INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa • University of Ottawa

Dynamic Call Admission Control under Partial Separation in ATM Networks

by

Hui Liu, M. Eng.

China Academy of Railway Science

A thesis submitted to
the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of
the requirements for the degree of
M.Sc., Systems Science

Faculty of Administration
for Systems Science Program

University of Ottawa
Ottawa, Ontario
August, 2001

The undersigned recommend to the Faculty of Graduate and Postdoctoral Studies acceptance of the thesis

Dynamic Call Admission Control under Partial Separation in ATM Networks

Submitted by

Hui Liu

China Academy of Railway Science

in partial fulfillment of
the requirements for the degree of
M. Sc., Systems Science

Thesis Supervisor

Chairman
Faculty of Administration

University of Ottawa
August 2001

Abstract

A dynamic call admission control scheme is proposed for ATM networks to support multiple Quality of Service (QoS) requirements. The scheme is based on the notion of partial separation. It divides the overall traffic flows into classes, within which each traffic source has similar QoS requirements, but may have different traffic characteristics, and the bandwidth of a link is dynamically allocated among the classes.

A dynamic bandwidth allocation strategy is introduced. It achieves the goal of bandwidth allocation by optimizing a cost function defined as an estimate of the overall cell loss rate that would be generated by the total offered load. It is dynamic in the sense that the bandwidth allocation is adjusted each time when a connection requires for an acceptance or a leave.

While optimizing the overall cell loss rate across classes, cell level constraint is satisfied for each class by deploying the cell loss rate estimate algorithm, proposed by H. Esaki in Connection Admission Control in ATM Networks. A simulation program is developed to validate the accuracy of the algorithm. Extensive numerical and simulation results are presented.

Numerical search method is deployed to locate the optimal bandwidth assignment set. Numerical results are presented to evaluate the efficiency and adequacy of the CAC scheme.

Acknowledgements

I would like to express my appreciation to my thesis supervisor Dr. Tet Yeap for his support, encouragement and guidance during the course of this work. I would also like to thank Professor Thizy and graduate staff of Faculty of Administration for their support.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Introduction

Computer and communication applications have led to the emerging of a variety of services, such as the World Wide Web, video conferencing, video on demand, and HDTV, which all require tremendous amounts of network resources for high quality transmissions. The needs for the integration of services and the promise of the technologies to support these services have resulted in interest in research, development and standardization of broadband integrated networks. One of the most prominent enabling technologies for the deployment of broadband ISDN is the asynchronous transfer mode (ATM) networks architecture.

ATM is the first scheme to provide a unified interface, which can be used by a variety of services with drastically different QoS requirements. It is based on packet switching in the sense that all traffic is transported through small, constant-sized cells. This technology allows flexibility in the choice of connection bit rates and enables the statistical multiplexing of variable bit rate traffic streams.

The exploitation of the ATM technology has its price: a rich set of rather sophisticated traffic and congestion control is required to protect the network and the user in order to achieve network performance objectives and optimize the usage of network resources. Congestion control schemes can be classified into preventive control and reactive control. In preventive congestion control, schemes can be set up at a switch to

prevent the occurrence of congestion; in active congestion control, a switch relies on feedback information controlling the level of congestion.

Call admission control (CAC) is a preventive control mechanism. CAC deals with the question of whether or not a switch can accept a new connection. When a new connection request is received at the network, a call admission procedure is executed to decide whether to accept or reject the call. A call is accepted if the network has enough resources to provide the QoS requirements of the connection request without affecting the QoS provided to connections already established in the network. Any technique designed in response to this question should function in real time, and try to maximize the utilization of network resources.

1.2 Motivations

Nevertheless, it must be ensured in the design of a CAC scheme that the QoS requirements of the existing connections do not degrade upon acceptance of the new call request, and meanwhile maximize the network resource utilization. Those come to two considerations, one is the way to satisfy the QoS requirements of a connection, the other is the way that the bandwidth is being allocated. For the past several years various call admission control and bandwidth allocation schemes have been proposed. However, some underlying constraints and conservatism exist in those schemes.

Before we proceed, let's take a step back and look at some basic concepts related to call admission control in ATM networks.

When a data source wants to transmit information, it requests a virtual channel (VC) to be established between the transmitter and receiver. Once a VC is established,

the data source generates a stream of ATM cells, each cell consisting of 53 bytes. A typical cell stream generated by an established VC consists of silent periods, during which no cells are generated, and active periods, during which cells are generated at the peak rate. An ATM multiplexer is a buffer and a high-speed link; the buffer receives the cells generated by established connections and transmits these cells, one after another, onto the high-speed link. Examples of services include voice, low-and high-quality fax, videoconference, video on demand, file transfer, image retrieval, and LAN interconnects.

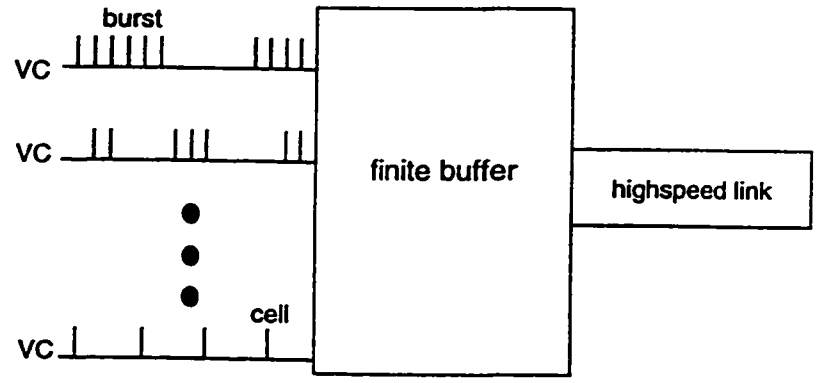

Figure 1. 1 an ATM multiplexer implementing complete sharing

During periods when the aggregate cell arrival rate exceeds the link capacity, the multiplexer can significantly delay or lose cells. A VC's allowable cell delay and loss are specified by its quality of service (QoS) requirements. To guarantee that all established VCs meet their QoS requirements, the multiplexer may have to deny certain VC establishment requests using a call admission control scheme.

Some existing admission schemes are based on the estimate of overall QoS using the queuing model illustrated in Figure 1.1. The effect of accepting a new connection on the QoS of existing connections is calculated. The new connection is accepted only if the

QoS of the existing connections is not degraded below the acceptable value and the QoS is guaranteed for the new connection as well. It guarantees an average QoS requirements over all connections at an ATM multiplexer, and therefore statistical multiplexing of connections across services rarely gives significant gains in performance when services have greatly different QoS requirements or greatly different statistical natures. Indeed, if we statistically multiplex services with widely different QoS requirements, then an overall QoS must realize the most stringent requirement; thus some services enjoy an overly generous QoS, which leads to inefficient use of transmission capacity.

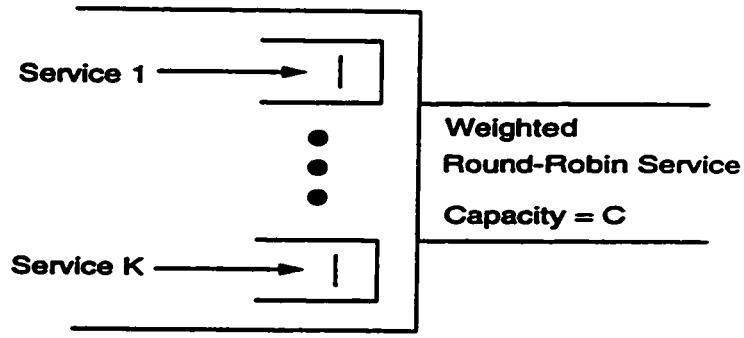

Figure 1. 2 an ATM multiplexer implementing service separation

Some admission schemes based on service separation allow for a degree of statistical multiplexing yet ensures satisfaction of the QoS requirements. They statistically multiplex VCs of the same service, but do not statistically multiplex across services. To explain basic notion behind service separation, assuming that the multiplexer's buffer is partitioned into K mini-buffers of equal capacity, and that an arriving cell from a service-k VC is placed in the kth mini-buffer. See Figure 1.2. The

link serves the mini-buffer in a weighted round-robin fashion. If a cell from service-*k* VC arrives to find the *k*th mini-buffer is full, it is lost.

Service separation restricts the multiplexing within the same service. It may not sensibly degrade the performance when services have largely different QoS requirements and/or statistical natures, but it does not take full advantage of statistical multiplexing.

Table 1. 1 Cell loss rate objectives for various B-ISDN applications

Application	Cell Loss Rate
Telephony	1×10^{-3}
Hi-fi sound	1×10^{-5}
Videophone (CBR)	1×10^{-8}
Videophone (VBR)	1×10^{-8}
Videoconference (VBR)	4×10^{-9}
TV Distribution	1×10^{-10}
MPEG1	1×10^{-8}
MPEG2	2×10^{-9}
Data transmission	1×10^{-6}
Distributed computing	1×10^{-6}

For some service profiles, it may be advantageous to statistically multiplex across some of the different services. For example, there may be pairs of services, which have similar, but not identical, cell generation characteristics and QoS requirements. Table 1.1

shows the QoS requirements with respect to cell loss rate for various B-ISDN applications [1]. It is obvious that some cell loss probabilities are close, some differ by more than an order of magnitude. Organizing similar services into groups is the basic concepts behind partial separation. It can be said that partial separation is a certain degree of service separation.

The other aspect needs to be considered is bandwidth allocation. At the heart of any CAC algorithm is the estimation of bandwidth required by a set of sources so as to respect each individual source's QOS requirements. Estimating the value of the statistical bandwidth for an incoming call request must address the following issues, first, the QoS requirements of the new connection must be guaranteed; second, the QoS requirements provided to preexisting connections must not be affected when they are multiplexed with the new connections.

Every statistical bandwidth allocation procedure uses either a static or dynamic allocation strategy. In static allocation, a reference model is used to determine the bandwidth allocation, and this strategy does not enable the system to adapt to changing situations. Dynamic allocation was proposed to overcome the deficiencies of static allocation. In dynamic allocation, actual traffic conditions are monitored, and bandwidth is reallocated based on the changing situations.

The tradeoff in a bandwidth allocation algorithm is between its simplicity and accuracy. A simple mechanism for bandwidth allocation may be used in real-time call admission control to determine whether the available bandwidth can accommodate a new connection, while an accurate formula usually results in higher bandwidth utilization.

Except by peak bandwidth assignment to each connection, there is no immediately clear way to identify the amount of bandwidth required by the connection since all the information is segmented into fixed-sized cells and multiplexing statistically. This has given rise to many investigations on bandwidth allocation and related CAC. Some determine the required bandwidth based on the estimated cell loss [2], some allocate bandwidth by optimizing a particular reward function either statically or dynamically [3].

1.3 Objectives

In this thesis, we propose a call admission control scheme that implements partial separation. This approach alleviates the problem with complete sharing and service separation by classifying the traffic into classes based on the QoS requirements with respect to the cell loss probability. Each source within a class has similar cell loss probability. The link capacity of an outgoing link is dynamically allocated among the classes by optimizing a cost function. It is dynamic in the following sense:

- It reallocates the bandwidth of each class when a new connection requests acceptance. The reallocation is done in such a way that the QoS of existing connections does not fall below the acceptable value on account of the new connection.
- Also, when a connection leaves the multiplexer, it re-distributes the bandwidth among the classes so as to improve the QoS given to these connections.

We use numerical search technique to obtain the optimal bandwidth allocation set while optimizing the cost function. The method is implemented in this thesis. Numerical results are presented and comparison is performed with the CAC method proposed in [2].

We also thoroughly evaluate the cell loss rate estimate algorithm in [2]. It is used in this thesis as an approach of estimating the cell loss rate for each class in order to guarantee the QoS requirements for each class. Numerical and simulated results are presented to evaluate its sensitivity to various traffic characteristics.

1.4 Thesis Organizations

The thesis is organized as follows.

In Chapter 2 we survey some related work on bandwidth allocation and CAC methods reported in the literature, and particularly identify some underlying constraints of existing schemes. We also explore some cell loss rate estimate approaches, and present the cell loss rate estimate algorithm in [2] in details.

In Chapter 3, we motivate and present a new CAC scheme which implements partial separation and dynamically allocates the total bandwidth among traffic classes. It also presents the implementation of a numerical search method to obtain optimal bandwidth allocation set.

In Chapter 4 we discuss the performance of the new CAC method compared with the existing method [2]. Also, we conduct extensive numerical and simulation experiment to validate the algorithm that estimates the cell loss rate for each class.

Conclusion and future works are presented in Chapter 5.

Chapter 2 Background

2.1 Interested Works on Call Admission Control

Various approaches have been proposed to solve the problems related to bandwidth allocation and call admission control in the literature. Several CAC schemes are based on the estimate of QoS requirements such as the cell loss probability. Some CACs are formulated as an optimization problem where a particular reward function is optimized. Among those proposed schemes, [1], [2], [3], and [4] have been of particular interest and investigated during the course of the work. In what follows, we will briefly review the underlying principle of these schemes.

The equivalent bandwidth [1] of a source is one of the most popular notions in call admission. It is defined, as the minimum bandwidth needed to carry the traffic generated by that source in isolation without violating the QoS requirements. The appeal of CAC schemes based on equivalent bandwidth concepts lies in their inherent simplicity when determining whether a given set of traffic sources can be accommodated without violating their QoS requirements reduces to comparing the sum of the equivalent bandwidth of individual sources to the link capacity.

Each source is assumed to be an IFP (Interrupted Fluid Process). Let R be its peak rate, r the fraction of time the source is active, and b the mean duration of the active period. Let K be the buffer capacity. Based on the asymptotic behavior of the tail of the queue length distribution, the following equation gives the equivalent bandwidth for a single source:

$$c = \frac{a - K + \sqrt{(a-K)^2 + 4Kar}}{2a} R \tag{2-1}$$

Where,

$$a = \ln(1/\varepsilon) b(1-r)R$$

In this framework, the total bandwidth of N multiplexed connections is equal to the sum of the equivalent bandwidths of individual connection c_i; that is:

$$C = \sum_{i=1}^{N} c_i \tag{2-2}$$

However, C significantly overestimates the required bandwidth for the aggregate traffic, since the interaction between individual connections is not taken into consideration. To capture the effect of statistical multiplexing, the Gaussian approximation is used together with the equivalent bandwidths, then C is,

$$C = \min\left\{ m + \alpha'\sigma, \sum_{i=1}^{N} c_i \right\} \tag{2-3}$$

Where, m is the mean bit rate of the aggregate traffic,

σ is the standard variance of the bit rate of the aggregate traffic.

Despite its simplicity, the equivalent bandwidth principle is highly conservative when the buffer size is small or moderate. The cause of this conservatism are that it is derived under the asymptotic regime where the product of buffer size and cell loss probability tend to zero, and it uses buffer overflow probability as the QoS requirements, which is normally larger than the corresponding cell loss probability. Some studies have been carried out in the literature to overcome the shortcomings of equivalent bandwidth method.

Several call admission control schemes have been proposed which are based on an upper bound for the cell loss probability. H. Esaki [2] proposed an upper bound based on the average number of cells transferred from multiplexed connections during some fixed interval. At the ATM multiplexer, all connections are divided into low and high groups based on their cell transmission speeds. 1/100 is defined as the threshold value whether a connection is a low speed connection or a high speed connection. A connection is a low speed connection when the ratio of its cell transmission speed to cell transmission speed of the multiplexed line is smaller than the threshold. On the contrary, a connection is a high speed connection when the ratio of its cell transmission speed to cell transmission speed of the multiplexed line is higher than the threshold. The time interval T_I was taken to be equal to the inverse of $V/100$, where V is the cell transmission speed of the multiplexed line. By assuming a low speed connection transferring cells at the speed of the threshold value, and a high speed connection transferring cells continuously at the speed of its peak cell transmission speed during T_I, an upper bound on the cell loss probability can be derived. A new connection is only admitted if the resulting cell loss probability is less than the required cell loss probability on account of the new connection.

This scheme is suitable for real-time operation even in large diversity of connection types because the amount of calculation for call admission control is reduced remarkably compared to conventional algorithms [2]. However, like most algorithms, the estimated cell loss rate is an average over all connections. Hence, the network has to be engineered for a QoS requirement that may be overly stringent for a large fraction of the

traffic in the case of services with largely different QoS; this may degrade network resource utilization.

Rather than having all connections to completely share the bandwidth of an outgoing link, some approaches impose a certain structuring on the allocation of the resources, where typically services are grouped into classes. Traffic within a class is homogeneous in terms of performance requirements and/or statistical characteristics of traffic sources, and bandwidth is allocated accordingly, thereby restricting the statistical multiplexing only within each class. This approach is called service separation [5]. In this way, it can satisfy each connection's cell loss probability requirement instead of guaranteeing an average CLP over all connections, also it eases the decomposition of a very complex overall control task into smaller and somehow independent problems. This philosophy has bee adopted in various forms in a number of works in the literature. Figure 2.1 illustrates the decomposition of a single queuing system with heterogeneous traffic sources.

By combining the service separation and the complete partitioning policy, Raffaele Bolla [3] proposed several bandwidth allocations and related CAC strategies. All the strategies can be interpreted as defining partitions of the region in call space with which QoS constraints at the cell level are satisfied. Moreover, they belong to the family of complete partitioning strategies, where the bandwidth of the outgoing link is entirely divided among the classes in a static way.

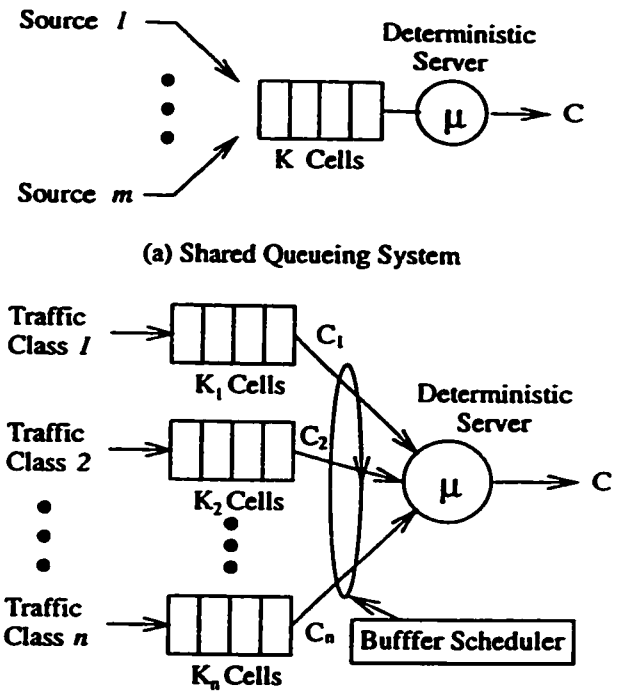

Figure 2. 1 ATM multiplexer queuing model

First of all, it divides the total traffic into classes. Each class is characterized by statistical parameters like peak rate, average rate, average burst length, as well as QoS requirements, like cell loss probability and cell delay. Each class is assigned a separate buffer and the total capacity of outgoing link is divided into virtual partitions among the classes.

Given the cell level constraints, i.e., the constraints on the cell loss probability and cell delay, the maximum number of connections that each class can support is obtained, and further it determines the boundary of the feasibility region.

Then four different CP schemes and corresponding CACs are proposed to compute the optimal number of connections and corresponding capacity for a class. Each

method is based on the minimization of a specific cost function, such as call blocking probability or overall cell loss rate, under a set of constraints. The result of minimization gives a 'rectangular' sub-region with a vertex on the boundary of the feasibility region. The problem with this scheme is that connection requests of a certain class may be rejected when their number of connections exceeds the optimal even though other classes may be under-utilizing the bandwidths allocated to them. Bolla attempted to alleviate the problem by soft partitions, i.e., all schemes have been embedded within an overall adaptive control architecture, whereby parameters are recomputed to adapt to load variations.

The admission of a new connection only involves a comparison of the current number of connections plus the new one with the optimal value.

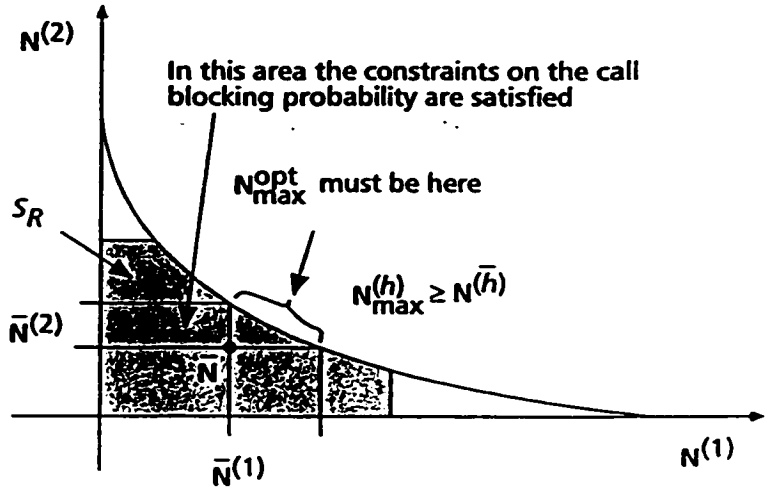

Figure 2. 2 Acceptance region for two traffic classes

The advantage of this scheme is the simplicity at connection acceptance. Complicated calculations and simulations can be executed off-line and the results can be stored in a table. However, the pre-computed tables require to be stored in the

multiplexer, the storage required increases with the number of classes. Besides, the computed values may not track the real-time traffic statistics very well, resulting in low utilization.

Another dynamic bandwidth allocation and related CAC scheme, called Minimum Overflow Traffic Algorithm (MOTA), which was proposed by H. Zhou [4], also adopts the philosophy of service separation. It groups the traffic sources into classes, but uses virtual cell loss probability as QoS requirements for each class. MOTA defines an overflow traffic function and uses it as the bandwidth assignment optimization function.

The overflow traffic function measures the ratio of the mean blocked peak traffic to the allocated bandwidth over the outgoing link for all the traffic classes. It is expressed in the following equation:

$$Q = \min \sum_{i=1}^{M} \sum_{n_i \in \{R_i - b_i > 0\}}^{n_i = L_i} p_i(n_i)\left(\frac{\sigma_i R_i - b_i}{b_i}\right) \tag{2-4}$$

s.t.

$$\sum_{i=1}^{M} b_i \le C$$

$$VCLP(i) < VCLP_{\max}(i), \quad i = 1, 2, \ldots, M$$

Where,

M is the number of traffic classes,

R_i is the peak arrival rate for the ith class,

$p_i(n_i)$ is the probability that n_i out of L_i connections are active, L_i is the number of connections in the ith class,

b_i is the bandwidth share for the ith class,

σ_i is a weighting coefficient.

The bandwidth assignment will be adjusted by optimizing the above function each time when there is a connection request. They also proposed a scheme to obtain the approximated optimal value for the cost function.

Both approaches proposed in [3] and [4] have been made in the context of service separation, where traffic sources have been grouped into classes, homogeneous in terms of performance requirements and statistical characteristics. Even though it guarantees each connection's QoS requirement, it may result in a low statistical multiplexing gain.

2.2 An Review of Cell Loss Rate Estimate Approaches

The one objective of traffic control is to make the network achieve performance objective associated with user requirements for communication quality. Typically, QoS is given in terms of the cell loss rate and cell delay. In this investigation, we consider the cell loss rate as the only QoS requirement.

In this section, we give an overview of some approaches of estimating cell loss rate. Then we looked at those interesting works having been done regarding call admission control.

Due to statistical multiplexing characteristics of ATM technology, at an ATM multiplexer, when more cells than the queue can handle will compete for this queue, cells will be lost. Cell loss rate is the proportion of cells lost at the entrance of the multiplexer due to buffer overflow, to total cells arriving to the multiplexer. It is not only an

important parameter of QoS requirements, but also a significant traffic performance measure needed for ATM network design and traffic management such as call admission control.

The exact calculation of cell loss rate is difficult because the aggregate arrival process is a superposition of traffics with diverse characteristics, and the arrival rates among successive time intervals are highly correlated, and also because the measures of interest include very small probabilities or cell loss rate. Various approaches have been suggested to evaluate the performance of ATM multiplexer. Some of these schemes which are based on some probabilistic representation of the traffic sources, include direct studies of the discrete process, and their continuous approximation such as flow fluid and diffusion approximation. Some only require traffic parameters such as peak bit rate and average bit rate.

A well-studied and established approach is the Gaussian approximation based on the zero-buffer assumption. It is aimed at taking advantage of statistical multiplexing gain when there is a large number of sources. In this approach, each connection is characterized by its average bit rate and its standard deviation; and the aggregate traffic is approximated by a Gaussian process with mean rate λ and variance σ^2. Two important estimates have been calculated using the Gaussian approximation:

- Overflow probability [1]

$$\Pr(overflow) = \Pr(R(t) \geq C) \approx \frac{1}{\sqrt{2\pi}} e^{-\frac{(\lambda - C)^2}{2\sigma^2}} \qquad (2\text{-}5)$$

- Upper bound on cell loss probability [6]

$$\Pr(loss) = \frac{E[(R(t)-C)^+]}{\lambda} \le \frac{\sigma}{\lambda\sqrt{2\pi}} e^{-\frac{(\lambda-C)^2}{2\sigma^2}} \tag{2-6}$$

Where $R(t)$ is the instantaneous cell arrival rate.

Despite its simplicity, there are some drawbacks in this approach. First, the Gaussian assumption does not hold when the number of sources being multiplexed is small. This approximation improves as the number of sources increases. Second, for a non-zero buffer system, the buffer's capacity to absorb traffic bursts is ignored in this Gaussian approximation.

Saito [7] proposed an upper bound for the cell loss probability based on the average number of cells that arrive during a fixed interval (*ANA*), and the maximum number of cells that arrive during the same fixed interval (*MNA*). The fixed interval was taken to be equal to $D/2$, where D is the maximum admissible delay in a buffer. Using these parameters, the following upper bound was derived. Let us consider a link serving N connections, and let $p_i(j)$, $i = 1, 2, \ldots, N$, and $j = 0,1, \ldots$ be the probability that j cells belonging to the ith connection arrive during the period $D/2$. Then the cell loss probability *CLP* can be bounded by

$$CLP \le B(p_1, \ldots, p_N; D/2) = \frac{\sum_{k=0}^{\infty} [k - D/2]^+ \, p_1 * \cdots * p_N(k)}{\sum_{k=0}^{\infty} k p_1 * \cdots * p_N(k)} \tag{2-7}$$

Where * is the convolution operation. Let $\theta_i(j)$ be the following functions:

$$\theta_i(j) = \begin{cases} ANA_i / MNA_i, & j = MNA_i, \\ 1 - ANA_i / MNA_i & j = 0, \\ 0, & otherwise. \end{cases}$$

Then it can be shown that

$$CLP \le B(p_1,\ldots,p_N;D/2) \le B(\theta_1,\ldots,\theta_N;D/2)$$
$$= \frac{\sum_{k=0}^{\infty}[k-D/2]^{+}\theta_1 *\cdots*\theta_N(k)}{\sum_{k=0}^{\infty}k\theta_1 *\cdots*\theta_N(k)} \qquad (2\text{-}8)$$

A new connection is admitted if the resulting $B(\theta_1,\ldots,\theta_{N+1};D/2)$ is less than the admissible cell loss probability. Satio also proposed a scheme for calculating the convolution efficiently.

B. Jabbari and F. Yegenoglu [8] also proposed an upper bound on the cell loss probability. It characterizes the probability of exceeding capacity in terms of cell loss. Rather than giving any assumption on the aggregate traffic, it divides the traffic sources into classes. Each source in a class has the same traffic characteristics and is described by its peak bit rate a_i and average bit rate r_i. Let L be the number of classes. Using a multi-dimensional birth-death process with each dimension denoting the population n_i of active sources for class i, the joint distribution of the population of each class can be obtained.

$$p(n_1,\ldots,n_L) = \begin{cases} \eta\prod_{i=1}^{L}p(n_i), & for \quad \sum_{i=1}^{L}n_i r_i \le \alpha C \\ 0, & else \end{cases} \qquad (2\text{-}9)$$

Where,

$$p(n_i) = \binom{N_i}{n_i} d_i^{n_i}(1-d_i)^{N_i-n_i}$$

$$d_i = \frac{r_i}{a_i}$$

Then the cell loss probability can be given as:

$$P(loss) = \frac{\sum_{n_L \cdots n_1, n_i \in \sum n_i a_i > C} P(n_1, \ldots, n_L)(\sum_{i=1}^{L} n_i a_i - C)}{\sum_{i=1}^{L} N_i r_i} \qquad (2\text{-}10)$$

The above cell loss probability does not take into account buffering, so this probability is only and upper bound on the cell loss.

The CAC scheme proposed by H. Esaki, mentioned earlier in the previous section, determines the required bandwidth through the estimate of overall cell loss rate. The method in estimating cell loss rate uses probability function for the number of cells transferred from the multiplexed connections. The amount of calculation for this algorithm is reduced remarkably even in large diversity of connection types. This methodology is adopted in the investigation and the details are given next.

2.3 Cell Loss Rate Estimate Algorithm

In what follows, the methodology to evaluate the cell loss rate proposed by H. Esaki is described.

The estimated cell loss rate in the algorithm here is based on the bufferless model assumption and is the average cell loss rate over all connections.

At an ATM multiplexer, the total traffic is divided into groups based on their peak cell transmission speed, i.e. low speed connections and high speed connections. As well known, the beneficial effect of statistical multiplexing falls off with the increase of peak cell transmission speed for the multiplexed connections. By the computer simulation, it could be said that the statistical multiplexing effect decreases when the peak cell transmission speed of the multiplexed connection is larger than around 1/100 of the cell

transmission speed of multiplexed line [2]. Therefore, 1/100 is defined as the threshold value whether a multiplexed connection is low speed connection or high speed connection in this algorithm. When the ratio of the peak transmission speed of the connection to the cell transmission speed of the multiplexed line is larger than the threshold value, we can expect small statistical multiplexing gain. On the other hand, when the ratio of the peak transmission speed of the connection to the cell transmission speed of multiplexed line is smaller than the threshold, we can expect enough large statistical multiplexing gain.

The traffic characteristic for a multiplexed connection is described by its peak cell transmission speed V_p and average transmission speed V_a. Each connection is modeled as On/Off type. During On period, cells are generated at the peak transmission speed of the connection; during Off period, there is no cell generated. V_{pl} is the maximum cell transmission speed of the low speed connections, and taken to be equal to $V/100$, where V is the cell transmission speed of the multiplexed line. A fixed interval T_l is also defined as the inverse of V_{pl}. The cell loss rate is estimated as follows based on the number of cells transferred from each connection during this interval.

The following bufferless model is assumed. During T_l period, M_{Tl} is the number of cells being transferred through the multiplexed line. When the number of cells that are transferred from high speed connections and low speed connections are larger than M_{Tl} cells, these cells beyond M_{Tl} are discarded.

In order to evaluate the cell loss rate for this bufferless model, the probability function for the number of transferred cells from each connection during T_l is obtained.

Let $G(n)$ be the probability that n cells transferred from the multiplexed connections during T_l, then $G(n)$ can be defined as follows:

$$G(n) = \sum_{k=0}^{n} p(k) F(n-k) \tag{2-11}$$

Where, $p(k)$ means the probability that k cells are transferred from low speed connections; $F(k)$ means the probability that k cells are transferred from high speed connections during T_l.

By assuming that each low speed connection is modeled as transferring cells at the speed of V_{pl}, which is the maximum peak cell transmission speed of low speed connections, the probability $p(k)$ can be derived as follows:

$$p(k) = \sum_{k=\sum_{i=1}^{L_l} n_i} \prod_{j=1}^{L_l} \binom{N_j}{n_j} \alpha_j^{n_j} (1-\alpha_j)^{N_j - n_j} \tag{2-12}$$

$$\alpha_j = V_{aj} / V_{pl}$$

Where

L_l is the number of low speed connection types,

N_j is the number of type j low speed connections $(1 \le j \le L_l)$,

n_j is the number of type j low speed connections that are in active state.

V_{aj} is the average cell transmission speed of a type j low speed connection,

α_j is the probability that a cell is generated during T_l period from a type j low speed connection.

On the other hand, $F(k)$ is the probability that k cells are transferred from high speed connections during period T_l. Since T_l is larger than the inverse value of the peak

cell transmission speed for a high speed connection, a high speed connection can transfer more than one cell during T_1. In order to calculate the cell loss rate in a conservative manner, the worst case cell transmission pattern is assumed for high speed connections. When $F(k)$ represents the probability of worst case transmission pattern, the estimated cell loss rate is surely larger than the actual cell loss rate. The worst case cell transmission pattern is the case where a high speed connection transfers cells continuously at the speed of its peak cell transmission speed V_{ps} during T_1. Since each connection keeps its average transmission speed of V_{as}, the probability that a connection type s transfers with the worst case pattern is V_{as}/V_{ps}. Then, $F(k)$ under the worst case mentioned above is given by Equation (2-13). Here $Q_s(n_s)$ is the probability that n_s cells are transferred from type s connections during T_1.

$$F(k) = \sum_{k=n_1+n_2+\ldots+n_{L_h}} \prod_{s=1}^{L_h} Q_s(n_s) \tag{2-13}$$

$$Q_s(n_s) = \begin{cases} \binom{N_s}{l} \alpha_s^l (1-\alpha_s)^{N_s-l} & (n_s = l\lceil V_{ps} T_1 \rceil) \\ 0 & (otherwise) \end{cases} \tag{2-14}$$

Where,

L_h is the number of high speed connection types.

N_s is the number of type s high speed connections.

V_{ps} is the peak cell transmission speed of a type s high speed connection,

V_{as} is the average cell transmission speed of a type s high speed connection,

α_s is the probability that a type s connection transfers with the worst case pattern.

$\lceil x \rceil$ means the smallest integer equal to or greater than x.

Then the estimated cell loss rate is given by Equation (2-15):

$$CLR = \frac{1}{\rho M_{T1}} \sum_{n=M_{T1}+1}^{\infty} (n - M_{T1}) G(n) \tag{2-15}$$

Where,

ρ is the average offered load to the multiplexed line, $\rho = \rho_l + \rho_h$;

ρ_l is the average offered load from low speed connections, $\rho_l = \sum_{j=1}^{L_l} \frac{N_j V_{aj}}{V}$;

ρ_h is the average offered load from high speed connections, $\rho_h = \sum_{s=1}^{L_h} \frac{N_s V_{as}}{V}$.

Equation (2-15) can be transferred as Equation (2-16), reducing the summation from $[M_{TI}, \infty]$ to $[0, M_{TI}]$,

$$CLR = \frac{1}{\rho M_{T1}} \sum_{n=0}^{M_{TI}} (M_{T1} - n) G(n) - \frac{1-\rho}{\rho} \tag{2-16}$$

The above estimated cell loss rate is an upper bound on the real value. The conservatism of this algorithm comes from three aspects, in what follows, we give a little discussion and further some modification.

The first aspect of conservatism is from its bufferless model. For a non-zero-buffer system, the buffer's capacity to absorb traffic bursts is ignored in this algorithm. An improved method was proposed by Z. Shi to take into account buffer effects [10]. In this thesis, we still deploy the original algorithm to reduce the computation complexity.

The other two aspects are from the way of computing cell loss for low speed connections and high speed connections. We already know that each low speed

connection transfers cells at their maximum peak rate, then it is quite conservative in case that low speed connections occupy a large fraction of total traffic. For high speed connections, a worst case transmission pattern is assumed, i.e. a high speed connection transfers cells continuously at its peak rate. It should not contribute too much compared to low speed connections, but the Equation (2-14) for computing $Q_s(n_s)$, the probability that n_s cells are transferred from type s connections during T_1, could give a big margin in some special cases. To avoid this happening, a modification on the formula is introduced as follows:

$$Q_s(n_s) = \begin{cases} \binom{N_s}{l} \alpha_s^l (1-\alpha_s)^{N_s - l} & (n_s = \lceil l\, V_{ps} T_1 \rceil) \\ 0 & (\textit{otherwise}) \end{cases} \qquad (2\text{-}17)$$

Chapter 3 Dynamic Call Admission Control Under Partial Separation

In this chapter, we present our proposed call admission control scheme with partial separation implementation that dynamically allocates bandwidth of the outgoing link.

3.1 Structure of bandwidth allocation system

To derive the appropriate bandwidth allocation strategy to be applied by the CAC procedure, at an ATM multiplexer, we first need a model to characterize traffic sources. Figure 3.1 illustrates the decomposition of a single queuing system with heterogeneous traffic based on partial separation.

We suppose the traffic at the multiplexer to be divided into H classes of on/off sources, each source being characterized by statistical parameters like peak rate and average rate, as well as QoS requirements like cell loss rate. In this investigation, we choose cell loss rate as the measurement of QoS requirements. Based on the notion of partial separation, sources have similar cell loss rates, but may have different traffic characteristics within a traffic class. We indicate with V_p the peak rate and average rate V_a for a traffic source respectively. $\varepsilon^{(h)}$ is the upper limit on the cell less rate for class h.

At the ATM multiplexer, traffic class h is assigned a separate buffer of length $Q^{(h)}$ cells, whose output is statistically multiplexed on the outgoing link by a scheduler, which substantially divides the global channel capacity V into virtual partitions $V^{(h)}$ among the

classes, whose sum amounts to V. The simplest way to maintain the partitions is by serving the buffer in a weighted round-robin fashion; other possibilities include the assignment of a slot (time to transmit a cell) to a cell of class h randomly, with probability $V^{(h)}/V$.

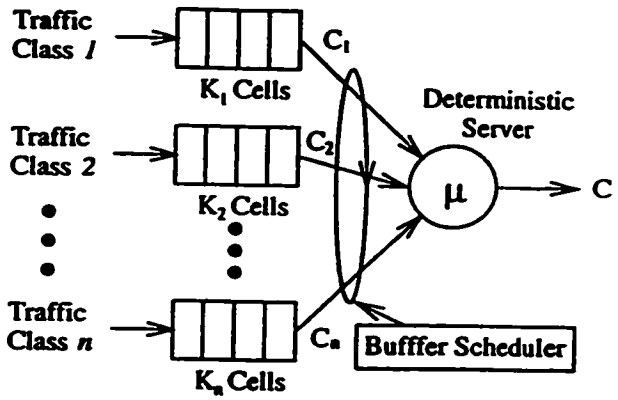

Figure 3. 1 an ATM multiplexer implementing partial separation

To simplify the bandwidth allocation problem, we have chosen, in this investigation, not to assign the buffer length dynamically, they are determined a priori and never changed.

3.2 Dynamic Bandwidth Allocation Strategy

To reach the bandwidth allocation goal, the virtual capacity $V^{(h)}$ is dynamically assigned by means of a procedure that minimizes a suitable cost function. The cost function to be minimized is chosen to take into account the expected number of lost cells, pertaining to the whole offered traffic. The structure of the cost function has been taken as:

$$J(V^{(1)}, V^{(2)}, \ldots, V^{(H)}) = \sum_{h=1}^{H} \sigma^{(h)} CLR^{(h)}(V^{(h)}) \tag{3-1}$$

Where,

$\sigma^{(h)}$, h=1, 2,..., H, is the weighting coefficient.

$CLR^{(h)}(V^{(h)})$ is the incurred cell loss rate under the current load for traffic class h.

For the choice of the weighting coefficients, it can be made in order to reflect the relative importance attributed to the various traffic classes and the possibly largely different scales of cell loss rates.

In the minimization of the cost function expressed in Equation (3-1), a set of inequality constraints must be taken into account, namely that is:

$$\sum_{h=1}^{H} V^{(h)} \leq V \tag{3-2}$$

$$CLR^{(h)}(V^{(h)}) \leq \varepsilon^{(h)} \tag{3-3}$$

Where,

$\varepsilon^{(h)}$, h=1, 2,..., H, is the upper limit on the cell loss rate for class h.

To compute the cell loss rate $CLR^{(h)}(V^{(h)})$ for class h given allocated bandwidth $V^{(h)}$, we deploy the methodology described in Chapter 2, that is,

$$CLR^{(h)}(V^{(h)}) = \frac{1}{\rho^{(h)} M_{T1}^{(h)}} \sum_{n=0}^{M_{T1}^{(h)}} (M_{T1}^{(h)} - n) G^{(h)}(n) - \frac{1-\rho^{(h)}}{\rho^{(h)}} \tag{3-4}$$

Where,

T_1 is a fixed interval, and is taken to be equal to the inverse of $V^{(h)}/100$.

$M_{T1}^{(h)}$ is the number of cells transferred from the buffer of class h during T_1.

$\rho^{(h)}$ is the offered load from traffic class h with respect to $V^{(h)}$.

$G^{(h)}(n)$ is the probability that n cells transferred from the multiplexed connections of class h during T_1.

The reader is referred to Chapter 2 for details about the calculation of cell loss rate.

3.3 Optimization Concepts and Techniques

Before we step into the optimization details of objective function given in section 3.2, some basic optimization concepts and techniques are presented first.

While analytical methods have been developed for handling optimization problems with objective functions simulated from analytical models, these methods cannot be applied in most practical cases, owing to the complexity of the required mathematical manipulations or the discontinuous nature of the functional relationships. Numerical search techniques are found to handle some optimization problems.

The general organization of the search for an extreme value can be summarized as a sequential search involving the successive calculation of new values of the objective function and the comparison of these values with the best value that has been so far obtained. With this iterative procedure, the basic difference in the available methods lies in the philosophy dictating the choice of the next location for objective function evaluation.

In any logical method, the choice of the new location for evaluation of the objective function is of major importance. The procedure for making such a choice is often described in terms of search strategy. In general, once a feasible base point is

chosen, a set of exploratory experiments is carried out in the vincinty of this base point. This exploration is made, initially, to study the behavior of the objective function in the neighborhood of the starting location, thus providing information about those directions of movement that might yield favorable results.

Therefore, the two essential features characterize the various available methods: one important part of any search method is the choice of a direction of movement to a better region. If the direction can be chosen only from a finite number of choices, the direction of movement will be in a favorable direction only, and not necessarily lying along the line of greatest improvement.

The second essential feature of any search is the distance of movement along the chosen direction. Only at this stage that possible improvement could occur in the values of the objective function, all previous work being required to lay the basis for efficient movement. The move in this selected direction can be a single step, or alternatively, a series of steps, in which the objective function is tested at each step, the movement in this direction being continued as long as the objective function improves.

Using the final location of the previous move as the new base, the cycle of exploratory experiments, choice of direction for movement, and movement are repeated as long as it is required. In each cycle of the search the value of the objective function will improve, or at least remain constant. Eventually, no further improvement can be obtained. Further optimization might be attempted by reducing the step size in the search. The search for the desired optimum will then end if it has been located within the

required degree of accuracy or if changes in its value have been found to fall within some pre-selected fraction of its value.

Two separate types of search method may be distinguished, those in which only the function is evaluated and those in which both the function and its first derivatives (the gradient) are evaluated about any given point. These two groups are classified as direct and gradient methods. Direct search methods require only the evaluation of the objective function at a particular location. This is in contrast with the gradient techniques, which require both function and gradient values to be evaluated at any position. It is therefore recommended that a gradient method be adopted, provided that suitable analytical expressions are available for defining the gradient.

While gradient methods are generally more effective than direct search methods, there are situations, for example, when the function gradient is not analytically defined, in which the direct search methods are more appropriate. These methods vary from the elementary uni-variant form to the sophisticated Powell and Rosenbrock techniques [9].

Above techniques can be directly applied to unrestricted objective functions. However, in reality, a more general type of optimization problem is that in which the objective function $y(x)$ is to be optimized subject to m inequality constraints,

$$g_k(x) \leq 0 \text{ for } k=1, 2, 3,\ldots,m \tag{3-5}$$

The problem may be tackled by one of two approaches.

First, the search for optimum subject to inequalities may be carried out using the unrestricted numerical search methods. However, an additional routine must be added to ensure that each new point is a feasible one and to direct the search in a feasible direction

should a non-feasible point be found. The second group of methods utilizes a modified objective function, which includes the constraints as an integral part, and therefore permits an unrestricted search.

The most common method combines the objective function and constraints in a single modified function by the use of penalty factors. In a sense, the penalty function method alters both the objective function and the constraints

Penalty functions offer a reasonable method of converting a restricted problem into one that is unrestricted by including the constraints in a modified objective function. There are two basic approaches. First, we may form, after Kelley [9], the modified objective function:

$$F = y + \sum_{k=1}^{m} P_k g_k^2 H(g_k) \tag{3-6}$$

Where,

$H(g_k)$ is the Heavyside unit step function defined so that

$$H(g_k) = \begin{cases} 1 & for\ g_k \geq 0 \\ 0 & for\ g_k < 0 \end{cases} \tag{3-7}$$

P_k are large penalty constants which are positive if seeking a minimum and negative for maximization.

Carroll has proposed an alternative penalty function formula that introduces a natural optimum within the feasible region [9],

$$F = y - P\sum_{k=1}^{m} \frac{1}{g_k} \tag{3-8}$$

3.4 Implementation of Optimization Technique

The objective function presented in section 3.2 is a general form where we divide the total traffic to n classes based on the QoS requirements. To simplify the discussion of this section, we take a 3 traffic classes as an example to illustrate the implementation of the optimization technique, therefore to obtain the optimized partition of an outgoing link capacity among classes.

The objective function can be expressed as follows in the case of 3 traffic classes:

$$J(V^{(1)},V^{(2)},V^{(3)}) = \sum_{h=1}^{3} \sigma^{(h)} CLR^{(h)}(V^{(h)}) \tag{3-9}$$

s.t.

$$CLR^{(h)}(V^{(h)}) \le \varepsilon^{(h)} \qquad (h = 1,2,3) \tag{3-10}$$

$$\sum_{h=1}^{3} V^{(h)} = V \tag{3-11}$$

By taking the constraint (3-11) into the objective function, Equation (3-9) can be modified further as a two-dimensional optimization problem,

$$\begin{aligned} J(V^{(1)},V^{(2)}) = {} & \sigma^{(1)} CLR^{(1)}(V^{(1)}) + \sigma^{(2)} CLR^{(2)}(V^{(2)}) + \\ & \sigma^{(3)} CLR^{(3)}(V - V^{(1)} - V^{(2)}) \end{aligned} \tag{3-12}$$

s.t.

$$CLR^{(1)}(V^{(1)}) \le \varepsilon^{(1)} \tag{3-13}$$

$$CLR^{(2)}(V^{(2)}) \le \varepsilon^{(2)} \tag{3-14}$$

$$CLR^{(3)}(V - V^{(1)} - V^{(2)}) \le \varepsilon^{(3)} \tag{3-15}$$

To reach the goal of optimization the above objective function so as to allocate the bandwidth among the classes, we adopt Kelley's method of adding penalties [9], in conjunction with the Powell's unrestricted search method to rapidly locate the approximate location of the optimum. Let y be the modified objective function:

$$Y = J + \sum_{k=1}^{3} P_k g_k^2 H(g_k) \tag{3-16}$$

Where,

$$g_1 = CLR^{(1)}(V^{(1)}) - \varepsilon^{(1)} \tag{3-17}$$

$$g_2 = CLR^{(2)}(V^{(2)}) - \varepsilon^{(2)} \tag{3-18}$$

$$g_3 = CLR^{(3)}(V - V^{(1)} - V^{(2)}) - \varepsilon^{(3)} \tag{3-19}$$

$H(g_k)$ is the Heavyside unit step function defined so that

$$H(g_k) = \begin{cases} 1 & for\ g_k \geq 0 \\ 0 & for\ g_k < 0 \end{cases} \tag{3-20}$$

P_k are large penalty constants which are positive if seeking a minimum and negative for maximization.

After the problem is modified so as to include the constraints within a modified objective function, using Kelley's method, Powell's search method can be carried out to find the optimum.

Powell's method, as one of the most widely accepted direct search method, is based on the concept of quadratic convergence. It determines a new search direction, using the constructional procedure depicted in Figure 3.2, but questions the value of using this new direction rather than blindly introducing it into the list of search directions

while rejecting another. There are two inequalities to be tested. The Powell's method can be described as follows for our two-dimensional search.

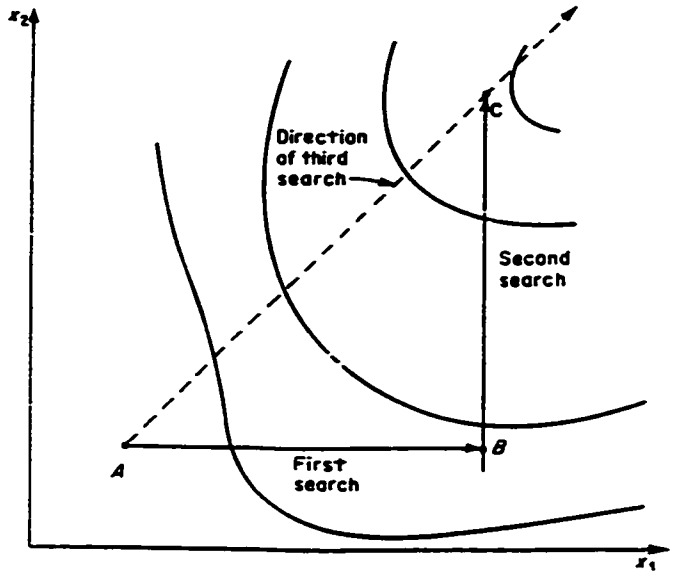

Figure 3. 2 an acceleration in direction of Powell's search method

Step 1: starting with the best previous value position $x_0^{(k)}$ (base point), $x_0^{(k)}$ here represents a pair of $V^{(1)}$ and $V^{(2)}$, and two linearly independent directions of search, $\xi_1^{(k)}$, $\xi_2^{(k)}$, then we carry out a univariant search by finding the position of the optimum along the direction $\xi_1^{(k)}$. At this optimum point $x_1^{(k)}$, continue a second search in the direction $\xi_2^{(k)}$ to get to $x_2^{(k)}$.

Step 2: We now find the particular point $x_m^{(k)}$ (m =1,2) for which the greatest improvement of the objective function over its previous value is realized. The point $x_m^{(k)}$ there for yields the largest change, Δ, of any the 2 moves, where $\Delta = \left| y(x_m^{(k)}) - y(x_{m-1}^{(k)}) \right|$. Also determine the vector $\mu = x_n^{(k)} - x_0^{(k)}$.

Step 3: Determine $y(2x_n^{(k)} - x_0^{(k)}) = y_t^{(k)}$.

Step 4: If, in seeking a minimum,

$$y_t^{(k)} \geq y_1^{(k)} \tag{3-21}$$

And/or

$$(y_0^{(k)} - 2y_n^{(k)} + y_t^{(k)})(y_0^{(k)} - y_n^{(k)} - \Delta)^2 \geq \frac{\Delta(y_0^{(k)} - y_t^{(k)})^2}{2} \tag{3-22}$$

Then μ is not a good direction to introduce into our search, and we begin the search again, starting at the last point and using the same directions, step 1 is repeated. If neither of these inequalities is satisfied, search along the direction μ until the minimum is found. This point is defined as $x_0^{(k+1)}$ and the new search directions for the (k+1) stage are $\xi_i^{(k+1)} = \xi_i^{(k)}, i = 1, 2, \ldots, m-1$; $\xi_i^{(k+1)} = \xi_{i+1}^{(k)}, i = m, \ldots, n-1$; $\xi_n^{(k+1)} = \mu$ (n=2). We then repeat the entire process, starting with Step 1.

The Powell's search method has been implemented and the source code is listed in the Appendix B.

3.6 Call admission Control Procedure

The procedure we describe here is based on the fact that a connection will only be established if there is enough bandwidth available on the outgoing link, and the capacity of outgoing link is allocated among the classes.

Assuming that weighting coefficients for each class are chosen, and the upper bounds of cell loss rate are given as well.

1. Choose the base point for the search. Use the previous optimal bandwidth assignment set as the base point. If there is not (when the system starts up), use the average rate of all connections in a class instead.
2. Search for the new bandwidth assignment set when connection requests acceptance or leaves.

Chapter 4 Case Studies

In this chapter some case studies have been carried out on the cell loss rate estimate algorithm [2] and the proposed CAC scheme.

4.1 Traffic Model at an ATM Multiplexer

Over the last few years, we have gone through several paradigm shifts regarding our understanding of how to model an ATM source. Following the first performance models, based on the Poisson or Bernoulli assumption, it became apparent that these traffic models did not capture the burstiness present in traffic resulting from applications such as moving a data file and packetized encoded video.

Thus, there was a major shift towards using distributions of the on/off type, such as the interrupted Poisson process (IPP) or its discrete-time counterpart, the interrupted Bernoulli process (IBP). In an IPP, there is an active period during which arrivals occur in a Poisson fashion, followed by an idle period during which no arrivals occur. These two periods are exponentially distributed, and they alternate continuously. An IBP is defined similarly, only the arrivals during the active period are Bernoulli distributed and the two periods are geometrically distributed. An IPP or IBP, however, does not capture the notion of correlation since successive interarrival times are independent of each other. Another way of describing a source is using the fluid approach in which arrivals occur at a continuous rate during the active period. This defines an on/off fluid source, i.e. an interrupted fluid process (IFP). More complex distributions were introduced for modeling ATM traffic. These distribution are in the form of a Markov modulated Poisson Process

(MMPP), its discrete-time counterpart, a Markov modulated Bernoulli process (MMBP), or a Markov modulated fluid process (MMFP). Much of the work on ATM traffic analysis and cell loss rate estimates is based on the on/off traffic model and on the superposition of such traffic streams. Thus it is of particular interest to evaluate the accuracy of the algorithm described in chapter 2 for this specific class of practically useful models.

Consider a traffic source whose traffic follows a simple on/off behavior, particularly an IBP. It either sends traffic into the multiplexer at a constant rate during the ON period, or it sends no traffic at all during the Off period. The following notation describes this traffic model:

- peak rate during the On period;
- average length of the On period;
- average length of the Off period;
- source activity.

The length of both On and Off periods are geometrically distributed random variables with parameters a and b respectively. The durations of the successive on and off periods are assumed to be independent, so that the cell arrival process from a single source is a renewal process.

4.2 Simulation Modeling of an ATM Multiplexer

In the simulation, each source is modeled as an IBP as we described in the previous section. The multiplexer considered output-buffer queuing system takes the total traffic as the input, and its output is multiplexed onto an outgoing link. Each source

generates cells during On period and no cells during Off period. The burst length m_a and the idle length m_i are modeled as independent geometrically distributed random variables having a certain mean value M_a and M_i, which represent the average burst length and average idle length respectively. The distribution for those random variables are given as follows:

$$P(m_a = n) = p^n q \qquad (n = 1,2,3,\ldots) \tag{4-1}$$

$$P(m_i = n) = pq^n \qquad (n = 1,2,3,\ldots) \tag{4-2}$$

Where,

p is the probability that a cell is generated;

q is the probability that no cell is generated; and $p + q = 1$;

the average burst length M_a and average idle length M_i are equal to $1/q$ and $1/p$ respectively.

Since the Geometrical distribution for the burst length ma and idle length has been specified, the method is sought to generate cell streams with this distribution to be used as input to a simulation model. Since Matlab does not have geometric distributed random variable generating function, a subroutine is written to serve this purpose. The equation for generating a geometric distributed random variable, n, is given by the following equations:

$$n = \left\lceil \frac{\ln(1-u)}{\ln(1-P)} \right\rceil - 1 \tag{4-3}$$

Where,

u is the generated random number between 0 and 1,

P represents *q* for burst period, and represents *p* for idle period.

C language is used to obtain simulation results. The program is listed in Appendix A. The program is compiled using gnu compiler, and run on SUN SPARC 5 workstation.

4.3 Comparison of Cell Loss Rate Algorithm with Simulations

In this section we present the numerical and simulation results to evaluate the accuracy of the cell loss estimate algorithm. The validation is focused on the comparison of the cell loss rate predicted by the algorithm and that obtained by simulations for the on/off traffic model described in the previous section. Analytical results and simulation results are compared in the following figures. Table 4.1 lists the traffic sources used in this chapter.

Table 4. 1 Traffic sources used in numerical and simulation studies

	source 1	Source 2	source 3	Source 4
Activity (α)	0.1	0.2	0.4	0.5
Average rate (V_a(Mbits/s))	0.75	0.75	0.75	0.75
Peak rate (V_p(Mbits/s))	7.5	3.75	1.875	1.5
Burst length (# of cells)	230	230	230	230
Idle length (# of cells)	2070	920	345	230

In Figure 4.1, the cell loss rate, as calculated from the algorithm, is plotted and compared with the value measured from the simulation for heterogeneous traffic under different load. We have chosen two types of sources: source 1 with activity factor α=0.1 (more burstier); source 2 with activity factor α= 0.5 (see table for details). Here the load

is defined as the ratio of aggregate average rate to the outgoing link capacity. The outgoing link capacity is V = 150Mbits/s, and we did not take into account buffering in the simulation.

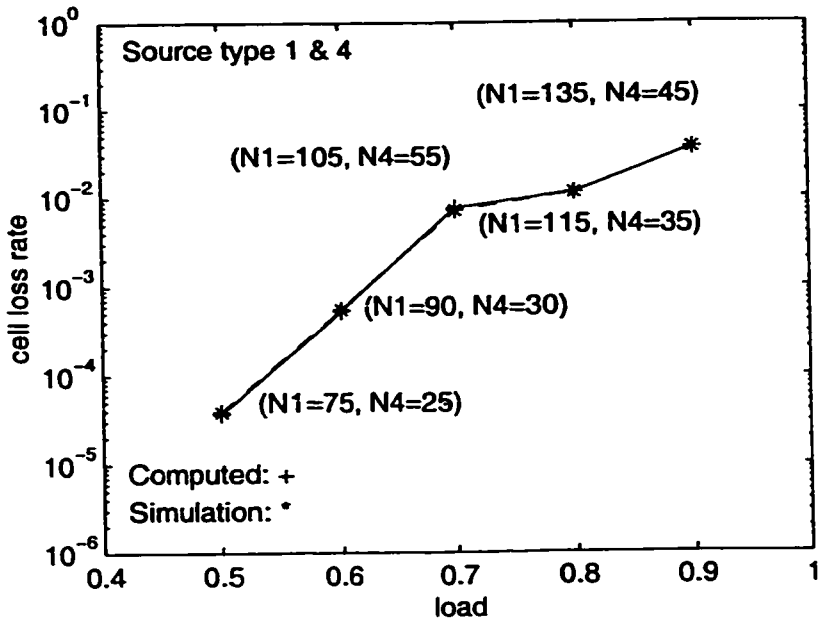

Figure 4. 1 Cell loss rate vs load

Figure 4.2 and 4.3 show cases with homogeneous traffic. Figure 5.2 compares the analytical results with the simulations under different loads by varying the source activity.

Figure 4.3 compares the results with simulation under variant peak rate while keeping the mean rate constant.

Excellent agreement between the algorithm predictions and simulations has been observed in the above figures. We see that the algorithm slightly overestimates the cell loss rate, which means that if we use this approach in call admission control it will provide accurate and conservative results.

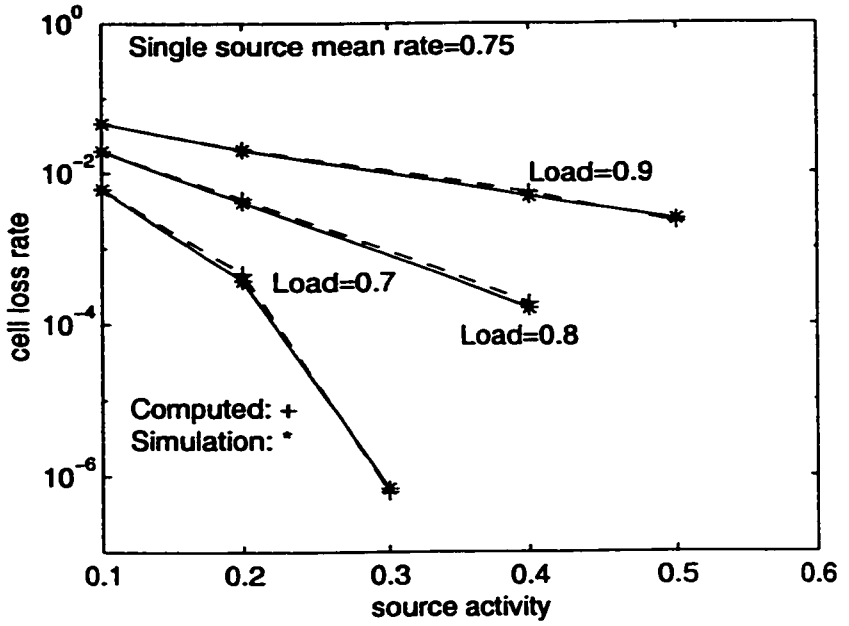

Figure 4. 2 Cell loss rate vs source activity

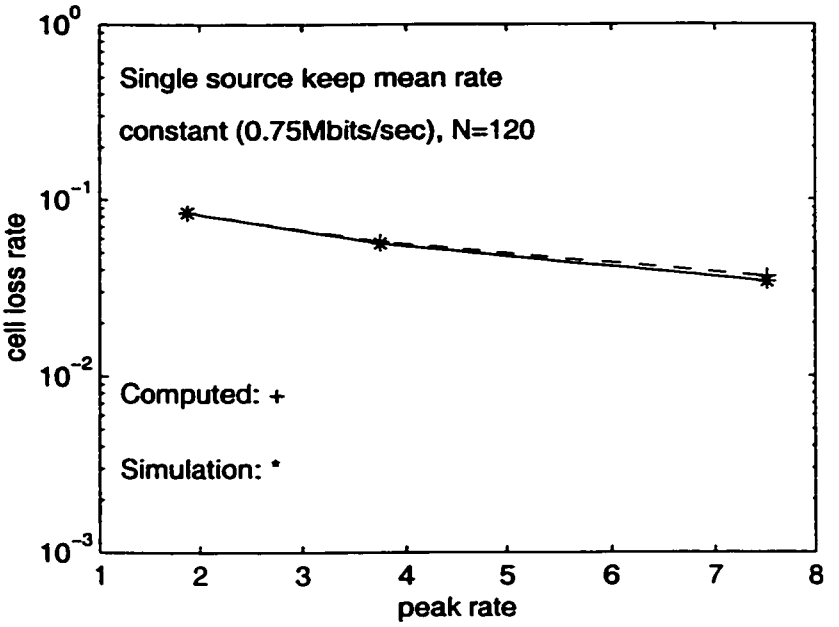

Figure 4. 3 Cell loss rate vs peak rate (keep mean rate constant)

4.4 Case Studies on the Proposed CAC Scheme

We evaluated our scheme by comparing it with the one proposed by H. Esaki [2]. At the meantime, it is aimed at showing the steps going through during the search for the optimal bandwidth assignment set when connection requests acceptance or leaves at the multiplexer.

The results are obtained with two types of traffic sources shown at the following table, and the total traffic is divided into 3 classes.

Table 4. 2 Traffic sources used in numerical studies

	Source 1	Source 2
Activity (α)	0.4	0.75
Average rate (V_a(Mbits/s))	0.75	0.75
Peak rate (V_p(Mbits/s))	1.875	1.0

Class 1 consists of connections of source 1 and the required cell loss rate for class 1 is upper-bounded by 1×10^{-6}. Class 2 is made up of those connections from source 2 and the upper-bound for the cell loss rate is 1×10^{-4}. The mixed traffic from source 1 and 2 is the input for Class 3, and the cell loss rate requirement is 1×10^{-3}. The overall channel capacity is 450 Mb/s.

The results are computed for the following cases.

Case 1: Number of connections in Class 1 is 150, number of connections in Class 2 is 150, and for Class 3, number of connections from source 1 is 75, from source 2 is 75 as well. We choose the average rate for each class as the base point to start the search,

that is the bandwidth assignment set is $(V_1, V_2, V_3) = (150, 150, 150)$. The following table shows the value of objective function and the corresponding bandwidth assignment set.

Table 4. 3 Computed results for case 1

	Value of objective function	Bandwidth assignment
Step 0	1.98×10^{-5}	(150, 150, 150)
Step 1	1.80×10^{-5}	(147, 150, 153)
Step 2	1.09×10^{-5}	(147, 153, 150)

With the same traffic load in case 1, we use the CAC method proposed in [2] to calculate the required bandwidth, it needs 495Mb/s to achieve the QoS requirements which in turn is the strictest one among the three classes, i.e. 1×10^{-6}.

Case 2: there is 10 more connection requests of source 1 in Class 1. We will search for the optimal bandwidth assignment set then to determine whether accepting or rejecting those requests. In this case we can use the result we obtained from Case 1 as the base point to start. The following table shows the value of objective function and the corresponding bandwidth assignment set. It tells that these 10 connections can be accepted while still guaranteeing each class's QoS requirements.

Table 4. 4 Computed results for case 2

	Value of objective function	Bandwidth assignment
Step 0	3.69	(147, 153, 150)
Step 1	1.89×10^{-4}	(159, 153, 138)
Step 2	7.41×10^{-5}	(159, 147, 144)

For the CAC method in [2], the total required bandwidth in case 2 is 505Mb/s to achieve the QoS requirements for all traffic sources.

Case 3: there is 10 connections in Class 2 leave at the multiplexer. The overall channel capacity is reallocated to give each class a better satisfaction. The following table shows the value of objective function and the corresponding bandwidth assignment set.

Table 4. 5 Computed results for case 3

	Value of objective function	Bandwidth assignment
Step 0	3.35×10^{-5}	(159, 147, 144)
Step 1	3.35×10^{-5}	(159, 147, 144)
Step 2	1.84×10^{-5}	(159, 143, 148)

Table 4. 6 Computed results for case 4

	Value of objective function	Bandwidth assignment
Step 0	62	(175.5, 143, 131.5)
Step 1	58	(175.5, 136.3, 138.2)
Step 2	18	(181.5, 136.3, 132.2)
Step 3	18	(181.5, 136.3, 132.2)

Case 4: Now, there is 40 more connection requests for class 1. We will search for the optimal bandwidth assignment set, then to decide whether accepting or rejecting those connection requests. Similar as other cases, we use the result obtained in case 3 as the base point to start the numerical search. The following table gives the value of objective function and the corresponding bandwidth assignment set. From the table, we can see that

the optimal bandwidth assignment set does not exist. Therefore, those connection requests got rejected.

Chapter 5 Conclusion

In this investigation we have reviewed some of call admission control schemes proposed in the literature. Our approach is based on the concept of partial separation, where traffic sources have been grouped into classes. Within each class, each source has similar QoS requirements, but may have different traffic characteristics. Then the CAC is formulated as an optimization problem where overall cell loss rate across the classes is minimized. In order to satisfy the cell level constraints for each class, we have chosen the cell loss rate estimate methodology proposed in [2]. Predictions has been presented and validated with extensive simulations with traffic resulting from the superposition of On-Off traffic models with varying degrees of burstiness. Simulation results have shown that the estimates are slightly conservative over the measured cell loss rate without considering buffer effect.

We also compared the computed results from the proposed scheme with that from the method proposed in [2]. It is what we have expected that required link capacity using [2] is higher than that using our proposed method since the system had to realize the most stringent QoS requirements in [2].

The cell loss rate estimate methodology does not take into account of burstiness and it could cause our CAC approach penalizes the very bursty traffic. Thus further research is needed to improve the algorithm where even burstier sources are not penalized by our proposed CAC scheme.

References

[1] R.O. Onvural, "Asynchronous Transfer Mode Networks: Performance Issues", *Artech House*, Boston(1993).

[2] H. Esaki, K. Wamura, and T. Kodama, "Connection Admission Control in ATM networks", *IEICE Trans. Commun.*, Vol. E77-B, No. 1 January 1994, pp15-26.

[3] R. Bolla, F. Davoli and M. Marchese, "Bandwidth Allocation and Admission Control in ATM Networks with Service Separation", *IEEE Communications Magzine*, May 1997, pp130-137.

[4] H. Zhou and C.H. Chang, "A New Dynamic Bandwidth Allocation Scheme for ATM Networks", *GLOBECOM'95*, Vol. 1, pp410-415.

[5] K.W. Ross, "Multiservice Loss Models for Broadband Telecommunication Networks", Springer.

[6] K.M. Rege, "Equivalent Bandwidth and Related Admission Control Criteria for ATM Systems - A Performance Study", *Int'l. J. Commun. Systems*, Vol. 7, 1994, pp181-197.

[7] H. Saito, "Call Admission Control in an ATM Network Using Upperbound of Cell Loss Probability", *IEEE Trans. Commun.*, Vol. 40, 1992, pp1512-21.

[8] B. Jabbari and F. Yegenoglu, "An Upper Bound for Cell Loss Probability of Bursty Sources in Broadband Packet Networks", *IEEE ICC'91*, Vol. 2, pp699-703.

[9] G. Beveridge and R. Schechter, "Optimization: theory and practice", New York, McGraw-Hill (1970).

[10] Z. Shi, "Connection Admission Control Methods for Multimedia Communication Systems", Master thesis, University of Ottawa, 1997.

Appendix A Source Code for Simulation

```
/*
* Name:    simu.h
*
* Purpose: define certain constants for cell lost rate simulation C-file
*
*/
#include <stdlib.h>
#include <math.h>
#include <malloc.h>

typedef unsigned char uint1;/* unsigned 1 byte integer*/
typedef unsigned short uint2;/* unsigned 2 byte integer*/
typedef unsigned long uint4; /* unsigned 4 byte integer      */
typedef char int1;/* signed 1 byte integer*/
typedef short int2;/* signed 2 byte integer*/
typedef long int4;/* signed 4 byte integer*/

#define min(x, y)   ((x > y) ? y : x);
```

```
/*
* Name:    simu.c
*
* Purpose: cell lost rate simulation for low and high speed connections
*
* Note:    In UNIX, use the following commands to compile and run:
*    > gnucc -g simu.c /usr/lib/libm.a -o simu
*    > simu
*/

#include "simu.h"

#define V       20/* capacity: # of cells trasmitted in one time-slot */
#define SpeedRate 5/* Vh/Vl */

#define NsegmL     1000  /* used to determining the length of cell array */

#define NlinkL     1       /* # of low speed connections */
#define NlinkH     115   /* # of high speed connections */

#define AvgBurstL  230   /* average burst length (cells) for low speed connections */
#define AvgIdleL   230     /* average idle length (cells) for low speed connections */

#define NsegmH   (NsegmL * SpeedRate * (AvgBurstL + AvgIdleL) / (AvgBurstH +
AvgIdleH))
#define AvgBurstH  230   /* average burst length (cells) for high speed connections */
#define AvgIdleH   2070   /* average idle length (cells) for high speed connections */

/*
* Name:     cellArrayInt - Initialize the cell array
*
* Params:
*    maxLength: Required length for this cell array.
*
* Return:   Pointer to the initialized the cell array
*/

uint2 * cellArrayInit(uint4 maxLength)
{
  uint2         *cellarray;
  uint4         i;

  cellarray = malloc(maxLength * sizeof(uint2));
```

```
    /* init the cell array */
    for (i = 0; i < maxLength; i++)
    {
      cellarray[i] = 0;
    }

    return cellarray;
}

/*
* Name:     linkAdd - Add one link to the cell array
*
* Params:
*     cellarray: Cell array to add one link to.
*
*     nSegm:Number of segments to be created. One segment = one
*burst period + one idel period.
*
*     avgBurst:average burst length.
*
*     avgIdle:average idle length.
*
* Return:   length of added cell stream
*
*/
uint4 linkAdd(uint2 cellarray[],
     uint2 nSegm,
     uint2 avgBurst,
     uint2 avgIdle)
{
    uint2          *nptr, *mptr;
    float          num, dom;
    uint2          i;
    uint4          k, index;

    /* allocate memory in words */
    nptr = malloc(nSegm*sizeof(uint2));
    mptr = malloc(nSegm*sizeof(uint2));

    /*
     * create the burst length array and the idle length array
     */
    for (i = 0; i < nSegm; i++)
    {
```

```
num = log(1.0 - (float) rand() / 32768.0);
dom = log(1.0 - 1.0 / (float) avgBurst);
nptr[i] = (int) (num / dom) + 1;

num = log(1.0 - (float) rand() / 32768.0);
dom = log(1.0 - 1.0 / (float) avgIdle);
mptr[i] = (int) (num / dom) + 1;
   }

   /* increment the cell array in the period of burst */
   index = 0;
   for (i = 0; i < nSegm; i++)
   {
for (k = index; k < (index + nptr[i]); k ++)
{
   cellarray[k]++;
}
index += (nptr[i] + mptr[i]);
   }

   /* free the allocated memory */
   free(nptr);
   free(mptr);

   return index;
}

/*
* Name:    cellArrayCr - Create a cell array
*
* Params:
*    cellarray: Cell array to be created.
*
*    linkNum:Number of connections to be added.
*
*    nSegm:Number of segments to be created. One segment = one
*burst period + one idel period.
*
*    avgBurst:average burst length.
*
*    avgIdle:average idle length.
*
* Return:  Length of the created cell array
*
```

```
* Note:    The memory for the cell array must be allocated and initialized
*      by the caller before calling the fucntion.
*/
uint4 cellArrayCr(uint2 cellarray[],
    uint2 linkNum,
    uint2 segmNum,
    uint2 avgBurst,
    uint2 avgIdle)
{
  uint4        minlen;
  uint4        arraylen;
  uint4        i, k;

  minlen = 0xffffffff;

  /* add required connections to the cell array */
  for (i = 0; i < linkNum; i++)
  {
arraylen = linkAdd(cellarray, segmNum, avgBurst, avgIdle);
if (minlen > arraylen)
{
  minlen = arraylen;
}
  }

  /*
   * chop off the first (avgBurst+avgIdle) to have a random start
   */
  minlen -= (avgBurst + avgIdle);
  for (k = 0; k < minlen; k++)
  {
cellarray[k] = cellarray[k + (avgBurst + avgIdle)];
  }

  return minlen;
}


/*
* Name:    clrCalcNoBuffer - CLR calculation without buffer
*
* Params:
*   cellarrayL: Created low speed cell array.
*
```

```
*    cellarrayH: Created high speed cell array.
*
*    lengthL:Length of low speed cell array.
*
*    lengthH:Length of low speed cell array.
*
*    capacity:Maximum allowed cells to send in a high speed time slot
*
*    rate:rate of high speed to low speed.
*
* Return:  Calculated Cell loss rate
*
*/

float clrCalcNoBuffer(uint2 * cellarrayL,
    uint2 * cellarrayH,
    uint4 lengthL,
    uint4 lengthH,
    uint2 capacity,
    uint2 rate)

{
  int4        delta;/* number of lost cells in a time slot */
  uint4       over;/* total number of lost cells */
  uint4       sumL;/* total number of low speed cells */
  uint4       sumH;/* total number of high speed cells */
  uint4       len;
  uint4       k, j;

  len = min(lengthH, lengthL * rate);

  over = 0;
  sumL = 0;
  sumH = 0;

  /*
   * count the lost cells in high speed connections
   */
  for (k = 0; k < len; k++)
  {
delta = cellarrayH[k] - capacity;
sumH += cellarrayH[k];
if (delta > 0)
{
```

```
    over += delta;
    cellarrayH[k] = capacity;
}
    }

    printf(" overH = %d\n", over);
    printf(" sumH  = %d\n", sumH);

    /*
     * count the lost cells in low speed connections
     */
    for (k = 0; k < (len / rate); k++)
    {
delta = cellarrayL[k] - capacity * rate;
for (j = 0; j < rate; j++)
{
    delta += cellarrayH[k * rate + j];
}

if (delta > 0)
{
    over += delta;
}
sumL += cellarrayL[k];
    }

    printf(" overAll = %d\n", over);
    printf(" sumL  = %d\n", sumL);


    return (float) over / (sumH + sumL);

}


float clrCalcWithBuffer(uint2 * cellarray,
        uint4 arrayLen,
        uint2 capacity,
uint2 bufferLen)

{
    int4          delta;/* number of lost cells in a time slot */
    uint4         over;/* total number of lost cells */
    uint4         sum;/* total number of cells to be transmitted */
```

```
    uint4       k;/* loop index */
    int4        buffer;

    /* count the lost cells */
    over = 0;
    sum = 0;
    buffer = 0;
    for (k = 0; k < arrayLen; k++)
    {
buffer += (cellarray[k] - capacity);
delta = buffer - bufferLen * capacity;
if (delta > 0)
{
    buffer = bufferLen * capacity;
    over += delta;
}
sum += cellarray[k];

if (buffer < 0)
{
    buffer = 0;
}
    }

    printf(" over = %d\n", over);
    printf(" sum  = %d\n", sum);

    return (float) over / sum;

}

void main()
{
    uint2       *cellarrayL, *cellarrayH;
    uint4       maxLengthL, maxLengthH;
    uint4       lengthH;/* valid length of the high-spped cell array */
    uint4       lengthL;/* valid length of the low-spped  cell array */
    uint4       minlen;/* min length of the created cell array */
    float       clr;/* cell lost rate */
    uint4       k;/* loop index */

   /*
    * estimate the maximum lengths for cell arras to be created
    */
```

```
    maxLengthL = (NsegmL * (AvgBurstL + AvgIdleL) * 2);
    maxLengthH = (NsegmH * (AvgBurstH + AvgIdleH) * 2);

    printf(" maxLengthL = %d maxLengthH = %d\n", maxLengthL, maxLengthH);

    /*
     * allocated memory and initalize the cell arrays
     */
    cellarrayL = cellArrayInit(maxLengthL);
    cellarrayH = cellArrayInit(maxLengthH);

    /* create the low speed cell array */
    lengthL = cellArrayCr(cellarrayL, NlinkL, NsegmL, AvgBurstL, AvgIdleL);

    /* create the high speed cell array */
    lengthH = cellArrayCr(cellarrayH, NlinkH, NsegmH, AvgBurstH, AvgIdleH);

    printf(" lengthL = %d lengthH = %d\n", lengthL, lengthH);

    clr = clrCalcNoBuffer(cellarrayL, cellarrayH,
   lengthL, lengthH, V, SpeedRate);

    printf(" clr  = %f\n\n", clr);

    /*
     * free memory for the cell arrays
     */
    free(cellarrayL);
    free(cellarrayH);
}
```

Appendix B Implementation of CAC under Partial Separation

```
% Main program of optimization for CAC under partial separation
%
%  This program will call the following matlab files:
%    OneDimSearch.m
%    object.m
%    clrate.m
%    prob.m
%    frob.m
%    comb.m

   % clear all the variables in matlab environment
   clear;

   % spcify the searching step length which is the minimum average rate for a link
   stepLen = 0.75e6;

   % specify the starting point and the initial searching direction
   X0 = [159e6, 143e6];
   Sigma1_old = [1, 0];
   Sigma2_old = [0, 1];

   % calculate the objective function at the starting point
   y0 = object(X0)

   %
   % start the seaching loop
   %
   for loop = 1:3

   loop

% One dimension search along the direction: Sigma1_old
  [X1, y1] = OneDimSearch(X0, y0, Sigma1_old, stepLen)

% One dimension search along the direction: Sigma2_old
  [X2, y2] = OneDimSearch(X1, y1, Sigma2_old, stepLen)

%
% assign the search direction for the next loop
%
  if (y2-y1) > (y1-y0)
```

```
      Sigma1_new = Sigma1_old;
      Sigma2_new = Sigma2_old;
 else
        Sigma1_new = Sigma2_old;
        Sigma2_new = Sigma1_old;
   end

%
   % evaluate the direction (X2 - X1) and determine the Sigma2_new
%
   yt = object(X0 + 2 * (X2 - X0));
  if (yt < y2)
      delta = max((y1 - y0), (y2 - y1));
      if (y0 -2 * y2 + yt)(y0 - y2 - delta)^2 >= delta * (y0 - yt)^2 / 2
   Sigma2_t = X2 - X0;
        Sigma2_new = Sigma2_t / norm(Sigma2_t)
      end
   end

  %
% assign the initial point and direction for next loop
%
    Sigma1_old = Sigma1_new;
        Sigma2_old = Sigma2_new;
  X0 = X2;
  y0 = y2;
   end


% ==========================================================
function [X_new, y_new] = OneDimSearch(X_old, y_old, sigma, stepLen)

% OneDimSearch  One dimension search for minimum of the objective function
%    OneDimSearch(X_old, y_old, sigma, stepLen) is to find the minimum point
%    along the direction of "sigma", starting at "X_old". "stepLen" is the
%    searching step length

   % determine the searching direction
   if (object(X_old + sigma * stepLen) > y_old)
   stepLen = - stepLen;
   end

   % start searching
   step = 0;
```

```
   for step = 1:100
   X1 = X_old + sigma * stepLen;
   y1 = object(X1);

% if objective function goes up, stop searching
   if y1 > y_old
      break;
     end

   X_old = X1;
   y_old = y1;
   end

   % assign the return values
   X_new = X_old;
   y_new = y_old;


% ================================================
function [obj] = object(C_vect)

% object  objective function calculation
%    object(C_vect) is to calculate the objective function using the given
%    capcity vector. The objective function is formed from the cell loss rates
%    penlty functions from three classes of links. The capcity vector contains
%    the given capcities for the first two classes to links

   % total capacity for all classes of links
   C_total = 450e6;

   % given capacity for the 1st class of links
   C1 = C_vect(1);

   % given capacity for the 2nd class of links
   C2 = C_vect(2);

   % given capacity for the 3rd class of links
   C3 = C_total - C1 - C2;

   % the maximum allowed cell loss rates for each classes
   Clrmax = [1.0e-6, 1.0e-4, 1.0e-3];

   % average rate for low speed and high speed links
   Val = 0.75e6;
```

```
Vah = 0.75e6;

% peak rate for high speed links
Vph = 1.875e6;

% number of links for the 1st class
Nl1 = 200;
Nh1 = 0;

% number of links for the 2nd class
Nl2 = 0;
Nh2 = 140;

% number of links for the 3rd class
Nl3 = 75;
Nh3 = 75;

% specify an arbitray penalty factor
factor = 100000;

% cell losss rate for the 1st class
object1 = clrate(C1, Nl1, Val, Nh1, Vah, Vph);

% cell losss rate for the 2nd class
object2 = clrate(C2, Nl2, Val, Nh2, Vah, Vph);

% cell losss rate for the 3rd class
object3 = clrate(C3, Nl3, Val, Nh3, Vah, Vph);

%
% add penalty of the cell loss rate is greater than the allowed maximum
%
  if object1 > Clrmax(1)
object1 = object1 + factor * (object1 - Clrmax(1));
  end
  if object2 > Clrmax(2)
object2 = object2 + factor * (object2 - Clrmax(2));
  end
  if object3 > Clrmax(3)
object3 = object3 + factor * (object3 - Clrmax(3));
  end

% return the objective value
obj = object1 + object2 + object3;
```

```
% ==========================================================
function [clr] = clrate(V, Nl, Val, Nh, Vah, Vph)

% clrate  cell loss rate calculation
%clrate(V, Nl, Val, Nh, Vah, Vph) is to calculate the cell loss rate for
%a class of links, which has "Nl" low speed links with average rate "Val"
%"Nh" links of high speed links with with average rate "Vah" and peak
%"Vph". The total capcity is V

   V1 = fix(V/100);

   alfl = Val/V1;
   alfh = Vah/Vph;
   beta = (Vph/V1);

   ru = (Nl*Val+Nh*Vah)/V;
   Mt = 100;
   P = prob(alfl, Nl);
   F = frob(alfh, Nh, beta);
   G = conv(P, F);
   clr = 0;
   for n=(Mt+1):(size(G)-1)
   clr = clr + (n/Mt-1) * G(n+1);
   end
   clr = clr /ru;


% ==========================================================
function [prob]=prob(alf, N)

% prob  probability distribution calculation for low speed links
%    prob(alf, N) is to calculate the probability distribution for low
%    speed links. "alf" is the source activity. "N" is the number of links.

prob = zeros(N+1,1);

for n=0:N
k = 1+n;
prob(k) = prob(k) ...
+comb(N,n)*alf^n*(1-alf)^(N-n);
end
```

```
% ================================================================
function [prob] = frob(alf, N, beta)

% frob  probability distribution calculation for high speed links
%    frob(alf, N, beta) is to calculate the probability distribution for high
%    speed links. "alf" is the source activity. "N" is the number of links.
%    "beta" is the rate of speed of high speed connections over the threshood
%    speed.

prob = zeros(1+ceil(N*beta),1);

for n=0:N
k = 1 + ceil(n*beta);
prob(k) = prob(k) ...
+comb(N,n)*alf^n*(1-alf)^(N-n);
end


% ================================================================
function [x] = comb(N,n)

%DEMODSignal demodulation for communications simulations.
%X = DEMOD(Y,Fc,Fs,METHOD,OPT) demodulates the carrier signal Y with a
%carrier frequency Fc and sampling frequency Fs, using the demodulation
%scheme in METHOD.  OPT is an extra, sometimes optional, parameter whose
%purpose depends on the demodulation scheme you choose.
%
% METHOD    DEMODULATION SCHEME
%Author(s): T. Krauss, 1993
%Copyright (c) 1984-94 by The MathWorks, Inc.
%$Revision: 1.9 $  $Date: 1994/01/25 17:58:59 $
x = 1;
if n == 0 | N == n
return;
end

for i=(N-n+1):N
   x=x*i/(i-N+n);
end
```