$

Observe that, when we run Algorithm 3.4 with $r = 1$, and evaluate (3.39) with $n' = n$ and $n = 1$, we get the CLPD over a row. In this case, we have only two non-zero terms in (3.38). Also because there are no unknowns to solve, Algorithm 3.4 is of negligible complexity as compared to Algorithm 3.1, which requires a matrix equation of the form $A\overline{x} = \overline{y}$ be solved for the vector $\overline{x}$ of $|\mathcal{S}_D|$ unknowns $n(n+1)/2 - 1$ times. Therefore, the CLPD over a row, and hence, the CLR for row-only codes can be found based on the truncated MC without any practical limit imposed on the row size n.

Because the n-th power of the state transition probability matrix for the truncated MC has fill-ins, the computational complexity of Algorithm 3.4 is higher if it is run with $r = n$ rather than with $r = 1$. Yet, it can be shown that the number of next states reachable from any given state in r transitions is $r+1$. Therefore, while finding the CLPD over a column, the number of non-zero terms in (3.38) is only $n+1$, and the CLR computation for column-only codes is of tractable complexity unless n is too large[4].

[3]Similar to the cases in (3.17) and (3.24), we have $f_0'(-1, n'-1) = 0$ and $f_m'(n', n'-1) = 0$ for $1 \leq m \leq M$ in (3.39).

[4]In fact, for cases in which the truncated MC size $M+1$ is not very large, dealing even with

For the final case of joint coding with the optimal decoder, the CLR is the ratio of the expected number of unrecoverable losses in a coding matrix left after optimal decoding to the coding matrix size nn'. Because its computation is very difficult due to the interdependence of successive row and column decoding phases, we rely on simulations in this case. However, unlike the case for row- and column-only coding techniques, unrecoverable cell losses exhibit a more bursty nature for practical joint codes, and it requires longer simulation runs to measure the same CLR with the same confidence. Owing to the special structure of the truncated MC, we perform analysis-driven simulations. That is, ignoring the tails that account to negligible cumulative probabilities, we first approximate the LLD and SLD of (3.27) and (3.36) by finite probability distributions. Then, by using the alias method which extracts an outcome from a finite set efficiently by just one pseudo-random number generator call [17], we determine the lengths of successive cell loss and success periods. In this manner, we quickly proceed on the tagged cell stream batch-by-batch rather than cell-by-cell or slot-by-slot. This approach is found to provide one to two orders of magnitude speed-up depending on the cell loss process burstiness.

3.A Computation of the GLM Parameters

Let $p_{s|rs}$ be the probability that the extended 3-D system enters state $s \in \mathcal{S}_S \cup \mathcal{S}_L$ upon arrival of the last cell of a row, given that the row is successful. Similarly let $p_{s|rl}$ be the probability of the same event conditioned on the loss of the row.

Observe that, given the success or loss of a row, the conditional state probability distribution for the last cell of the row looking forward in time is equal to that for the first cell looking backward in time. Therefore, to compute $p_{s|rs}$ and $p_{s|rl}$, $s \in \mathcal{S}_S \cup \mathcal{S}_L$, we reverse the extended 3-D MC in time. That is, recalling that $q_{ss'}$ is the transition probability from state $s \in \mathcal{S}_E$ to $s' \in \mathcal{S}_E$, we

a full state transition probability matrix is feasible. For example, we have $M = 222$ for the all-bursty traffic scenario of Figure 3.6, and various column-only codes with n as large as 256 are studied for this scenario in Section 4.2.

consider the MC characterized by the state transition probabilities

$$\overline{q}_{ss'} = p_{s'} q_{s's} / p_s, \tag{3.40}$$

where p_s is the stationary probability of state $s \in \mathcal{S}_E$ for both MCs [17].

Now let $\overline{f}_s(v, w)$, $v = 0, 1, 2, \ldots, w$, be the conditional probability that v of previous w tagged cells are lost given that the time-reversed system is initially in state $s \in \mathcal{S}_E$. Suppose that $\overline{f}_s(v, n-1)$, $v = 0, 1, 2, \ldots, n-1$ are computed by Algorithm 3.1 with the state transition probabilities $q_{ss'}$ replaced by $\overline{q}_{ss'}$ of (3.40). Then, the conditional state probabilities $p_{s|rs}$ and $p_{s|rl}$, $s \in \mathcal{S}_S \cup \mathcal{S}_L$, for the last cell of a row follow from the Bayes' rule as

$$p_{s|rs} = \frac{p_s}{\rho_{et} P_{rs}} \sum_{v=0}^{h} \overline{f}_s(v, n-1) \text{ for } s \in \mathcal{S}_S, \tag{3.41}$$

$$p_{s|rs} = \frac{p_s}{\rho_{et} P_{rs}} \sum_{v=0}^{h-1} \overline{f}_s(v, n-1) \text{ for } s \in \mathcal{S}_L, \tag{3.42}$$

$$p_{s|rl} = \frac{p_s}{\rho_{et} P_{rl}} \sum_{v=h+1}^{n-1} \overline{f}_s(v, n-1) \text{ for } s \in \mathcal{S}_S, \tag{3.43}$$

and

$$p_{s|rl} = \frac{p_s}{\rho_{et} P_{rl}} \sum_{v=h}^{n-1} \overline{f}_s(v, n-1) \text{ for } s \in \mathcal{S}_L, \tag{3.44}$$

where ρ_{et} is the normalized effective tagged load given by (3.3), and P_{rs} and P_{rl} are respectively the row success and loss probabilities, which follow from the CLPD over a row by

$$P_{rs} = 1 - P_{rl} = \sum_{v=0}^{h} F(v, n). \tag{3.45}$$

Although $\overline{f}_s(v, w) \neq f_s(v, w)$, reversing the extended 3-D MC in time will not change the CLPD. Then, it can be shown from (3.17) and (3.45) that

$$P_{rs} = \frac{1}{\rho_{et}} \left(\sum_{s \in \mathcal{S}_S} p_s \sum_{v=0}^{h} \overline{f}_s(v, n-1) + \sum_{s \in \mathcal{S}_L} p_s \sum_{v=0}^{h-1} \overline{f}_s(v, n-1) \right). \tag{3.46}$$

Consequently, (3.41), (3.42), and (3.46) imply that

$$\sum_{s \in \mathcal{S}_S \cup \mathcal{S}_L} p_{s|rs} = 1,$$

or equivalently, $p_{s|rs}$, $s \in \mathcal{S}_S \cup \mathcal{S}_L$, is a probability distribution as desired. It can similarly be shown that $p_{s|rl}$, $s \in \mathcal{S}_S \cup \mathcal{S}_L$, is a probability distribution, too.

Computing $p_{s|rs}$ and $p_{s|rl}$, $s \in \mathcal{S}_S \cup \mathcal{S}_L$, as described above, we can condition the CLPD over a row on the success and loss of the previous row as follows:

$$F(v, n \mid RS) = \sum_{s \in \mathcal{S}_S \cup \mathcal{S}_L} p_{s|rs} f_s(v, n), \tag{3.47}$$

and

$$F(v, n \mid RL) = \sum_{s \in \mathcal{S}_S \cup \mathcal{S}_L} p_{s|rl} f_s(v, n), \tag{3.48}$$

where $v = 0, 1, 2 \ldots, n$, and $f_s(v, n)$, $s \in \mathcal{S}_S \cup \mathcal{S}_L$, are computed by Algorithm 3.1 based on the original 3-D MC. The GLM parameters Q and q are then given by

$$Q = \sum_{v=0}^{h} F(v, n \mid RS), \tag{3.49}$$

and

$$q = \sum_{v=h+1}^{n} F(v, n \mid RL). \tag{3.50}$$

Chapter 4

Performance of Two-Level FEC

In this chapter, we present and discuss the performance results obtained based on the analytical framework described in Chapter 3. We fix the number of sources as $N = 16$ to obtain a reasonable traffic mix, and study the row-only, column-only, and joint coding techniques for a buffer capacity of $B = 100$ cells. In addition to demonstrating the effectiveness of FEC, we want to understand the mechanisms effecting the coding performance, and particularly want to show that joint coding is useful. To this end, we compare the performances of the three coding techniques for various traffic scenarios whose load and burstiness parameters are given in Table 4.1 together with the CLRs for the no coding case.

Recalling the extremely diverse characteristics of the sources with clustering coefficients 1.2 and 1.9 illustrated in Figure 3.3, we respectively refer to these generic sources as "random" and "bursty" sources, and denote the traffic scenario with $c_t = 1.2$ and $c_u = 1.9$, for example, by RB15 which stands for "random tagged traffic interfering with 15 bursty untagged streams." Similarly, RR15 and BB15 respectively denote the cases in which all the 16 sources are "random" and "bursty," whereas the tagged source is "bursty" and the untagged sources are "random" in the BR15 scenario[1]. On the other hand, UR16 and UB16 stand for the "uniform" traffic scenarios with $\rho_t = \rho_u$ where all the 16 sources are "random"

[1]Observe that the RR15 and BB15 scenarios are respectively the all-random and all-bursty traffic scenarios of Figure 3.6.

Traffic Parameters				CLR for the No Coding Case	Traffic Scenario
ρ_t	ρ_u	c_t	c_u		
0.06	0.06	1.2	1.2	1.216×10^{-4}	UR16
0.05	0.05	1.9	1.9	1.412×10^{-2}	UB16
0.075	0.06	1.2	1.2	6.787×10^{-4}	RR15
			1.9	3.752×10^{-2}	RB15
		1.9	1.2	1.337×10^{-2}	BR15
			1.9	5.858×10^{-2}	BB15

Table 4.1: The traffic scenarios and the corresponding CLRs for the no coding case. The traffic parameters ρ_t, ρ_u, c_t, and c_u are the normalized loads offered by and the clustering coefficients for the tagged and untagged sources, respectively.

and "bursty," respectively.

Observe that the load parameters ρ_t and ρ_u of Table 4.1 add up to very high normalized aggregate loads, $\rho = \rho_t + 15\rho_u$. We have $\rho = 0.96$ and 0.8 respectively for the UR16 and UB16 scenarios, and $\rho = 0.975$ for the RR15, RB15, BR15, and BB15 scenarios. Although such high loads may not be typical for the long-term behavior of an ATM network, our steady-state analysis thus gives emphasis to transient heavy traffic periods during which congestion is more likely to occur, and hence, FEC is functional.

In the following, assuming that only the tagged traffic is coded, we evaluate the coding performance for the constrained and optimal decoding cases. Since row-only codes are decoded in the same manner in both cases, we first study these codes separately from the others in Section 4.1. Then, we study the constrained and optimal decoding of column-only and joint codes in Sections 4.2 and 4.3, respectively. The results are discussed in Section 4.4.

Because we will be referring to various codes of each type quite often, we introduce the following notation to indicate the coding parameters: *(i)* (n, h) for row-only codes, *(ii)* $[n, (n', h')]$ for column-only codes, and *(iii)* $[(n, h), (n', h')]$ for joint codes, where n and n' are the row and column sizes, and h and h' are the numbers of parity cells per row and column, respectively.

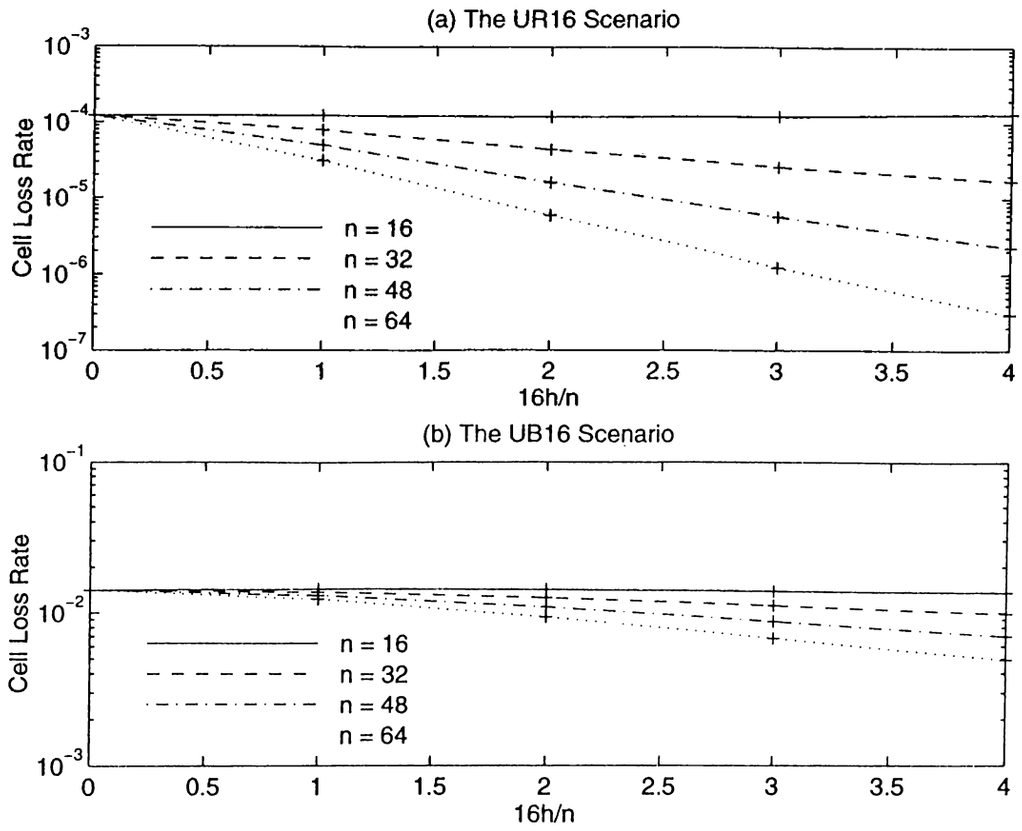

Figure 4.1: Effects of n and h/n on the row-only coding performance for the UR16 and UB16 scenarios, where n is the row size and h is the number of parity cells in a row.

4.1 Row-Only Coding Results

We study the performance of row-only coding first for the UR16 and UB16 scenarios. For both scenarios, we consider the row sizes $n \in \{16, 32, 48, 64\}$, and constrain the parity cells to constitute at most 25% of the total tagged traffic, i.e., $h/n \leq 0.25$. The CLR results are shown in Figure 4.1.

It is observed from Figure 4.1(a) that, for the UR16 scenario, row-only coding with $n = 16$ does not improve the CLR for any $h \leq 4$. Referring to a row with one or more losses as being "corrupted," this is because a significant fraction of the corrupted 16-cell rows lose more than 4 cells. Since the most powerful code among

those with a given rate is the one that has the largest block length, the row-only coding gain increases uniformly with n for all $0 < h/n \leq 0.25$. In particular, the $(64, 16)$ code provides a CLR reduction of about 2.6 orders of magnitude as compared to the no coding case. In Figure 4.1(b), we observe similar trends for the UB16 scenario. However, since the cell loss process burstiness increases with the traffic burstiness, row-only coding in this case is much less effective than it is for the UR16 scenario. Obviously, better results can be achieved if row sizes larger than 64 are considered. Fixing $h/n = 0.25$, we next search for this possibility.

Figure 4.2 illustrates the CLR for $(n, n/4)$ row-only coding as a function of the row size n for the RR15, RB15, BR15, and BB15 scenarios. Observe that this coding technique is quite effective when the tagged traffic is random. For example, the moderate row size of 256 suffices to achieve a CLR well below 10^{-10} for the RR15 scenario. Although the CLR then appears in the order of 10^{-5} for the RB15 scenario, this accounts to more than 3 orders of magnitude reduction as compared to the no coding case, and hence, is still a significant achievement as the heavy load conditions are taken into account. Also, with the normalized tagged load of 0.075, the average decoding latency for the $(256, 64)$ code is fairly small. That is, assuming that the data transmission rate is 155.52 Mbps, even the first cell of a row, if lost, will wait 6.98 ms on the average for a possible recovery.

For the bursty tagged traffic case, however, row-only coding is not effective unless the row size is too large. For example, in order to guarantee a CLR of 10^{-5}, the row size must be roughly 500 and 900 for the BR15 and BB15 scenarios, respectively. Since codes with large block lengths have high decoding complexities, code interleaving which mitigates the effect of burst losses here appears as a good means of compromise to achieve a desired CLR with limited decoding complexity but with increased decoding latency. We will study column coding which serves this purpose in the sequel. First, we discuss Figure 4.2 more as it provides valuable insight into how the cell loss process is influenced by the traffic characteristics.

It is known that the more bursty the traffic streams are, not only the higher the average loss rate becomes, but also the more likely the losses are to occur in

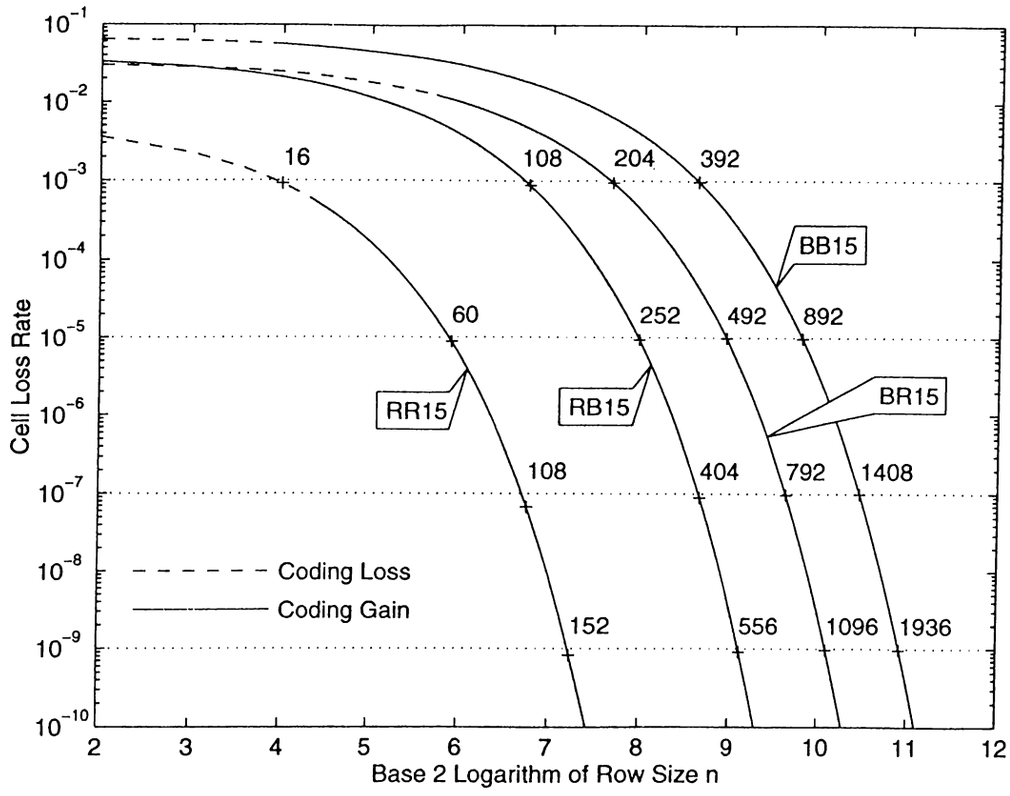

Figure 4.2: The CLR for $(n, n/4)$ row-only coding as a function of $\log_2 n$ for the RR15, RB15, BR15, and BB15 scenarios, where n is the row size. The smallest row sizes that guarantee the CLRs of 10^{-3}, 10^{-5}, 10^{-7}, and 10^{-9} are also indicated for each traffic scenario.

bursts [5], [6]. The difference between the RB15 and BR15 curves of Figure 4.2 is particularly noteworthy in this regard. Observe that the CLRs for the RB15 and BR15 scenarios do not differ much for the no coding case, that for the former being even slightly larger as Table 4.1 indicates. However, row-only coding performs far better for the RB15 scenario than for the BR15 scenario. The burstiness of the tagged cell losses should then be the dominant factor determining the coding performance here, and not their average rate. This difference further indicates that the tagged cell loss process burstiness depends to a greater extent on the burstiness of tagged traffic than on that of untagged traffic. This is because the inter-arrival time distribution of tagged cells has a *direct* effect on the tagged cell

loss process as opposed to that of untagged ones. The more likely the tagged cells are to arrive in clusters, the stronger the correlation between the buffer states seen by successive tagged cells is, without being affected much by the untagged traffic characteristics. Conversely, if the tagged arrivals are dispersed in time, this correlation will be definitely weaker but relatively less immune to the indirect effect of untagged traffic characteristics.

4.2 Constrained Decoding Case

In this section, focusing on the UR16 and UB16 scenarios, we study the column-only and joint coding techniques for the case of constrained decoding. The column-only and joint coding results are discussed separately first. Then, we also compare the performances of the two coding techniques for both traffic scenarios at different load values.

4.2.1 Column-Only Coding Results

Observe that the row size n has two opposite effects on the performance of column-only coding with the constrained decoder. Since loss of even one cell results in the loss of the row that it belongs to, the row loss process is adversely affected by increasing n. On the other hand, too small an n may not provide sufficient code interleaving effect desired to overcome the burstiness of cell losses. As our preliminary results indicate that $n = 16$ is a reasonable choice in these two respects, we study $[16, (n', h')]$ codes in the following.

For the UR16 scenario, we consider the column sizes $n' \in \{16, 32, 48, 64\}$. The CLR results obtained for $h'/n' \leq 0.25$ are shown in Figure 4.3(a). Comparing these results with those of Figure 4.1(a), it is observed that column-only coding with $n' = 16$ improves the CLR slightly in contrast with the case for row-only coding with $n = 16$. This indicates that although the traffic is highly random in the UR16 scenario, cell losses are still bursty, and code interleaving provides an improvement. This is observed much more clearly for larger n'. For example,

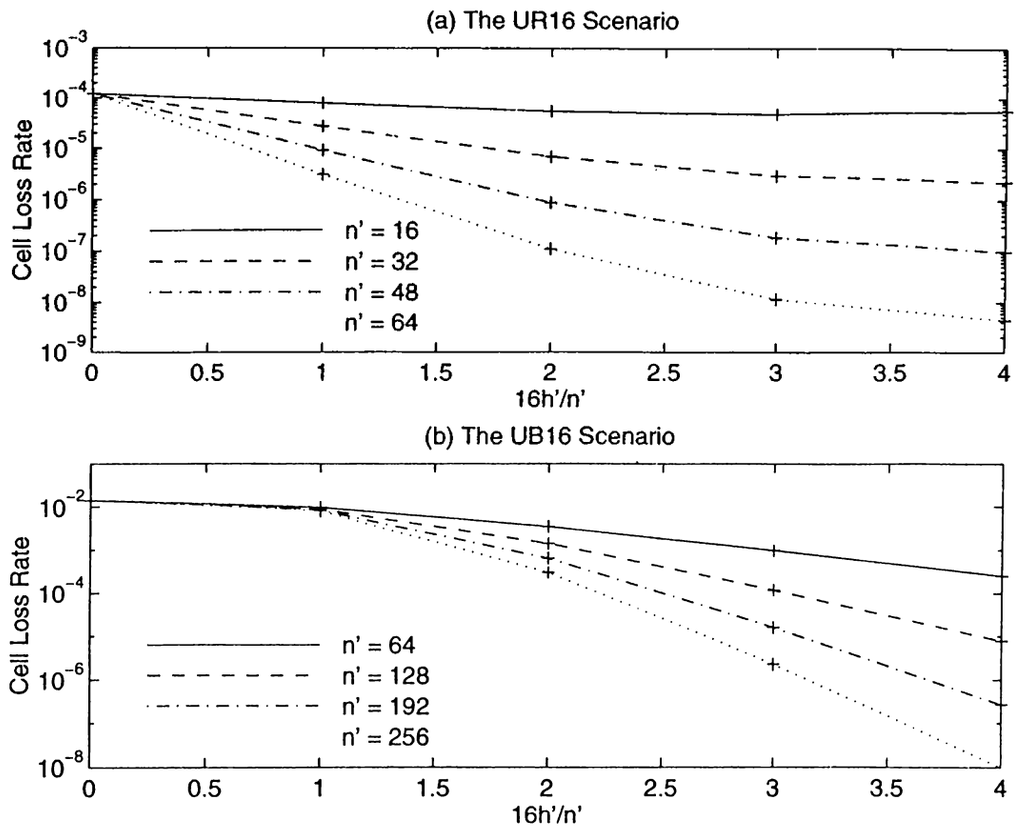

Figure 4.3: Effects of n' and h'/n' on the performance of column-only coding with the constrained decoder for the UR16 and UB16 scenarios, where n' is the column size, h' is the number of parity cells in a column. The row size n is fixed as 16 for both scenarios.

the $[16, (64, 16)]$ code outperforms the $(64, 16)$ code by providing nearly 2 orders of magnitude additional reduction in the CLR. From another point of view, the $[16, (64, 8)]$ and $[16, (48, 9)]$ codes both introduce less parity overhead, but still achieve smaller CLRs as compared to the $(64, 16)$ code.

Also observe that the column-only coding gain saturates with increasing h'/n' for the UR16 scenario. In fact, when $n' = 16$, there is a relative worsening for $h' = 4$ as compared to $h' = 3$. Referring to a coding matrix with one or more lost rows as being "corrupted," this saturation indicates that a significant fraction of the corrupted coding matrices arrive with no more than $0.25n'$ lost rows. For

$n' = 32$ or larger, if h'/n' is increased further from 0.25, the increase in the tagged load due to the parity overhead will start to dominate, and a relative worsening will occur as in the case of $n' = 16$ and $h' = 4$.

The column-only coding results for the UB16 scenario are shown similarly in Figure 4.3(b). Due to the heavily bursty nature of cell losses, $n' < 64$ does not yield significant improvements in this case, and we consider column-only codes with $n' \in \{64, 128, 192, 256\}$. As expected, column-only coding is quite effective for this bursty traffic scenario. In fact, contrary to the saturation behavior observed for the UR16 scenario, it becomes more and more effective with increasing $h'/n' > 0.0625$. This difference between the UR16 and UB16 scenarios is reasonable. The more bursty the traffic is, the longer a congested period will last for to corrupt several consecutive rows. Consequently, given n and n', the column-only coding gain may increase with h'/n' if the traffic is bursty, whereas it saturates otherwise.

Finally, note that Figure 4.3 indicates significant column-only coding gains for both UR16 and UB16 scenarios. These gains are achieved despite constrained decoding. Much better results should be expected when we switch to optimal decoding in which the columns of the coding matrix are decoded independently.

4.2.2 Joint Coding Results

For the case of joint coding, we fix the column size n' as 64 for the UR16 scenario and as 256 for the UB16 scenario. Also fixing the row size n as 16 for both scenarios, we investigate the effects of h and h' on the joint coding performance, where the total number of parity cells in the coding matrix is constrained to be less than $0.3nn'$. The results for the UR16 and UB16 scenarios are shown in Figures 4.4(a) and 4.4(b), respectively.

Recall from Figure 4.1(a) that row coding with $n = 16$ alone is not effective for any $h \leq 4$ for the UR16 scenario. However, Figure 4.4(a) indicates that it provides a significant improvement when used together with column coding. In particular, when $n' = 64$ and $h' = 16$, even the simple single-parity row coding with $n = 16$

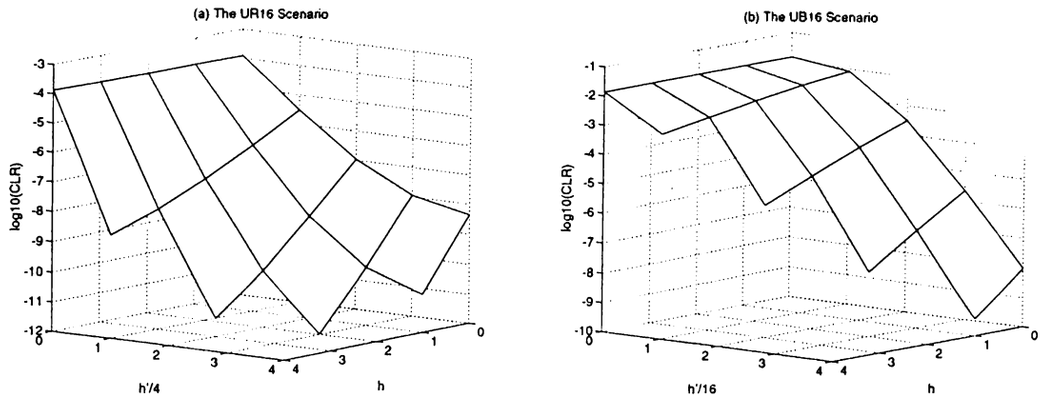

Figure 4.4: Effects of h and h' on the performance of joint coding with the constrained decoder for the UR16 and UB16 scenarios, where h and h' are the numbers of parity cells per row and column, respectively. The row size n is fixed as 16 for both scenarios, and the column size n' is 64 for the UR16 scenario and 256 for the UB16 scenario.

provides approximately 2.3 orders of magnitude additional reduction in the CLR as compared to column coding alone. With the use of constrained decoder in mind, this is because row coding improves the row loss process characteristics, and in turn, increases the effectiveness of column coding considerably. In fact, even the $[(16,1),(64,12)]$ code is superior to the $[16,(64,16)]$ code as it achieves a smaller CLR with less parity overhead.

In Figure 4.4(b), a similar trend is observed for the UB16 scenario. However, the additional gain of coding rows in this case is not as significant as it is for the UR16 scenario. Nevertheless, despite the heavily bursty nature of cell losses, the $[(16,1),(256,64)]$ code provides an additional gain of approximately 1.4 orders of magnitude as compared to the $[16,(256,64)]$ code.

The CLRs for the no coding case, and for the $[16,(64,16)]$ and $[(16,1),(64,12)]$ codes are plotted as functions of the normalized aggregate load for the UR16 scenario in Figure 4.5(a). Observe that the joint code is superior to the column-only code over the entire load range. The performances of the $[16,(256,64)]$ and $[(16,1),(256,48)]$ codes are compared similarly for the UB16 scenario in Figure 4.5(b). Unlike the case for the UR16 scenario, the column-only code is

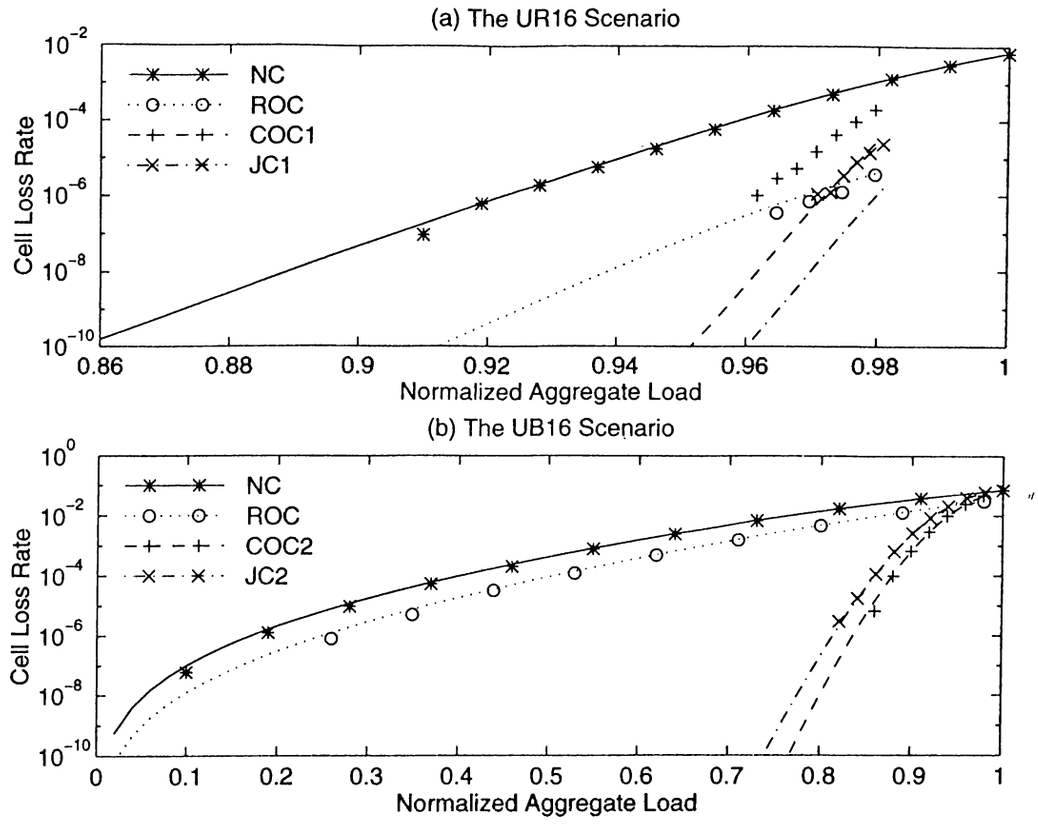

Figure 4.5: Coding performance with the constrained decoder for the UR16 and UB16 scenarios as a function of the load. Different loads are obtained by keeping $\rho_t = \rho_u$ and varying both. The legend NC corresponds to "no coding," ROC to "the $(64, 16)$ code," COC1 to "the $[16, (64, 16)]$ code," COC2 to "the $[16, (256, 64)]$ code," JC1 to "the $[(16, 1), (64, 12)]$ code," and JC2 to "the $[(16, 1), (256, 48)]$ code." The discrete points are the results obtained by event-driven simulations.

superior in this case. This is because, due to the heavily bursty nature of cell losses, the probability of having cell losses scattered over the coding matrix is smaller for the UB16 scenario than it is for the UR16 scenario. Therefore, given the coding matrix dimensions and the code rate, it is better to devote all the parity cells to code columns[2].

[2]Observe that the rates, $kk'/nn' = (1 - h/n)(1 - h'/n')$, of the codes compared in Figure 4.5 are approximately the same. The column-only codes are exactly of rate 3/4, and the joint codes are of rate $0.762 > 3/4$. Therefore, these comparisons are fair, and do not favor joint coding.

The CLR for the $(64, 16)$ code is also plotted in Figure 4.5. Observe that this row-only code is not effective for the UB16 scenario even at very low loads. For the UR16 scenario, on the other hand, it provides a CLR reduction of about 2.5 orders of magnitude over the entire load range. Although the column-only and joint codes with $n = 16$ and $n' = 64$ achieve much lower CLRs with decreasing load in this case, it should be noted that they have 16 times larger decoding latency as compared to the $(64, 16)$ code.

We have also simulated transmission of 10^8 tagged cells on a discrete-event basis for both UR16 and UB16 scenarios. The results are shown in comparison with the computational results in Figure 4.5. The almost perfect matches in the no coding case for both scenarios, and in the row-only coding case for the UB16 scenario are because we have not made any simplifying assumptions in these cases (see Section 3.2). The slight deviation in the row-only coding case for the UR16 scenario should be attributed to the fact that the smaller the CLR is, the lower the confidence of a fixed-length simulation becomes.

Recall that the row loss process is approximated by the GLM to compute the CLR for the column coded cases. The GLM claims that the loss and success of a row depends merely on the loss and success of the previous row. The extent to which this claim holds is determined by the burstiness of cell losses and the row size n. Obviously, the larger n is, the less relevant the information about more than one preceding row is. Consequently, the GLM becomes more accurate. The surprisingly perfect match between the computations and simulations in Figure 4.5(b) indicates that $n = 16$ is a sufficiently large row size in this regard for the UB16 scenario. However, this is not the case for the UR16 scenario as can be seen in Figure 4.5(a). To figure out the reason of this mismatch, we ran simulations to obtain the consecutive row loss and success length distributions. For both scenarios, the consecutive row loss behavior is approximated well by a geometric distribution. But, for the UR16 scenario, the consecutive success length distribution for rows is found to be similar to that for cells illustrated in Figure 3.6. Therefore, for an accurate analysis of the constrained decoder for the random traffic case, a row loss process model similar to that of Figure 3.7

can be developed. However, we do not attempt to correct our results in this manner because the computations for the UR16 scenario based on the GLM at least capture the trends and relative performances of the column-only and joint coding techniques.

4.3 Optimal Decoding Case

In the following, we study the column-only and joint coding techniques for the case of optimal decoding. Since row-only coding was shown to be quite effective for the random tagged traffic cases (see Figure 4.2), we focus particularly on the BB15 scenario. We fix the code rate as 3/4, and compare different codes that achieve the same CLR with regard to their decoding latencies and complexities.

4.3.1 Column-Only Coding Results

Figure 4.6 illustrates the performance of $[n, (n', n'/4)]$ column-only coding for the BB15 scenario via constant-CLR curves on the logarithmic nn'-plane. The row size n and the column size n' of the codes that guarantee the CLRs of 10^{-3}, 10^{-5}, 10^{-7}, and 10^{-9} with smallest nn' (decoding latency) and n' (decoding complexity) are also indicated separately in Figure 4.6. The $[4, (248, 62)]$ code, for example, guarantees the CLR of 10^{-5} with smallest possible decoding latency under the constraint that $n' \leq 256$. This accounts to a significant interleaving gain in terms of decoding complexity achieved at the expense of just about 11% increase in the average decoding latency as compared to the $(892, 223)$ code that yields the same CLR. Comparison of Figures 4.2 and 4.6 in this regard indicates that the lower the CLR objective is, the more one should pay in terms of an increase in decoding latency in order to limit the decoding complexity. Yet, even for the CLR of 10^{-9}, this increase is less than 40% as the $[11, (244, 61)]$ and $(1936, 484)$ codes are compared, but the reduction in decoding complexity is then considerably large as the code length drops to 244 from 1936.

Figure 4.6 also indicates that it is possible to reduce the decoding complexity

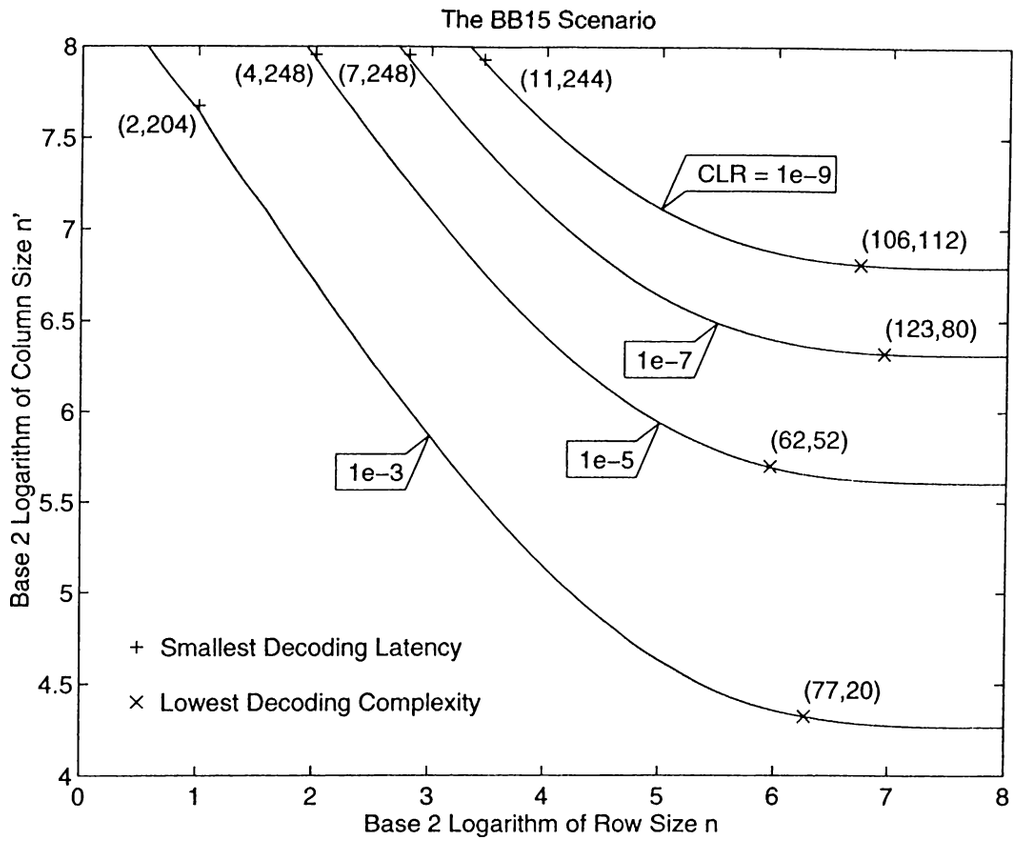

Figure 4.6: The constant-CLR curves for $[n, (n', n'/4)]$ column-only coding on the logarithmic nn'-plane for the BB15 scenario, where n and n' are the row and column sizes, respectively. The integer valued (n, n') pairs which minimize the decoding latency and complexity while guaranteeing the CLRs of 10^{-3}, 10^{-5}, 10^{-7}, and 10^{-9} are indicated separately.

further from that of the smallest-decoding-latency column-only code achieving a given CLR if an additional decoding latency penalty can be tolerated. However, since the cell loss process over a column gets closer to being uncorrelated as the row size is increased, the interleaving gain, and in turn, the column size n' required to guarantee a given CLR saturates with increasing n. For the CLR of 10^{-5}, for example, the $[62, (52, 13)]$ code is the one with lowest possible decoding complexity under the constraint that $n \leq 256$. Reducing n' further to only 48 requires n to be increased by a factor more than 4, leading to an unnecessarily

large increase in decoding latency.

In fact, code interleaving, i.e., column-only coding is in general preferable when we have burst losses. Comparison of Figures 4.2 and 4.6 shows that, for the BB15 scenario, column-only coding is superior to row-only coding with the same rate as it enables the achievement of CLRs as low as, or even lower than, 10^{-5} by reducing the decoding complexity to practical levels with acceptable increases in decoding latency. Joint coding with the optimal decoder has the potential to add to the interleaving gain of column-only coding as we discuss next.

4.3.2 Joint Coding Results

In Figure 4.7, the performance of $[(n, n/8), (n', n'/8)]$ joint coding is compared to that of $[n, (n', n'/4)]$ column-only coding for the BB15 scenario via constant-CLR curves on the logarithmic nn'-plane again. The CLR $= 10^{-5}$ curve and the associated 99% confidence interval for joint coding are obtained by 9 independent analysis-driven simulation runs with 10^9 tagged cells generated in each run. Since unacceptably long simulations are required to obtain appreciably confident results for lower CLRs, we content with the CLR $= 10^{-5}$ results in this comparison.

Figure 4.7 indicates that, for the BB15 scenario, column-only coding far outperforms joint coding when the row size n is small. For example, the $[(16, 2), (256, 32)]$ code yields a CLR of about 10^{-5}, whereas the CLR for the $[16, (256, 64)]$ code is below 10^{-10}. In fact, given n, n', and the code rate, joint coding can be regarded as a variant of column-only coding where part of the parity cells are devoted to coding rows with the hope that the row and column codes will reinforce each other during successive row and column recovery phases. But since even the row-only code with n as small as 16 is totally ineffective as Figures 4.1 and 4.2 indicate, this provides nothing, and the parity cells used for coding rows are simply wasted. On the other hand, as n increases, the interleaving gain and the effectiveness of the row code both increase, and the column size n' required to maintain the CLR of 10^{-5} decreases. However, because the increasing row coding effectiveness balances the decrease in the additional interleaving gain

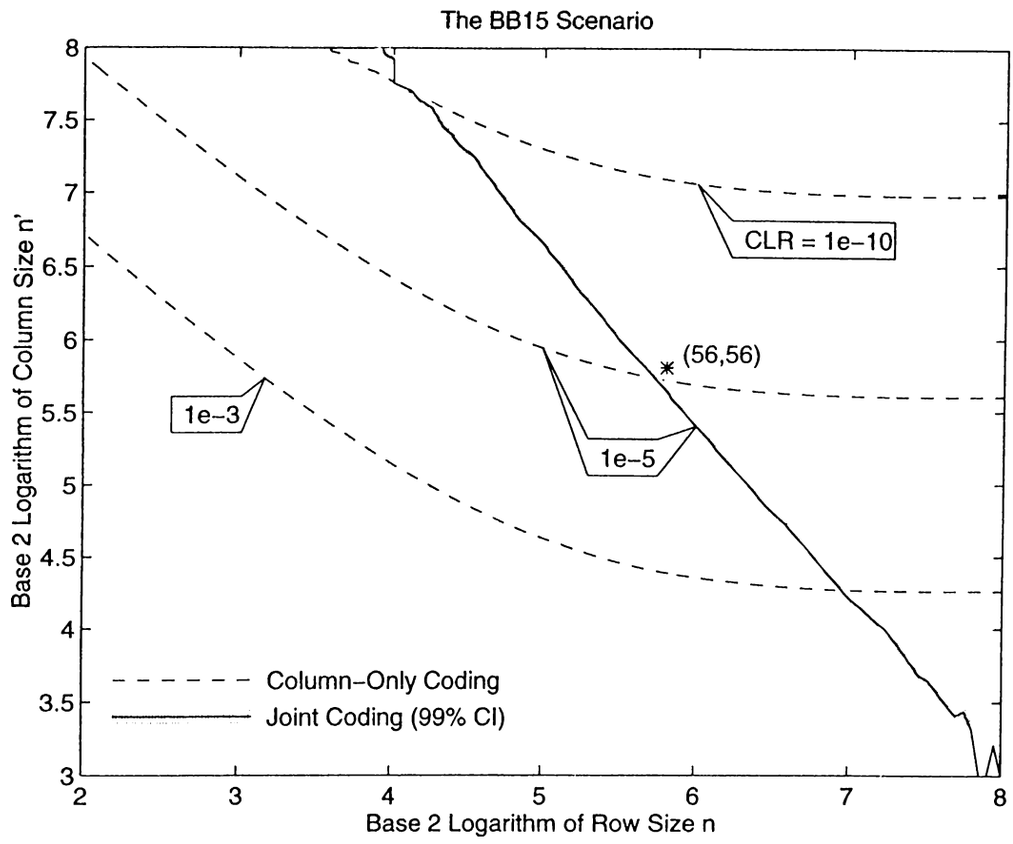

Figure 4.7: Comparison of $[(n, n/8), (n', n'/8)]$ joint and $[n, (n', n'/4)]$ column-only coding techniques via constant-CLR curves on the logarithmic nn'-plane for the BB15 scenario, where n and n' are the row and column sizes, respectively.

with increasing n, this decrease is uniform as opposed to the saturation observed in the case of coding columns only. Therefore, beyond a certain critical point as we move along one of the two CLR $= 10^{-5}$ curves of Figure 4.7 in the direction of increasing n and decreasing n', the joint codes that fall on this curve become superior to the column-only codes with the same n and n'. In particular, when $n = 256$ and $n' = 8$, joint coding provides a substantial improvement over column-only coding by yielding more than 2 orders of magnitude additional CLR reduction. However, the $[4, (256, 64)]$ code, for example, is still superior to the $[(256, 32), (8, 1)]$ code because both achieve roughly the same CLR with

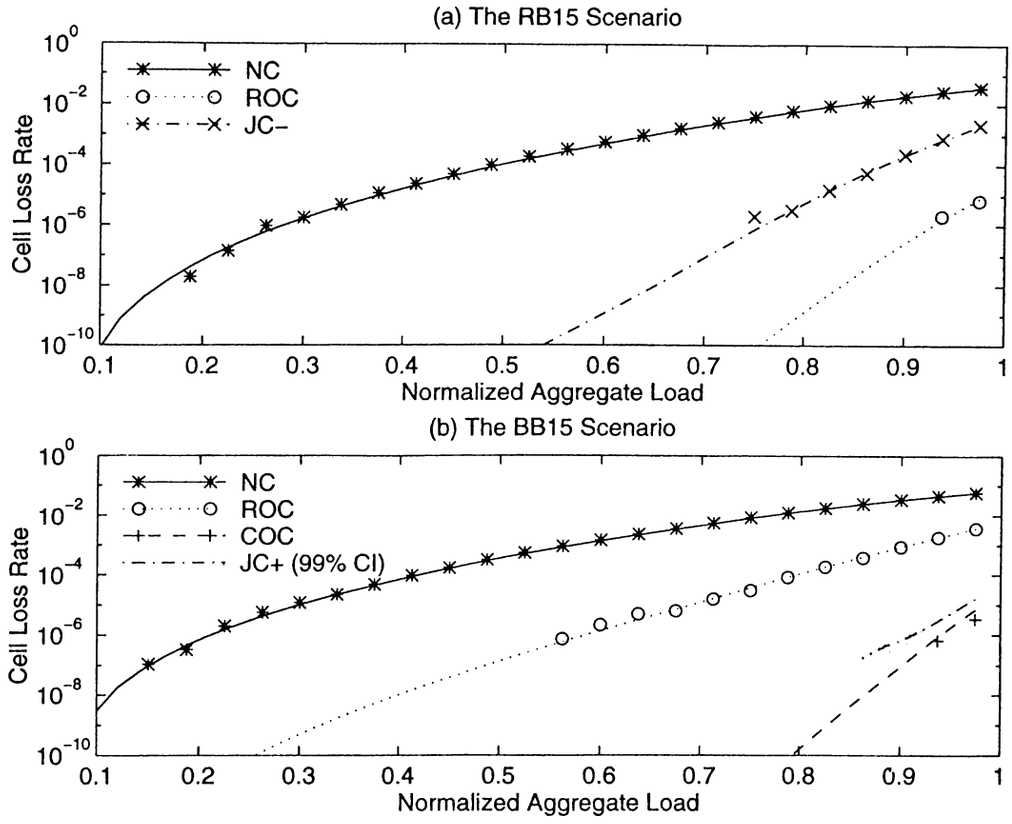

Figure 4.8: Coding performance with the optimal decoder for the RB15 and BB15 scenarios as a function of the load. Different loads are obtained by keeping $\rho_t = 0.075$ unchanged and varying ρ_u only. The legend NC corresponds to "no coding," ROC to "the $(256, 64)$ code," COC to "the $[4, (256, 64)]$ code," JC+ to "the $[(256, 36), (8, 1)]$ code," and JC- to the same joint code with row decoding only. The JC+ results are obtained by 6 independent analysis-driven simulation runs with 10^9 tagged cells generated in each run. The discrete points are the results obtained by event-driven simulations.

comparable decoding complexities, but the decoding latency for the former is half of that for the latter. Yet, a joint code with such large row size deserves special credit since its row code component alone can serve as a fast recovery means for random tagged traffic.

To illustrate this possibility, we consider the $[(256, 36), (8, 1)]$ code, and compare the performance of its row code component to that of the $(256, 64)$

code for the RB15 scenario. The results are shown in Figure 4.8(a) as functions of the normalized aggregate load. Obviously because the row code component has less number of parity cells, it is inferior to the dedicated row-only code. Nevertheless, even for high loads, it provides approximately an order of magnitude CLR reduction as compared to the no coding case. This gain increases as the load decreases, and a 3 orders of magnitude reduction is observed at about the normalized aggregate load of 0.8.

For the BB15 scenario, Figure 4.8(b) similarly compares the performances of the $[4, (256, 64)]$ code and the above joint code, with the optimal decoder this time. Observe that the two codes perform comparably well, and both provide a CLR reduction of more than 4 orders of magnitude over almost the entire load range. The CLR results for the $(256, 64)$ code are also shown in Figure 4.8(b). These results once again indicate the inadequacy of row-only coding for the bursty traffic case.

In Figure 4.8, the results of event-driven simulations that last for 10^8 tagged cell transmissions are also shown. The perfect match between these results and those obtained by computations indicate that the analytical framework of Chapter 3 is quite accurate under arbitrary traffic conditions.

4.4 Discussion of Results

The results presented in this chapter, first of all, indicate that FEC can effectively recover lost cells over a wide load range for traffic streams of extremely diverse characteristics (see Figures 4.5 and 4.8). Even for very high loads, it is practically possible to achieve more than 3 orders of magnitude CLR reduction for both random and bursty traffic. However, this requires the cell loss process characteristics be taken into account, and the code type be chosen accordingly. Of the average rate and burstiness of cell losses, the latter is found to play a more important role in this regard. Obviously, the user, or even the network, does not have precise a priori information about the cell loss process burstiness. Yet, information about the traffic characteristics is usually available as used by the

connection admission control and traffic policing protocols. An appropriate code type can be determined based on this information and the current network status during the connection establishment phase. As our results indicate that the cell loss process burstiness for a particular traffic stream is influenced to a greater extent by the burstiness of that stream than by that of the interfering ones, the network status is of secondary concern here. Consequently, a code can usually be designed based merely on the burstiness and the delay and loss requirements of the traffic stream for which FEC is desired. Typically, row-only coding can be used if the traffic is random since then it becomes quite effective with appreciably small decoding latencies even under heavily bursty interfering traffic conditions. For bursty traffic however, it is not adequate unless the row size is too large. In this case, one would better use column-only coding as it effectively recovers burst losses with limited decoding complexities at the expense of acceptable decoding latency penalties.

Joint coding, on the other hand, combines the respective fast and burst loss recovery capabilities of the individual row and column coding techniques. It is possible to find practical joint codes that perform quite well for both random and bursty traffic. However, here it is crucial that the row code component is designed sufficiently powerful with large row size and small code rate so that a reasonable gain is achieved, and hence, the fast recovery advantage is taken during the first row decoding phase for random traffic. Such a row code component is also desirable as it will be effectively involved in optimal decoding for bursty traffic. Note that, since the constrained decoder ignores successful cells of unrecoverable rows, the number of cells that it wastes will increase with the row size. Therefore, the use of optimal decoder is preferable for joint codes with large row sizes.

Although using row- and column-only codes tailored specifically to the the traffic for which FEC is desired can be more advantageous, a joint code as described above provides a certain degree of insensitivity to the traffic characteristics. In other words, it relaxes the need for a priori traffic information. Column-only coding may be expected to have the same feature as it will be effective for random traffic as well as for bursty traffic. However, it will typically

incur unnecessarily large decoding latencies if it is tailored to bursty traffic and the traffic is actually random, or will be ineffective for bursty traffic otherwise. In fact, joint coding is useful especially in this regard, and is attractive for traffic streams of unpredictable characteristics. Furthermore, it has the potential to respond to time-varying traffic characteristics and requirements. A traffic stream may have random and bursty periods. Then, the row code component enables fast recovery during random periods whereas the optimal decoder acts effectively during bursty periods. Due to this feature, the use of joint coding is promising for a VP connection which aggregates several traffic streams of diverse characteristics.

Chapter 5

Performance of Hybrid Two-Level FEC and ARQ

This chapter focuses on the use of two-level FEC in conjunction with ARQ for a long-distance VC service which tolerate no loss at all. Assuming that there are multiple nodes on the VC, we carry out detailed simulations, and quantify the improvement in delay-throughput performance achieved by using FEC under relatively heavy and bursty traffic conditions.

In the AAL (ATM adaptation layer), the user data is segmented into variable-size packets at the source site [21]. These packets, known as PDUs (protocol data units), are further segmented into cells, and transmitted cell-by-cell to be reassembled at the destination site. A "corrupted" PDU with one or more missing cells is retransmitted by the source. In [1], a particular FEC technique has been considered to minimize PDU retransmissions. This technique involves transmission of M parity PDUs along with each group of N information-bearing PDUs. The parity PDUs are generated by using an optimal, maximum-distance separable erasure correcting code cell-wise over the information-bearing PDUs, which are padded sufficiently many empty cells to be all of the same size. Then, up to M corrupted PDUs out of $N + M$ can be reconstructed at the destination without requiring retransmissions. The performance of this technique for a multi-node VC connection was analyzed in [2] assuming transportation, switching, and

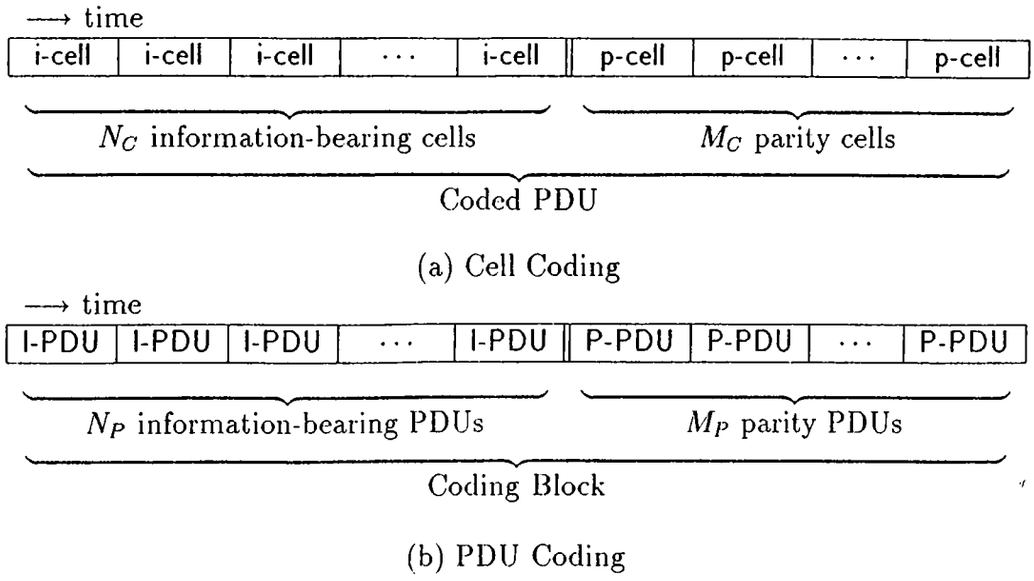

Figure 5.1: Two-level FEC over cells and PDUs: an alternative illustration of the coding matrix of Figure 2.1.

queueing of PDUs. Our simulations extend and complement this analysis mainly in two respects. First, we simulate the system on a discrete-event basis slot-by-slot, and realize transportation, switching, and queueing of cells. Second, we consider the use of two-level FEC. That is, we employ two codes simultaneously: one is over PDUs as just described, and the other is over cells within a PDU.

In the following, we first describe the hybrid two-level FEC and ARQ scheme in Section 5.1. Then, we introduce the simulation model in Section 5.2. The results obtained are presented and discussed in Section 5.3.

5.1 Hybrid Two-Level FEC and ARQ Scheme

In this study, we assume that PDUs are of fixed size, and use N_C to denote the number of cells in a PDU. Two-level FEC involves coding over cells within a PDU and coding over PDUs, which we refer to as *cell coding* and *PDU coding*, respectively. This is illustrated in Figure 5.1. We first append M_C parity cells to

each N_C-cell PDU. Then, we have a *coded* PDU of $N_C + M_C$ cells, and similarly append M_P parity PDUs to each group of N_P coded PDUs. We call the resulting block of $N_P + M_P$ PDUs a *coding block.* In fact, a coding block can be viewed as a matrix of cells where each PDU constitutes a row. Then, cell coding corresponds to "row coding," and PDU coding to "column coding," as illustrated in Figure 2.1. We consider the uses of optimal encoder and constrained decoder described in Section 2.1. Therefore, the destination can recover up to M_C lost cells per PDU. Similarly, referring to a corrupted PDU with more than M_C missing cells as being "lost," a coding block can be reconstructed completely provided that it has no more than M_P lost PDUs.

The ARQ protocol we consider involves selective retransmissions of lost PDUs based on timeouts or negative acknowledgment messages (NAKs) from the destination. The implementation details of this protocol, i.e., the rules followed by the source and the destination are summarized below.

The transmissions of new PDUs are governed by a window flow control mechanism with *permits* [3]. Initially, the source has a certain number of permits which are consumed one by one as PDUs are transmitted for the first time. Once the source runs out of permits, it cannot transmit new PDUs until some permits come back in the form of acknowledgment messages (ACKs) from the destination. Each transmitted PDU can be viewed as capturing a permit, and holding it until success. Therefore, PDU retransmissions do not consume permits.

Observe that the permit mechanism described above avoids an unacceptable increase in the number of PDUs in the network due to retransmissions. Such a PDU-level flow control may not be appropriate for ATM networks. Nevertheless, we use it to mimic the effects of window- or counter-based cell-level traffic policing mechanisms that are considered to prevent congestion in an open-loop fashion at the user-network interface [16].

We assume that the destination has *cell level memory.* That is, it stores successful cells although the PDUs that they belong to are lost. Obviously, this feature should be expected to have a strong impact on the performance of the hybrid scheme as the work left for successive retransmission cycles decreases from

one cycle to the next.

The source stores copies of the PDUs it has transmitted in a local PDU buffer until they are ACK'ed by the destination. There are two types of ACKs. The first one is a *PDU ACK* indicating the success of a particular PDU. The second one is a *block ACK*, and indicates that a coding block is completely reconstructed at the destination. The destination sends PDU ACKs immediately upon accumulation of N_C cells in the PDUs. A block ACK is sent similarly upon complete reception or recovery of N_P PDUs in the associated coding block. Observe that a block ACK is equivalent to a PDU ACK if there is no PDU coding, i.e., if $M_P = 0$.

Upon reception of a PDU ACK, the source drops the ACK'ed PDU from the local PDU buffer, and increments the permits by one. If a block ACK is received, all the relevant PDUs that have not been ACK'ed before are dropped, and the permits are incremented accordingly. For example, assuming that ACKs are never lost, a block ACK corresponds to $M_P + 1$ PDU ACKs, and the permits are incremented by that number. Also, if the source is busy with retransmission of a PDU, it breaks this retransmission upon receiving an ACK for that PDU itself or for the coding block that the PDU belongs to.

Similar to PDU ACKs, a NAK indicates the loss of a particular PDU. When there is no PDU coding, the destination sends NAKs immediately upon detecting more than M_C lost cells in the PDUs. In the PDU coded cases, NAKs are delayed until the destination realizes that the lost PDUs cannot be recovered by the PDU decoder.

Upon reception of a NAK, the source, if it is not busy with (re)transmission of another PDU, immediately starts retransmitting the NAK'ed PDU. Otherwise, the NAK'ed PDU is retransmitted after the source completes that (re)transmission. When there are no pending NAKs, it retransmits timed out PDUs if there are any. It ignores the NAKs of newly timed out PDUs. If there is not any timed out PDU, either, it can transmit a new PDU provided that it has at least one permit.

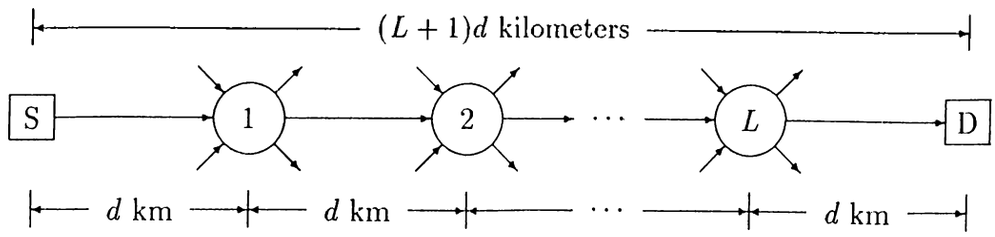

Figure 5.2: L-node VC model.

5.2 Simulation Model

We consider a long-distance VC connection as shown in Figure 5.2. There are L equally spaced nodes on the VC. The source and the destination are $(L+1)d$ km apart. The time is slotted, and all the events along the VC occur synchronously at the slot boundaries. In the following, we use d to denote the propagation delay on the links in terms of slots.

At each node, there is an $N \times N$ ATM switch capable of transporting all the simultaneous input cells to the requested output ports in zero time without internal blocking. We consider two output queueing techniques [9], [22]. In the first technique, there is a dedicated buffer of capacity B cells for each output port (*dedicated queueing*). In the second technique, all the cells share a common buffer pool of capacity NB cells regardless of their output requests (*shared queueing*). Dedicated queueing avoids interaction of traffic streams destined for different output ports, and yields smaller average queueing delays as compared to shared queueing. However, the latter technique makes more efficient use of the total storage capacity, and hence, provides savings in cell losses when the traffic is bursty. The primary disadvantage of shared queueing is that congestion at one output port degrades the quality of service for traffic streams destined for other output ports.

In the simulations, we focus on the forward traffic flowing from the source to the destination, and perform FEC on this *tagged* traffic. At the nodes, the tagged cells interfere with the *untagged* cells belonging to other VCs. We assume that

N_C-cell tagged PDUs arrive at the source continuously according to a Poisson process with rate p/N_C, where $p \in (0,1)$ is the normalized tagged load. The cells of a tagged PDU, coded or not, are transmitted consecutively in successive slots. The untagged PDUs are generated at the $N-1$ untagged input ports of L nodes according to independent Poisson processes all with the same rate p/N_C, and are transmitted similarly. An untagged PDU joins the tagged VC with probability $1/N$. If it joins, its cells depart from the tagged VC at the downstream nodes independently with probability $(N-1)/N$. Note that the aggregate load increases in the downstream direction. Ignoring losses, the normalized aggregate load at the tagged output port of node l, $1 \leq l \leq L$, can be shown to be $p(2-1/N^l)$.

Observe that the worst case end-to-end delay from the source to the destination is $(L+1)d + LB$ and $(L+1)d + LNB$ slots for the cases of using dedicated and shared buffers at the nodes, respectively. In real life, ACKs and NAKs flow from the destination back to the source through the network, possibly on the same path as the forward traffic follows. Therefore, they are subject to both losses and random queueing delays. For the sake of programming simplicity, we assume that there are dedicated, lossless, constant-delay ACK/NAK channels. To mimic the increase in queueing delays that ACKs and NAKs would normally encounter with increasing network load, we take the delays of these channels as $(L+1)d+LBp$ and $(L+1)d+LNBp$ slots for the dedicated and shared queueing cases, respectively.

Upon (re)transmission of the last cell of a PDU, the source sets a counter for that PDU which is decremented by one in each slot until a relevant ACK or NAK is received. If the counter expires before the PDU is ACK'ed or NAK'ed, a timeout occurs, and the PDU is retransmitted. The initial value of the counter (the timeout period) is set as $2(L+1)d+LB(1+p)$ and $2(L+1)d+LNB(1+p)$ slots for the dedicated and shared queueing cases, respectively. These initial values correspond to the maximum time that the source has to wait for a PDU ACK. Since ACKs are sent by the destination without any delay, unnecessary retransmission of a successful PDU upon a timeout is not possible. However, in the PDU coded cases, the NAKs of lost PDUs are delayed with the hope

that they can be recovered by the PDU decoder. This may cause unnecessary PDU retransmissions. But, in the case that the PDU decoder cannot recover those lost PDUs, the timeout mechanism is useful in that it compensates for the unnecessary NAK delays.

Finally, the initial number of permits at the source is set to the expected number of PDUs that the source can transmit in the maximum round-trip propagation and queueing time, i.e., the above timeout periods for the two output queueing techniques. Let T denote this time. Then, $T/(N_C + M_C)$ PDUs fit in time T, and this number multiplied by the normalized effective tagged load, $p(1 + M_C/N_C)(1 + M_P/N_P)$, gives the initial number of permits. After a little manipulation, it can be shown that the source initially has $\lambda T(1 + M_P/N_P)$ permits[1], where λ is the tagged PDU arrival rate p/N_C.

5.3 Results and Discussion

In the simulation study, we took the number of nodes on the VC as $L = 4$, the number of input/output ports at the nodes as $N = 8$, and the end-to-end propagation delay as 10240 slots[2]. Then, fixing the parameters N_C and N_P respectively as 16 and 256, we first performed various simulations to optimize M_C and M_P separately for both output queueing techniques with the buffer capacities of $B = 16$, 64, and 256 cells per output port. In these optimizations, taking the throughput limitation due to coding overhead into account, we measured the *average PDU delay,* i.e., the average time in slots that a tagged PDU spends in the network, as a function of $p < [(1 + M_C/N_C)(1 + M_P/N_P)]^{-1}$. When there is PDU coding, the averages were computed over information-bearing PDUs so as to make meaningful comparisons with the no coding case.

While optimizing M_C, we kept $M_P = 0$, and tried $M_C = 1$, 2, 3, 4, 6, 8, and 12. The results of these simulations indicated uniformly increasing cell coding

[1] Here, $\lambda T(1 + M_P/N_P)$ is assumed to be rounded to the nearest integer.

[2] Observe that if the data transmission rate is 155.52 Mbps, and the propagation speed is 2×10^8 m/s, this delay corresponds to a total link length of about 5600 km, or 3500 miles, which is the length of a coast-to-coast U.S. or transcontinental Europe connection.

gain with increasing M_C especially for high loads. Due to the severe throughput limitation associated with large M_C, $M_C = 4$ appeared as a reasonable choice for both queueing techniques. In the optimization of M_P, we similarly kept $M_C = 0$, and tried $M_P = 2$, 4, 6, 8, 10, 12, 14, 16, 24, and 32. This time, we did not observe a uniform increase in the PDU coding gain with increasing M_P, and over the entire load range, we found $M_P = 4$ and $M_P = 16$ as the optimum choices for the dedicated and shared queueing cases, respectively. In the following, we discuss the performances of the cell-only, PDU-only, and joint cell-and-PDU codes with $N_C = 16$, $N_P = 256$, and the parameters M_C and M_P chosen as above.

5.3.1 Results for the Dedicated Queueing Case

The results obtained for the case of dedicated output buffers are shown in Figure 5.3. Observe that, for $B = 16$, cell-only coding outperforms PDU-only coding over the entire load range. In fact, PDU coding, alone or together with cell coding, is quite unsatisfactory in this case. Due to the bursty traffic characteristics, i.e., due to the arrival and transmission of cells in batches, the buffers of such small capacity are almost always full even for small p, and cells are lost randomly with high probabilities. Consequently, a coding block is very likely to have several corrupted PDUs. Although those corrupted PDUs themselves may suffer from several cell losses, successful cells accumulate at the destination due to the cell level memory. Therefore, the cell decoder can recover lost PDUs by requiring fewer number of retransmissions as compared to the no coding case.

There are two major reasons behind the poor performance of PDU coding for $B = 16$. First, because cells are lost randomly with high probabilities, a coding block is very likely to have several lost PDUs, as stated above. Second, the latency associated with PDU decoding is large, and the permit mechanism has an adverse effect in this respect. Recall that each PDU holds a permit until success. The more the number of times a PDU is to be retransmitted, the longer it will take for the permits to be released by the destination. Therefore, in heavily congested conditions, the source is expected to be almost always in lack

of permits. This may expand the coding block transmission in time, and even result in quite unpredictable and undesired transmission scenarios. An extreme example is that the PDUs in the tail of a coding block, including the parity PDUs, may be still waiting for their first transmissions while those in the head are circulating in the network as if there is no coding. Obviously, PDU coding becomes totally useless, and even the cell level memory does not help in such a case.

As the buffer capacities increase to $B = 64$ and 256, PDU coding starts to provide gain. In fact, for moderate p, PDU-only coding performs better than cell-only coding. For such buffer capacities and loads, cells are lost in rare bursts, and hence, PDU coding exhibits gain. As the normalized effective load approaches unity, congestion builds up as in the case of $B = 16$, and PDU coding becomes useless similarly. On the other hand, joint coding outperforms cell-only coding or PDU-only coding for almost all p, except for a small degradation around $p = 0.45$ when $B = 256$, which is due to the individual performance degradation in cell coding.

5.3.2 Results for the Shared Queueing Case

Figure 5.4 shows the results obtained for the shared output queueing case. Comparing these results with those of Figure 5.3, we observe better delay-throughput performance for low loads. This is because the traffic is bursty for low loads, and hence, shared queueing saves several cell losses. In particular, when $B = 16$, the gain of shared queueing is quite significant, and the critical load after which retransmissions begin is shifted to about $p = 0.35$ from very small values. In fact, for small to moderate p, shared queueing with $B = 16$ achieves the performance of dedicated queueing with $B = 64$. This sharing gain diminishes with increasing B.

Contrary to this observation for low loads, dedicated queueing outperforms shared queueing when the load is high. This is mainly because traffic destined for a certain output port dominates in the common buffer pool to the detriment of

the tagged traffic (the primary disadvantage of shared queueing). Also, cells suffer from higher average queueing delays in the shared queueing case as compared to the case of dedicated queueing with the same B.

The important difference between the results for the two queueing techniques with regard to coding performance is observed in the case of PDU coding when $B = 16$. Recall that, for dedicated queueing with $B = 16$, cells are lost randomly with high probabilities even for small p, and hence, PDU coding provides no significant gain. Shared queueing with the same B, however, saves a significant fraction of cell losses. This results in bursty cell loss process characteristics, and in turn, makes PDU coding effective. For $B = 64$ and 256, the coding performance exhibits similar trade-offs for both queueing techniques.

5.3.3 Concluding Remarks

The results presented in this chapter indicate that FEC can provide significant reductions in the average PDU delay when used together with ARQ for a long-distance, loss-sensitive VC service. In accordance with the results of Chapter 4, Figures 5.3 and 5.4 indicate that two-level coding combines, or adds, the gains of individual cell and PDU coding techniques, which act independently, exhibiting gain and loss by means of different mechanisms. Cell coding is effective when cells are lost randomly. With the help of the cell level memory at the destination, it is still quite effective even when the cell loss probability is very high. For moderate loads where we have burst losses, PDU coding performs better than cell coding as expected. Two-level coding exhibits the advantages and the disadvantages of the two coding techniques, and provides a net gain in most cases. In particular, it can reduce the average PDU delay approximately to the extent of a half or more over the entire load range.

In the simulations, we consider the use of constrained decoder, which limits the effectiveness of PDU coding. In the light of the results of Chapter 4, we expect that much better results can be achieved by using the optimal decoder described in Section 2.1.

Finally, there can be alternative hybrid two-level FEC and ARQ schemes. Recall that the AAL-PDUs are actually of variable size. In the case that the PDU size variance is high, coding over cells within a PDU and coding over PDUs both may require too many empty cells be padded, and hence, reduce the coding efficiency significantly. Therefore, it is better to confine the two-level code to blocks of fixed number of cells regardless of whether these blocks cross over the AAL-PDU boundaries or not. Then, one can still consider retransmissions of the AAL-PDUs, or take the two-level coding block as the basic unit of retransmission. The latter technique seems to be more advantageous as compared to the former. First, it is easier to deal with retransmission of fixed-size data units. Second, in the case of a large AAL-PDU that consists of several coding blocks, the former technique may be quite inefficient because a failure in complete recovery of only one of those coding blocks results in retransmission of the entire AAL-PDU.

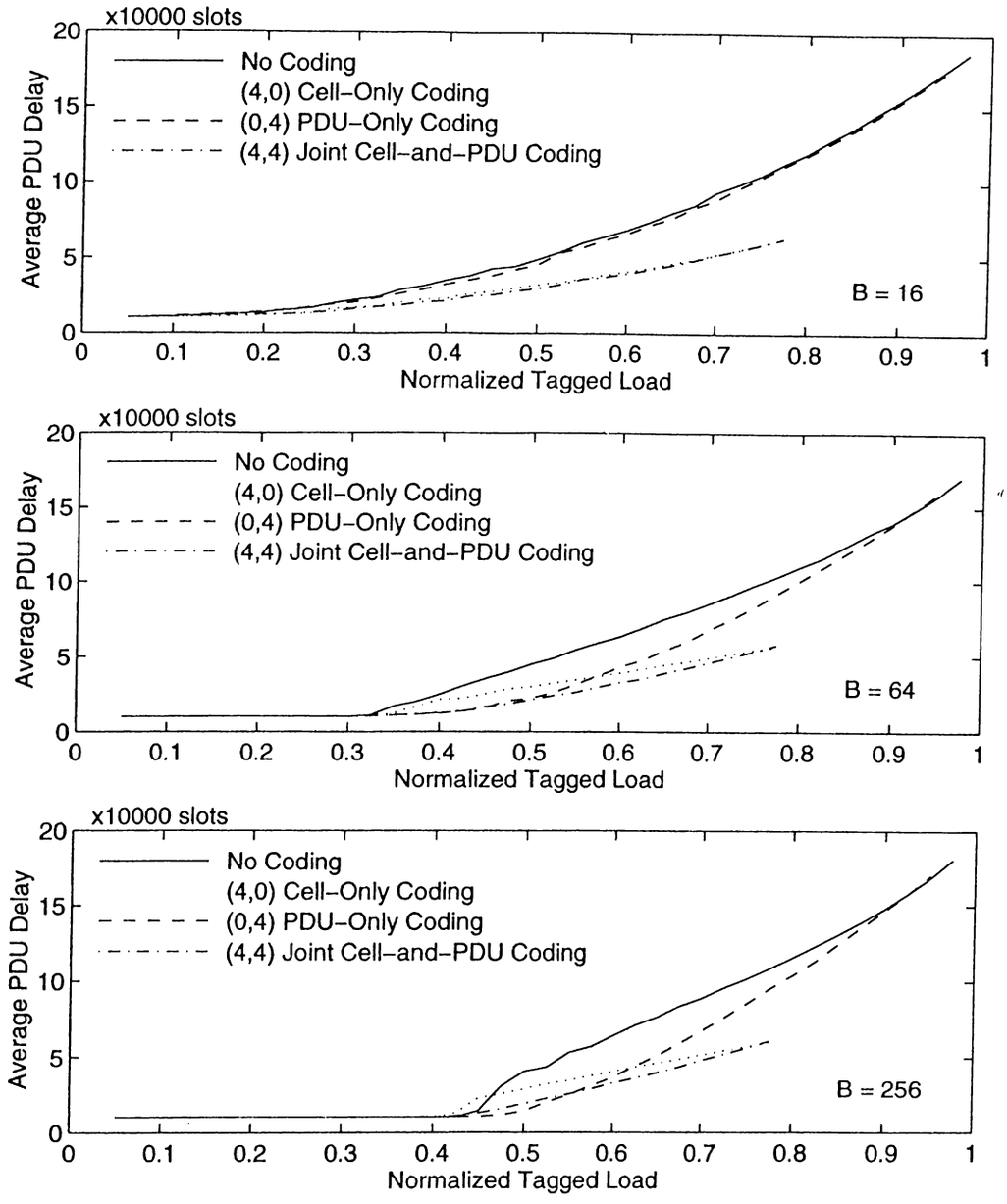

Figure 5.3: Performance of the hybrid scheme for the dedicated output queueing case. The numbers of information-bearing cells in a PDU, N_C, and information-bearing PDUs in a coding block, N_P, are fixed as 16 and 256, respectively. The numbers of parity cells in a PDU, M_C, and parity PDUs in a coding block, M_P, are indicated in the format (M_C, M_P). Averages are computed over 512 coding blocks.

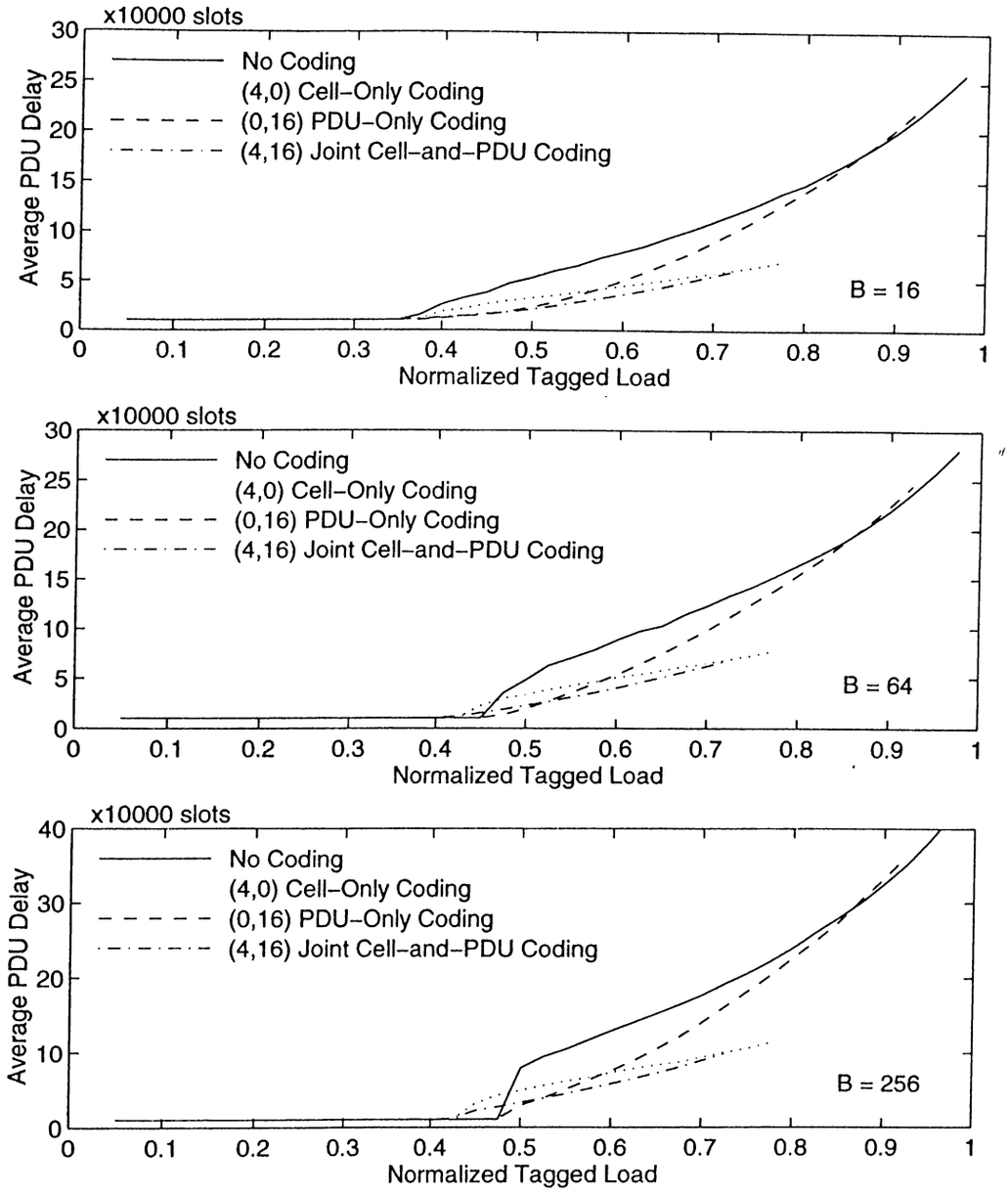

Figure 5.4: Performance of the hybrid scheme for the shared output queueing case. The numbers of information-bearing cells in a PDU, N_C, and information-bearing PDUs in a coding block, N_P, are fixed as 16 and 256, respectively. The numbers of parity cells in a PDU, M_C, and parity PDUs in a coding block, M_P, are indicated in the format (M_C, M_P). Averages are computed over 512 coding blocks.

Chapter 6

Summary and Conclusions

In this thesis, we considered the use of a two-level forward error correction (FEC) scheme for lost cell recovery in wide-area ATM networks. The scheme exploits the individual simple block coding and code interleaving techniques simultaneously, and is based on transmitting separate parity cells along with the information-bearing ones. The parity cells are generated by using optimal, maximum-distance separable erasure correcting codes so that as many losses in a coded block as there are parity cells can be recovered.

We discussed the two-level encoding and decoding operations in detail. In particular, we proposed alternative decoding techniques, and briefly addressed the practical problems associated with their implementation. Two-level FEC can be performed for both virtual channel (VC) and virtual path (VP) connections in an ATM network. We also discussed and proposed methods to deal with the practical issues of erasure channel realization for the cases of using the scheme for these two types of connections. Apart from these practical issues, we put the major emphasis on the performance of the two-level FEC scheme, and showed that it is useful.

We carried out an extensive performance study to compare the two-level coding performance with the individual simple and interleaved block coding performances. This study falls into two parts. In the first part, we focused on the case of using FEC for high-speed, delay-sensitive services such as real-time video,

interactive computing, and distributed processing. In the second part, motivated by services with stringent loss requirements such as file transfer, we considered the use of a hybrid cell loss recovery scheme which employs FEC in conjunction with ARQ.

The performance study in the first part was based on a discrete-time analytical framework. Modeling a single-node VC as an output-buffered ATM multiplexer, we first developed an exact cell loss model which captures the bursty nature of cell losses precisely under arbitrary traffic conditions. Based on this model, we analyzed the performance of two-level coding with a rather constrained decoder that cannot take full advantage of code interleaving. Then, we developed a novel cell loss model which has much less complexity as compared to the exact one, and is still quite accurate. This model was used to analyze the two-level coding performance with an optimal decoder that performs all possible cell loss recoveries in a two-level coding block.

In all these analyses, we particularly investigated the effect of traffic characteristics on the coding performance. The results obtained, first of all, indicate that FEC can be very effective in recovering lost cells over a wide range of network load under diverse traffic conditions. However, this requires the use of an appropriate coding technique chosen according to the traffic characteristics. The traffic stream for which FEC is performed is found to have a more important role in this regard as compared to the interfering traffic. Typically, simple block coding can be quite effective with appreciably small decoding latencies for a random traffic stream. For a bursty traffic stream, however, the cell loss process is more bursty, and it becomes inadequate unless the block length is very large, or equivalently, the decoding complexity is very high. In this case, interleaved block coding is preferable. On the other hand, it is possible to find two-level codes which combine the fast and burst loss recovery capabilities of the individual coding techniques. Here, it is crucial that the simple block code component is sufficiently powerful, and the decoder is the above optimal decoder or a simpler but near-optimal one. Two-level coding then relaxes the need for a priori knowledge of the traffic stream. It allows fast loss recoveries if the traffic is random, and burst loss

recoveries if the traffic is bursty. Therefore, it is attractive especially for traffic streams of unpredictable or time-varying characteristics. As VP traffic exhibits such characteristics, the use of two-level FEC is promising for a VP connection.

The performance of the hybrid FEC and ARQ scheme was studied in detail via event-driven simulations. In this study, we focused on a long-distance, four-node VC connection, and quantified the reduction in the average end-to-end PDU delay achieved by using FEC, where a PDU is a group of fixed number of cells taken as the basic unit of retransmissions. The two-level FEC scheme we considered in this context involves coding over cells within a PDU (simple block coding) and coding over PDUs (interleaved block coding). The unrecoverable PDUs with too many lost cells were assumed to be retransmitted selectively based on timeouts or negative acknowledgment messages from the destination.

The simulation results obtained for the hybrid scheme are in accordance with the results of the analyses. That is, coding over cells within PDUs is effective for the case of random cell losses, and coding over PDUs is effective for the case of burst cell losses. Two-level coding, on the other hand, is again found to combine the advantages of the individual coding techniques.

Bibliography

[1] E. Ayanoğlu, R.D. Gitlin, P.K. Johri, and W.S. Lai, "Protocols for error/loss recovery in broadband ISDN," *Proc. 7-th Int'l. Teletraffic Congr. Sem.,* Morristown, NJ, October 1990.

[2] E. Ayanoğlu, R.D. Gitlin, and N.C. Oğuz, "Performance improvement in broadband networks using forward error correction for lost packet recovery," *Journal of High Speed Networks,* vol. 2, no. 3, pp. 287-303, 1993.

[3] D. Bertsekas and R. Gallager, *Data Networks.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1987.

[4] V.K. Bhargava, D. Haccoun, R. Matyas, and P.P. Nuspl, *Digital Communications by Satellite: Modulation, Multiple Access and Coding.* John Wiley&Sons, Inc., New York 1981.

[5] E. W. Biersack, "A simulation study of forward error correction in ATM networks," *Computer Communications Review,* Vol. 2, pp. 36-47, January 1992.

[6] E.W. Biersack, "Performance evaluation of forward error correction in ATM networks," *Proc. ACM SIGCOMM'92,* pp. 248-257, Baltimore, MD, August 1992.

[7] R.E. Blahut, *Theory and Practice of Error Control Coding.* Addison-Wesley Publishing Company, Inc., Massachusetts 1983.

[8] I. Cidon, A. Khamisy, and M. Sidi, "Analysis of packet loss processes in high-speed networks," *IEEE Trans. Inform. Theory,* vol. 39, no. 1, pp. 98-108, January 1993.

[9] M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE J. Select. Areas in Commun.,* vol. 6, no. 9, pp. 1587-1597, December 1988.

[10] ITU-T Recommendation I. 311: *B-ISDN General Network Aspects.* June 1991.

[11] T. Kitami and I. Tokizawa, "Cell loss compensation schemes employing error correction coding for asynchronous broadband ISDN," *Proc. INFOCOM'90,* pp. 116-123, San Francisco, CA, June 1990.

[12] L. Kleinrock, *Queueing Systems Volume I: Theory.* John Wiley & Sons, Inc., New York 1975.

[13] N.F. Maxemchuk, "Dispersity routing," *Proc. ICC'75,* pp. 41.10-41.13, San Francisco, CA, June 1975.

[14] A.J. McAuley, "Reliable broadband communications using a burst erasure correcting code," *Proc. ACM SIGCOMM'90,* pp. 287-306, Philadelphia, PA, September 1990.

[15] H. Ohta and T. Kitami, "A cell loss recovery method using FEC in ATM networks," *IEEE JSAC,* vol. 9, no. 9, pp. 1471-1483, December 1991.

[16] M. de Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN.* Ellis Horwood Limited, London 1991.

[17] S.M. Ross, *Introduction to Probability Models.* Academic Press Inc., Florida 1985.

[18] N. Shacham, "Packet recovery and error correction in high-speed wide-area networks," *Proc. MILCOM'89,* pp. 29.5.1-29.5.7, May 1989.

[19] N. Shacham and P. McKenney, "Packet recovery in high-speed networks using coding and buffer management," *Proc. INFOCOM'90,* pp. 124-131, San Francisco, CA, June 1990.

[20] W. Stallings, *Data and Computer Communications.* MacMillian Publishing Company, Inc., New York 1991.

[21] W. Stallings, *ISDN and Broadband ISDN.* MacMillian Publishing Company, Inc., New York 1992.

[22] F. A. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks," *Proceedings of the IEEE,* vol. 78, no. 1, pp. 133-167, January 1990.

[23] L. Zhang and K.W. Sarkies, "Modeling of a virtual path and its applications for forward error recovery coding schemes in ATM networks," *Proc. SICON'91,* pp. 259-264, Singapore, September 1991.

[24] L. Zhang, "Statistics of cell loss and its application for forward error recovery in ATM networks," *Proc. ICC'92,* pp. 325.3.1-325.3.5, Chicago, IL, June 1992.

Vita

Nihat Cem Oğuz was born in Söke, Aydın, Turkey, in 1965. He received his B. Sc. degree from the Middle East Technical University, Ankara, Turkey, and M. Sc. degree from the Bilkent University, Ankara, Turkey, in 1987 and 1990, respectively, both in electrical and electronics engineering. His current research interests include performance evaluation of high-speed packet-switched communication networks, cell loss process characterization, and forward error correction for lost cell recovery in ATM networks.