24

Single Node and End-to-End Buffer Control in Real Time

Erol Gelenbe[a] Sridhar Seshadri[b] and Vijay Srinivasan[c]

[a]Department of Electrical Engineering,
Duke University, Box 90291, Durham, NC 27708-0291, USA

[b]Department of Statistics and Operations Research,
Leonard N. Stern School of Business, New York University, NY 10012-1118, USA

[c]Department of Computer Science,
Duke University, Box 90129, Durham, NC 27708-0129, USA

Abstract

High speed integrated services networks will multiplex distinct traffic streams that have different sensitivities to packet or cell loss. We consider traffic streams or "classes" that have a cost function representation related to cell loss. These traffic classes share a common finite capacity buffer. Using this model we characterize optimal scheduling and rejection rules for minimizing the cost of lost cells, both over the finite and infinite time horizon. Optimization is considered both at the single switch level and for a sequence of switches with cross traffic. For the single node case we describe optimal policies for cell scheduling and rejection when loss related cost functions are used. These cost functions can either be linear or convex. In the case of a two-node system, we describe numerical experiments using a dynamic programming formulation. Although this leads to large computational cost, it does provide insight into simple rules which could be used to approximate optimal policies.

Keyword Codes: C.2.1; C.2.3; C.4
Keywords: Network Architecture and Design; Network Operations; Performance of Systems

1. INTRODUCTION

The next generation of high speed packet-switching networks based on the Asynchronous Transfer Mode (ATM) are expected to provide guaranteed Quality of Service (QoS) to a wide range of types of connection. Perhaps the most important measure of QoS is the level of cell or packet loss seen by a connection. Cell loss will occur largely due to lack of physical buffer space in ATM nodes during temporary periods of congestion. The purpose of this paper is to characterize *buffer control policies* for connections sharing finite capacity buffers in ATM network nodes, with respect to the cell loss rates which each user may encounter. Thus, the goal of these policies is to optimize loss-oriented performance criteria. When several priority levels or classes of traffic are present, these

D. D. Kouvatsos (ed.), *Performance Modelling and Evaluation of ATM Networks*

criteria can be incorporated into different types of cost functions depending on the desired effect.

We consider optimal policies for packet or cell scheduling, based on minimizing a cost function which is directly related to cell or packet loss. For the single node case, we will consider both simple linear "weighted sum" type cost functions, and more complex convex functions. We will exhibit the optimal policy in the linear case, and show non-optimality results for intuitive policies in more general cases. We study the multi-node case via a simple dynamic programming formulation of a two-node network. Through insights gained from the dynamic program, we construct a simple heuristic that significantly outperforms static priority rules.

We note here that when we refer to the rejection of a cell from a particular class, we are implicitly assuming that the sequential ordering of cells from that class is preserved. When there is more than one cell from that class to choose from for discarding, this can be achieved by selecting the cell that entered the buffer last. Similary, when there is more than one cell from that class to choose from for transmission, we select the cell that entered the buffer first. This preserves the in-sequence delivery of cells, a basic property of virtual circuits in ATM networks.

We briefly describe below significant related prior work. In this paper, we summarize several results from [1] on optimal control policies for a single ATM node with finite buffer space. In [2] a *space priority* single-server queue is analyzed with two classes of traffic as an $M/G/1$ system. The space priority discipline permits cells from the more important class to take the place of a cell from the less important class when the finite waiting room is full. Cidon, Guerin and Khamisy in [3] are concerned with buffering policies that can guarantee a loss probability to cells from high priority classes regardless of the arrival characteristics of cells from low priority classes. Lin and Silvester have considered (see [4]) two classes of service in ATM networks under several priority queueing strategies. The performance criterion they consider is minimum overall loss probability. Bounds are provided on the performance of the high and low priority classes. Petr and Frost in [5] use a dynamic programming approach to minimize the average cell discarding cost in a system with two classes of traffic and waiting room for one cell. In [6], three classes of buffer control policies are identified and optimal control policies with respect to some performance criteria are characterized. The problem of characterizing feasible regions to guarantee delay and loss-related QoS measures and the issue of designing efficient control policies from these regions has been studied in [7], [8] and [9]. Clare and Rubin in [8] consider a slotted-time queueing system in which the service time is deterministic and cells can begin and end transmission only at slot boundaries. For each source, the arrival process is modeled by assuming that the number of cell arrivals in the slots form an i.i.d. sequence of random variables. However, the arrivals of cells from different sources in the same slot could be correlated. Under these assumptions they show that a work conserving policy – i.e. one that never idles when there are cells awaiting transmission – and a *non-expelling* rejection policy – i.e. one that rejects a cell only when the buffer is full – minimizes the total number of lost cells at every time instant. They prove these results using sample path analysis. We extend these results to more general arrival patterns by assuming a certain conservation law holds. They also define a *strict discipline* as one that fixes the priority levels once and for all for all the classes and uses these to determine

scheduling and rejection actions: the highest priority cell is scheduled first, and the lowest priority cell is dropped first. We call such a policy a *static priority* policy. Clare and Rubin show that the vector of long-run average losses is the largest in the majorization order and use a limit theorem for regenerative processes to prove this result. We show that the same result holds for general arrival processes on every sample path using the recently developed theory by [10].

In the sequel we will interchangeably use the term "packet" and "cell". It is to be understood however that we basically consider systems with fixed cell size (set to 1) and that our work is oriented towards ATM. Our results are nevertheless applicable to any networking environment where cells of constant size are being transmitted, and in which packet (or cell) loss is a major issue.

2. THEORETICAL RESULTS FOR A SINGLE NODE

The theoretical results we develop consider a multiplexer with a single server and a finite buffer with waiting room for K cells. We do *not* consider the effect of cross traffic. The simulation results of the next section do include cross traffic, however.

The multiplexer is serviced by a channel of capacity C. Cells arrive to the multiplexer from M different traffic classes, and we denote by $a_1 < a_2 < \ldots$ the successive arrival times. We will denote the service or transmission times of cells as $\{S_1, S_2, \ldots\}$. If a cell arrives to a full buffer, i.e. with K cells awaiting transmission, either the newly arriving cell or one of the K cells in the buffer is dropped (lost to the system). The loss process is described by $\{L_i(t), i = 1, 2, \ldots, M\}$ where $L_i(t)$ is the number of cells discarded from class i until time t, starting from time $t = 0$.

We consider control policies which select cells for transmission among those which are waiting, when a transmission ends, and for rejection when the buffer is full and a new arrival occurs. We assume that the scheduling discipline is non-idling (work conserving), non-anticipatory and non-preemptive. A rule is said to be non-anticipatory if it does not use information about the future. Specifically, it is assumed not to use any information about the service times of the cells waiting for transmission or about when cells from different classes will arrive to the multiplexer (but it may use information about the number and type of cells that have already been serviced/discarded or are awaiting service). The rejection rule is also assumed to be non-anticipatory. The class of all scheduling and rejection rules satisfying the above assumptions will be denoted as $\mathcal{A}$. The elements of $\mathcal{A}$ will be denoted as $u = (s, r)$ where s is a scheduling rule and r is a rejection rule.

Define $N_i^u(t)$ to be the number of cells from class i waiting for transmission in the buffer at time t when control $u \in \mathcal{A}$ is used. Let $L_i^u(t)$ be the number of cells from class i that have been discarded until time t under control $u \in \mathcal{A}$. We shall assume that sample paths of $N_i(t)$ and $L_i(t)$ are right continuous and have left hand limits. We make the key assumption that the choice of any of the policies we consider will not affect $N(t)$, the total number of cells in the buffer at any time. This assumption is realistic in many cases of interest. For instance, if all cells have the same length – and therefore service or transmission times – then this assumption will clearly be satisfied. This is obviously true in ATM networks where all cells are of constant (53 bytes) length. In fact, it can be shown that for a given sequence of arrival instants, the assumption that the number of

cells in queue does not depend on the policy is equivalent to assuming that the departure instants do not depend on the policy.

Now let $L^u(t)$ be the total number of cells lost until time t when control u is used:

$$L^u(t) = \sum_{i=1}^{M} L_i^u(t).$$

Also, let $N^u(t)$ be the total number of cells waiting for transmission in the buffer at time t when control u is used:

$$N^u(t) = \sum_{i=1}^{M} N_i^u(t)$$

The following is an important property of the policies we consider.

Lemma 2.1 *If for any* $u,\ u' \in \mathcal{A}$, *we have* $N^u(t) = N^{u'}(t)$, *then*

$$\sum_{i=1}^{M} L_i^u(t) = \sum_{i=1}^{M} L_i^{u'}(t).$$

Proof : We omit the proof of this lemma. Please see [1] for details.

We first assume that the objective to be minimized is a linear loss related cost function of the form:

$$\min_u \sum_{i=1}^{M} c_i L_i^u(t), u \in \mathcal{A}. \tag{1}$$

Without loss of generality, assume that $c_1 > c_2 > \ldots > c_M > 0$. Define π to be the scheduling and rejection rule that gives the highest priority to class 1, the next highest priority to class 2 and so on. This means that the next cell to be selected for transmission from the pool of waiting cell will be from the class with the smallest index. Similarly, when a cell has to be discarded because the buffer is full, it will be chosen from the class with the largest index. We will prove that π achieves the objective specified above for every instance t. Note that the "squeeze-out" policy described in [6] is a specific case of π when the number of classes $M = 2$.

Theorem 2.1 $\sum_{i=1}^{M} c_i L_i^{\pi}(t) \leq \sum_{i=1}^{M} c_i L_i^u(t), u \in \mathcal{A}$

Proof : We use an inductive proof. Assume that given any $b_1 > b_2 > \ldots > b_n > 0$, $\sum_{i=1}^{n} b_i L_i^{\pi}(t) \leq \sum_{i=1}^{n} b_i L_i^u(t)$ is true. Then given $b_1 > b_2 > \ldots > b_n > b_{n+1} > 0$, consider

$$\sum_{i=1}^{n+1} b_i (L_i^{\pi}(t) - L_i^u(t)) = \sum_{i=1}^{n} (b_i - b_{n+1})(L_i^{\pi}(t) - L_i^u(t)) + b_{n+1} \sum_{i=1}^{n+1} (L_i^{\pi}(t) - L_i^u(t)) \tag{2}$$

The first term on the right hand side of (2) is less than or equal to 0 by the inductive hypothesis. The second term is also non-positive by Lemma 2.2 given below. ∎

Lemma 2.2 $\sum_{i=1}^{n}(L_i^{\pi}(t) - L_i^{u}(t)) \leq 0, n = 1, 2, \ldots, M$

Proof : Recall that $\sum_{i=1}^{M} L_i^{\pi}(t) = \sum_{i=1}^{M} L_i^{u}(t)$ by Lemma 2.1. Assume that the above statement is true at time $t = 0$. Let T_1 be the first time that the buffer is full and we have a new arrival, i.e. T_1 is the first time at which the rejection rule is invoked. If T_1 is not finite, then the lemma is true. Otherwise, in $[0, T_1)$ there have been no cells discarded under either control π or control u. Moreover, since no cells have been lost in this interval, and because policy π gives priority for transmission to classes 1 to n,

$$\sum_{i=1}^{n} N_i^{\pi}(T_1^-) \leq \sum_{i=1}^{n} N_i^{u}(T_1^-) \tag{3}$$

The lemma is true in $[0, T_1)$ since we have no losses in this interval. The lemma is true at T_1 and thus in $[0, T_1]$ because of (3) and the priority given for not rejecting cells from classes 1 to n. Let T_2 be the next time instant that the rejection rule is invoked. If T_2 is not finite, the lemma is true. We now need to consider two cases.

Case 1. Both π and u discarded a cell from the same class at T_1. Then,

$$\sum_{i=1}^{n} N_i^{\pi}(T_2^-) \leq \sum_{i=1}^{n} N_i^{u}(T_2^-) \tag{4}$$

If the cell discarded by control π at time T_2 is not from classes 1 to n, then the lemma holds during $[0, T_2]$. Otherwise, if π discards a cell from classes 1 to n then there must only be cells from classes 1 to n to choose from due to the nature of the rejection rule. Thus by (4) u also discards a cell from classes 1 to n.

Case 2. π and u discarded cells from different classes at T_1. Then,

$$\sum_{i=1}^{n} N_i^{\pi}(T_2^-) \leq \sum_{i=1}^{n} N_i^{u}(T_2^-) + 1 \tag{5}$$

On the right hand side of (5) we have an additional cell since control π may have rejected a cell not belonging to classes 1 to n, while control u may have rejected a cell from classes 1 to n. At time T_2 if control π does not reject a cell from classes 1 to n, the lemma is true during $[0, T_2]$. On the other hand, if there is equality in (5) we must have $\sum_{i=1}^{n} L_i^{\pi}(T_2^-) = \sum_{i=1}^{n} L_i^{u}(T_2^-) - 1$. Therefore, even if control π discards a cell from classes 1 to n at T_2, the lemma is true in $[0, T_2]$.

The construction can be continued in the same manner to $T_3, T_4, \ldots$. The general induction hypothesis is that if at time T_m^-:

$$\sum_{i=1}^{n} N_i^{\pi}(T_m^-) = \sum_{i=1}^{n} N_i^{u}(T_m^-) + x_m$$

and

$$\sum_{i=1}^{n} L_i^{\pi}(T_m^-) = \sum_{i=1}^{n} L_i^{u}(T_m^-) + y_m, y_m \geq \max(x_m, 0)$$

Then:

$$\sum_{i=1}^{n} N_i^{\pi}(T_{m+1}^-) = \sum_{i=1}^{n} N_i^{u}(T_{m+1}^-) + x_{m+1}$$

and

$$\sum_{i=1}^{n} L_i^{\pi}(T_{m+1}^-) = \sum_{i=1}^{n} L_i^{u}(T_{m+1}^-) + y_{m+1}, y_{m+1} \geq \max(x_{m+1}, 0)$$

We consider four cases below.

Case 1. At time T_m^- policy π does not reject a cell from classes (1,2,...,n) and policy u also does not reject a cell from these classes. We must have $x_{m+1} \leq x_m$ because of the scheduling rule and $y_{m+1} = y_m$.

Case 2. At time T_m^- policy π does not reject a cell from classes (1,2,...,n) but policy u rejects a cell from these classes. We must have $x_{m+1} \leq x_m + 1$ because of the scheduling rule and $y_{m+1} = y_m + 1$.

Case 3. At time T_m^- policy π rejects a cell from classes (1,2,...,n) and policy u rejects a cell from these classes. We must have $x_{m+1} \leq x_m$ because of the scheduling rule and $y_{m+1} = y_m$.

Case 4. At time T_m^- policy π rejects a cell from classes (1,2,...,n) but policy u does not reject a cell from these classes. We must have $x_{m+1} \leq x_m - 1$ because of the scheduling rule and $y_{m+1} = y_m - 1$. In addition because policy π rejected a cell from the first n classes, it must be true that $x_m > 0$. Therefore, $y_{m+1} = y_m - 1 \geq x_m - 1 \geq 0$. ∎

Theorem 2.1 characterizes the optimal policy for the finite time horizon case. For the infinite time horizon case we can employ a continuous discount factor ρ and by using Theorem 2.1 obtain:

Corollary 2.1

$$\int_0^{\infty} e^{-\rho t}(\sum_{i=1}^{M} c_i L_i^{\pi}(t))dt \leq \int_0^{\infty} e^{-\rho t}(\sum_{i=1}^{M} c_i L_i^{u}(t))dt, u \in \mathcal{A}$$

Several remarks are worth making at this point. Note that in order to achieve the minimum cost, it is necessary to control both the order in which cells are scheduled for transmission as well as the manner in which packets are selected for discarding. The optimal solution is said to be *path-wise least cost* if the objective function is minimum at each point of time on every sample path under the optimal control. For examples of the definition and for uses of path-wise least cost solutions, the reader may see [11–14]. Thus the above proof shows that the policy π is a path-wise least cost solution.

We can define the performance vector to be the expected number of lost cells of each class $(EL_i(t), i = 1, 2, .., M)$. The arguments developed above show that this performance vector satisfies a *strong conservation law* as defined in [10]. We will use this fact in the next section.

3. CONVEX COST FUNCTIONS

Although the linear cost functions discussed in the previous sections are intuitively appealing, there are instances where other cost functions would be of interest – for instance when one would like to differentiate more strongly between priority classes, so as to let some priority class become "much more important" than other classes with respect to cell loss. As an example, for a 2-class system one could have an objective of the form:

$$\min_u \ (c_1 L_1(t) + c_2[L_2(t) - \gamma]^+), u \in \mathcal{A}.$$

where we are saying that losses for the second class of cells are to be considered only if they exceed a quantity $\gamma > 0$.

Thus we will now consider strictly increasing and convex cost functions $f_i(L_i(t))$, $i = 1, \ldots, M$ and the objective is now to find policies u which minimize:

$$\sum_{i=1}^{M} f_i(L_i^u(t)), u \in \mathcal{A}. \tag{6}$$

We first give an example that shows it is impossible in general to obtain a path-wise least cost solution, as for the strictly linear case, when the cost functions are strictly convex. However we do not exclude that such path-wise optimal solutions exist for special cases.

A counterexample Consider the case where we have two classes sharing a buffer of size 1, i.e. $M = 2$ and $K = 1$. Assume that both classes have the same strictly increasing and convex cost function given by $f_1(0) = f_2(0) = 0$, $f_1(1) = 2$, $f_1(2) = 12$, $f_2(1) = 4$, and $f_2(2) = 12$.

Let us assume that we have a path-wise least cost control π for this system. We will show the existence of a control $u \in \mathcal{A}$ that achieves $\sum_{i=1}^{M} f_i((L_i^u(t)) < \sum_{i=1}^{M} f_i((L_i^\pi(t))$ for some t and a specific realization of events. Label the system controlled using π as I and the other system as II. At the first time, t_1, that a cell has to be dropped let there be a cell of type 2 in the buffer in both systems (this can be achieved in II by following the same scheduling policy as in I). At t_1 let the newly arriving cell be of type 1. Let π decide to drop the type 1 cell and u decide to drop the type 2 cell. From t_1 onwards, let there be no more arrivals of type 2 and only arrivals of type 1. Therefore, at the next instant t_2 that a cell has to be dropped, both π and u will have to drop a type 1 cell. Then $f_1(L_1^\pi(t_2)) + f_2(L_2^\pi(t_2)) = 12 > 6 = f_1(L_1^u(t_2)) + f_2(L_2^u(t_2))$. If at time t_1, π drops a type 2 cell, then u drops a type 1 packet. If we assume that only type 2 cells arrive after time t_1, we obtain a similar contradiction at time t_2.

We will now turn to the characterization of optimal control policies when the performance measure is expressed as an expected or average value.

3.1. Optimality for Average Convex Cost

In the sequel we will assume that the model is defined in a probability space, by an appropriate probabilistic representation of the arrival and service process.

Our development would also be applicable if we were dealing with infinite sample paths. Let

$$\lim_{t\to\infty} \inf \frac{L_i^u(t)}{t} = \beta_i^u, i = 1, 2, \ldots, M. \tag{7}$$

Then the infinite time horizon minimization problem is:

$$\min_u \sum_{i=1}^{M} f_i(\beta_i^u), u \in \mathcal{A} \tag{8}$$

Assuming an appropriate probability space, we will consider the finite time horizon problem of minimizing the expected cost:

$$\min_u \sum_{i=1}^{M} E[f_i(L_i^u(t))], u \in \mathcal{A} \tag{9}$$

As we will see below, the solution to (8) can be characterized via absolute priority rules, similar to what was done previously – see Gelenbe and Mitrani (Chapter 6 of [15]). We will show that a lower bound for the optimal solution to (9) can be constructed using the recently developed theory by Shanthikumar and Yao (see [10]).

Define an *absolute priority rule* α to be a permutation of $\{1, \ldots, M\}$ which gives the highest scheduling and rejection priorities to class $\alpha(1)$, the second highest to $\alpha(2)$ and so on. From Lemma 2.1 it follows that for all $u \in \mathcal{A}$:

$$\sum_{i=1}^{M} L_{\alpha(i)}^{\alpha}(t) = \sum_{i=1}^{M} L_{\alpha(i)}^{u}(t), \tag{10}$$

and

$$\sum_{i=1}^{m} L_{\alpha(i)}^{\alpha}(t) \leq \sum_{i=1}^{m} L_{\alpha(i)}^{u}(t), \forall m \leq M \tag{11}$$

From (10), (11), and the definition given in [10] (and earlier work in [15] on achievable performance vectors), the performance vector of expected number of lost cells, $\{EL_1(t), EL_2(t), \ldots, EL_M(t)\}$, satisfies a strong conservation law. From Theorem 1 of [10] it then follows that the achievable performance vectors are contained in the (base of a) polymatroid, P_1, whose vertices correspond to the absolute priority rules. Using this fact, we have:

Theorem 3.1 *If the objective is*

$$\min_u \sum_{i=1}^{M} E[f_i(L_i^u(t))], u \in \mathcal{A},$$

a lower bound on the optimal value of the objective function is given by

$$\min_{x \in P_1} \sum_{i=1}^{M} f_i(x_i). \tag{12}$$

Proof : By the reasoning given in the previous paragraph, every achievable value of the performance vector belongs to the polymatroid P_1. Therefore minimizing $\sum_{i=1}^{M} f_i(x_i)$ over P_1 solves

$$\min_u \sum_{i=1}^{M} f_i(E[L_i^u(t)]), u \in \mathcal{A}.$$

Since the $f_i(.)$ are convex, an application of Jensen's inequality (see for example p. 47 in [16]) gives the required result. ∎

Remarks:

1. This theorem suggests that absolute priority rules need not be optimal and we may have to search for randomized rules to achieve satisfactory performance.

2. We can use Theorem 3.1 to compare the performance of several heuristics that are simpler to implement.

As shown for the performance vector $\{EL_1(t), EL_2(t), \ldots, EL_M(t)\}$, we can show that the performance vector $\{\beta_1, \beta_2, \ldots, \beta_M\}$ satisfies a strong conservation law. Therefore every achievable performance vector is contained by the convex polytope, P_∞ (which is the base of a polymatroid) generated by the performance vectors $\{\beta_1^\alpha, \beta_2^\alpha, \ldots, \beta_M^\alpha\}$ corresponding to the absolute priority rules (α). We shall restrict our attention to the class of controls $\mathcal{A}_L$ (or systems) in which the following equality is attained almost surely in the limit:

$$\lim_{t\to\infty} L_i^u(t)/t \stackrel{a.s.}{=} \beta_i^u, i = 1, 2, \ldots, M. \tag{13}$$

Theorem 3.2 *If the objective is*

$$\min_u \sum_{i=1}^{M} f_i(\beta_i^u), u \in \mathcal{A}_L,$$

the optimization problem is equivalent to solving the convex program:

$$\min_{x\in P_\infty} \sum_{i=1}^{M} f_i(x_i) \tag{14}$$

Proof : Similar to the first part of Theorem 3.1. ∎

In practice, to apply the results of Theorems 3.1 and 3.2, we need to know the break-up of losses suffered by different classes under each absolute priority rule in order to determine the vertices of the polymatroid. One method for obtaining this break-up is to use simulation. However, since the loss probability is often of the order 10^{-9}, simulation runs will have be very long. If it were possible to characterize the arrival patterns before hand – eg. for some packet audio applications – the calculations for the break-up of loss can be performed off-line and the actual control determined in real time. Another method of estimating losses is to use approximations, but using the available approximations for queues with bounded buffer space (see Buzacott and Shanthikumar [17] and Gelenbe and

Mitrani (chapter 4 of [15])), losses cannot be computed accurately. Simulations suggest that even the relative losses of different classes for a given absolute priority rule can be extremely sensitive to the distribution of the arrival streams of cells. This area of approximations holds potential for future research.

Remark: For a system with Poisson arrivals and i.i.d. service times, once the optimal value of $\mathbf{x}$ has been obtained from solving the convex programs (12) or (14), the priority rule can be determined by using the methods of Federgruen and Groenvelt [18]. For the general case we can resort to randomized application of absolute priority rules or apply different absolute priority rules over different time slices. These options and their implications for the quality-of-service are currently being investigated.

4. END-TO-END CONTROL VIA DYNAMIC PROGRAMMING

For the general case of a multi-hop (multi-switch) network when there can are many streams of traffic to control, it is not easy to characterize the best control policies or even to specify a generally acceptable cost function. Nor is it obvious to determine what appropriate measurements could be, or would have to be, collected to design meaningful controls. Thus the general problem we are addressing is very challenging and can lead to some new insight and research issues.

For the sake of simplicity, we still limit our attention to the key issue of how to schedule cells for transmission and for rejection. One intriguing question is to determine whether static policies with limited information could in some cases perform as well (or perhaps even better) than "optimal" ones using global information, especially when traffic patterns fluctuate rapidly. The preliminary evaluation in this section will reveal that even though the computation of an optimal control rule cannot be carried out on-line (as would be expected), the insight it produces would provide useful guidelines for designing simple heuristic rules.

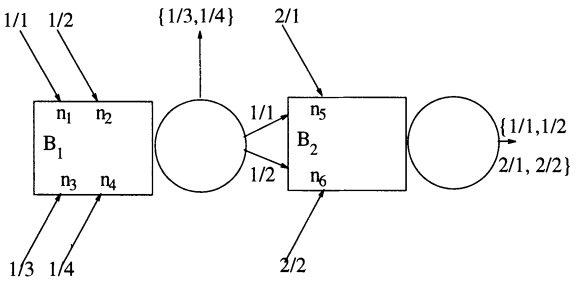

Figure 1. Model for Two Node Control Problem

In order to illustrate the issues which are involved in a more the more global optimization problem, we will now address the simplest and still interesting model of a multi-node

control problem involving two nodes shown on Figure 1.

We assume that the cost function used is based on a linear functions of cell loss. We also assume that global information is available to a central controller. In future work we will address the practically more interesting case when only partial information is available to each node. As we shall see, even this model (using global information) gives us very valuable insight into what an optimal control under local information would be if supplemented with even marginal global information about the other nodes.

In Figure 1, we show four classes of external traffic at node 1 and two classes of external traffic at node 2. We denote the external traffic by i/j, where i is the node and j is the class of traffic. Specifically, 1/1 is "end-to-end" traffic, and cells from this class pass through nodes 1 and 2 before exiting the system. Traffic class 1/2 is identical to 1/1 but it is more costly to lose a cell of 1/2 compared to 1/1. 1/3 and 1/4 are traffic streams that use only node 1 and then exit the system, with 1/3 being cheaper than 1/4 to lose. Thus, 1/3 and 1/4 constitute the "cross" traffic at node 1. 2/1 and 2/2 are streams that use only node 2 and then exit the system. There is a shared finite buffer at nodes 1 and 2, and the buffer sizes are denoted as B_1 and B_2 respectively. The arrival processes are all assumed to be Poisson and the arrival rate of stream i/j will be denoted as λ_{ij}. The cost of losing a cell of type ij will be denoted C_{ij}. We assume for reasons given later that $C_{11} = C_{13} = C_{21}$ and that $C_{12} = C_{14} = C_{22}$. Another important simplifying assumption that we make is that the propagation delay for a cell to travel from node 1 to node 2 is negligible. This assumption is somewhat unrealistic in most fast ATM networks where the propagation delay can be a large part of the delay experienced by a cell while traversing the network.

The dynamic program will be set in discrete time, $t = 0, 1, 2, \ldots$, and the time to transmit a cell will be one time unit. We suppose that rejection decisions are made at instants $0^-, 1^-, \ldots$ and scheduling decisions are made at $0^+, 1^+, \ldots$. For convenience we assume that $0, 1, ..$ correspond to $0^+, 1^+, \ldots$.

The state space of the optimization problem will be given by,

$$\{n_1, n_2, n_3, n_4, n_5, n_6\}, \sum_{i=1}^{4} n_i \leq B_1, \sum_{i=5}^{6} n_i \leq B_2, n_i = 0, 1, 2, \ldots,$$

where $n_i, i = 1, \ldots, 4$ correspond to the number of cells at node 1 from class 1/1, 1/2, 1/3 and 1/4 respectively; n_5 corresponds to the total number of class 1/1 and 2/1 cells at node 2; and n_6 to the total number of cells of class 1/2 and 1/4 at node 2. The assumption about the costs was made partially to simplify the state space representation at node 2, as is evident from the above description.

The transition probabilities between states of the state space will definitely depend on the controls allowed for cell scheduling and rejection. For simplicity we side-step the rejection decision by assuming that $C_{12} = C_{22} = C_{14} >> C_{11} = C_{21} = C_{13}$. This assumption means that there are two kinds of traffic in the network: a "costly" kind and a "cheap" kind. By the assumption that cells of one kind are much more valuable compared than the other cells, we side-step the rejection decision as follows. It is clear that cells from class 1/4 will be preferred to cells from 1/2 because 1/4 has a shorter route and is just as costly to lose. Similarly, 1/3 will be preferred over 1/1. Because $C_{12} >> C_{13}$, cells from 1/2 will be preferred over those from 1/3. So we have a complete ordering at

node 1 as far as rejection is concerned: $1/4 > 1/2 > 1/3 > 1/1$. At node 2, the static priority rule discussed in [1] will be still valid.

Even with all these simplifications, the dynamic program is computationally difficult to solve. The control is now to decide which cell to schedule for transmission at node 1. However in principle node 1 need not transmit any cell, even when there is work to be done. The transition probabilities will result from this control, the optimal static control at node 2, and the arrival rates as well as the rejection rule. The computational details are omitted from this paper.

4.1. Numerical Computation of the Controls

To simplify the numerical computations we truncate the Poisson distribution by allowing at most eight arrivals during one time slot. The state space of the dynamic program can still be extremely large for large buffer sizes and given time constraints on our computing facilities, we have worked with small buffer sizes as given below. We solve for the optimal average cost and compare it with the average cost using the optimal single node static priority policy (modified slightly as described below) at both nodes. The dynamic program uses successive iterations to solve for the value function and 100 iterations were employed. The details of the recursive computations for the value function are not given since the interested reader can refer to any standard work on stochastic dynamic programming (such as S. Ross [19]).

The number of iterations was chosen to meet the time constraint of solving a program within 2 to 4 hours of running time on a DEC Alpha 3000 Model 500 workstation and obtaining at least convergence to the third decimal place in the average cost. Clearly this approach is not feasible in real time. However it does provide a useful comparison for simple static policies, and it does provide insight into some simplified rules which may be used to mimic the optimal control.

The optimal static priority policy for a single node will not distinguish between $1/1$ and $1/3$ at node 1, but due of the nature of the routing, we modified the static policy to prefer $1/3$ over $1/1$ at node 1. The optimal static policy was used for node 2. We set the parameters for the dynamic program as in Table 1. With the arrival rates shown

Table 1
Parameters for the Dynamic Program

Traffic Type i/j	Cost C_{ij}	Arrival Rate λ_{ij}
1/1	1	0.08
1/2	2	0.16
1/3	1	0.32
1/4	2	0.24
2/1	1	0.32
2/2	2	0.48

in Table 1 the load will be 80% at node 1 and 104% at node 2. The buffer sizes were

varied by setting $(B_1, B_2) = (2, 2)$, $(5, 5)$ and $(7, 7)$. The results from solving the dynamic program are shown in Table 2. The column under optimal cost gives the average cost estimate after 100 iterations, the costs under the static policy were also obtained similarly using dynamic programming. The column labeled *deviations* gives the percentage of the total number of states in which the optimal policy deviated from the static policy. Two interesting points can be made from this example:

1. The percentage of the cases in which the optimal policy deviates from the static policy is not too large ($< 25\%$) and
2. The improvement from using the optimal policy over the static policy increases with the buffer size.

Table 2
Comparison of Static Policy to the Optimal Policy using the Dynamic Program

Buffer Sizes (b_1, b_2)	Static Cost	Optimal Cost	Percent Improvement	Deviations
(2,2)	0.390	0.377	3.4	11.11%
(5,5)	0.133	0.114	6.7	24.34%
(7,7)	0.095	0.077	23.4	23.25%

The first point is noteworthy because it implies that we can cleverly modify the static policy by adding some information about the state of node 2 to the information already available at node 1 and possibly obtain improvements. The second point is of interest because it implies that when losses are very small, it can be very beneficial to use the optimal control. The practical case of interest is obviously when when losse probabilities are small, say 10^{-7}. Variations in the problem parameters other than the buffer size give very similar results to those shown in Table 1. Based on these findings we were led to ask two questions:

1. What differentiates the optimal policy from the static policy, and
2. How does the optimal policy minimize losses and in what sense ?

Apart from some nuances that seem to be applicable only to networks with small buffers, the optimal policy changes the scheduling rule at node 1 by using information about node 2's overall load. The rule changes as follows:

- When the buffer at node 2 is relatively empty, say only 15 to 20% of the buffer is occupied, then the optimal policy schedules 1/2 in preference to 1/4 and 1/1 in preference to 1/3, and the overall scheduling priority at node 1 can be described by $1/2 > 1/4 > 1/1 > 1/3$, and
- When the buffer at node 2 is relatively full, say 60 to 70% percent occupied, then the optimal policy does not schedule 1/1 or 1/2 and schedules only 1/3 and 1/4.

- When the buffer in both nodes are close to full, and there are only cells of type 1/1 and 1/2 in the buffer at node 1, then the policy transmits cells preferring 1/1 to 1/2.

The second rule means that the scheduling policy at node 1 is no longer work conserving. The modification of the static rule using these insights will be called the **HILO** heuristic.

We first simulate the same two-node example using the Poisson process to model arrivals for the different traffic classes. The run lengths are 1000000 time units. We compare two policies: (*i*) The HILO heuristic, and (*ii*) the Static policy described earlier. For the HILO heuristic, when the number of cells in the buffer at node 2 is above a preset level, called **HI**, the scheduling rule is $1/4 > 1/3$ and never to schedule 1/1 or 1/2; when the level is below a preset limit, called **LO**, the rule is to schedule according to $1/2 > 1/4 > 1/1 > 1/3$, and when it is in between these limits the rule is to use the static priorities $1/4 > 1/2 > 1/3 > 1/1$. When the buffer occupancy at node 2 is above **HI**, the only cells in buffer at node 1 are of type 1/1 and 1/2, and the number of cells in the buffer at node 1 are above a level denoted as **FULL**, cells of type 1/1 are transmitted in preference to cells of type 1/2. Denote the buffer sizes as B_1 and B_2, respectively. In our simulations, we set **LO** to 20% of B_2, **HI** as 70% of B_2, and **FULL** as 90% of B_1. The static rule has been discussed earlier.

We simulated the two-node system with Poisson traffic for two examples. The parameters used in the simulations for the two examples are given in Table 3, and the results are compared in Table 4. From these results, we can conclude the Static rule saves cheaper cells but does not reduce overall cell loss when compared to FIFO. The HILO heuristic reduces overall cell loss by one order of magnitude.

Table 3
Two Examples for Poisson Arrivals

Buffer Size		Arrival Rates					
B_1	B_2	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{21}	λ_{22}
30	30	0.07	0.14	0.28	0.21	0.28	0.42
30	30	0.21	0.21	0.14	0.14	0.2	0.3

A little experimentation convinced us that the performance was relatively insensitive to changes in these levels. As anticipated the losses truly decline with the buffer size, and amazingly we get up to 18 times improvement with limited feedback. Similar results were obtained for other values of the problem parameters and are not reported here. We did observe that the HILO heuristic is effective only when the losses are relatively small. Moreover the rule can be improved by more descriptive feedback such as counting a cheap cell as half a cell and a costly one as one cell, i.e. by counting cells in the proportion of costs. We do not dwell on such issues because the key issues are more practical in nature, such as will the HILO heuristic perform well in more complex networks, whether the improvements will be of the same order of magnitude, will limitations due to the large

Table 4
Comparison of Cell Losses for Poisson Arrivals

Traffic Type	Example 1		Example 2	
i/j	HILO	Static	HILO	Static
1/1	0	0	11	0
1/2	0	0	0	0
1/3	0	0	0	0
1/4	0	0	0	0
2/1	16	278	0	433
2/2	0	0	0	0
Total for all types	16	278	11	433

bandwidth-delay product in ATM networks inhibit the use the feedback and lead to the consideration of cells already transmitted and not yet received by node 2.

5. Conclusions

We have considered issues of optimal cell (or constant length packet) scheduling and rejection to minimize loss oriented cost functions.

We have derived optimal policies for a single node with multiple traffic streams. Then we have discussed the design of optimal controls for multiple node systems with cross traffic. For the latter case we have examined a two-node example using a dynamic programming formulation. We have observed that even though the computational cost of such an approach is prohibitive, it can be useful for two reasons. First it provides an evaluation of how well (or poorly) simple static policies perform compared to the optimal. Secondly, the dynamic programming formulation provides insight on the design of simple policies which mimic the optimal schedule. This points to new research avenues in designing sub-optimal schedules for future high-speed ATM networks.

REFERENCES

1. E. Gelenbe, S. Seshadri and V. Srinivasan, "Pathwise Optimum Policies for ATM Cell Scheduling and Rejection ," Department of Computer Science, Technical Report, Duke University, CS-1994-24, 1994.
2. G. Hebuterne and A. Gravey, "A Space Priority Queueing Mechanism for Multiplexing Channels," **Computer Networks and ISDN Systems,** 20, 1990, 37-43.
3. I. Cidon, R. Guérin and A. Khamisy, "On Protective Buffer Policies," *Proc. INFOCOM '93,* 1051-1058, 1993.
4. A.Y-M. Lin, and J. A. Silvester, "Priority Queueing Strategies and Buffer Allocation Protocols for Traffic Control at an ATM Integrated Broadband Switching System," **IEEE Journal on Selected Areas in Communications,** Vol. 9, No. 9, Dec. 1991, 1524-1536.

5. D.W. Petr, and V.S. Frost, "Optimal Packet Discarding: An ATM-Oriented Analysis Model and Initial Results," *Proc. INFOCOM '90,* 537-542, 1990.
6. L. Tassiulas, Y. Hung and S.S. Panwar, "Optimal buffer control during congestion in an ATM network node," *Proc. INFOCOM '93,* 1059-1065, 1993.
7. Y-H. Jeon, "Conservation Laws and Multiplexing Schemes for Delay and Loss Requirements in Broadband Networks," Ph.D thesis, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, 1992.
8. L.P. Clare and I. Rubin, "Performance Boundaries for Prioritized Multiplexing Systems," IEEE Transactions on Information Theory, Vol. IT-33, No. 3, May 1987, 329-340.
9. L. Georgiadis, R. Guérin and A. Parekh, "Optimal Multiplexing on Single Link: Delay and Buffer Requirements," to appear in *Proc. INFOCOM '94*, 1994.
10. J.G. Shanthikumar and D.D. Yao, "Multiclass Queueing Systems: Polymatroidal Structure and Optimal Scheduling Control," **Operations Research,** Vol. 40, Supp. No. 2, May-June 1992.
11. J.M. Harrison, "Brownian Models of Queueing Networks with Heterogeneous Customer Populations," *Stochastic Differential Systems, Stochastic Control Theory and Applications,* ed. by W. Fleming and P. Lions, IMA Vol. 10, Springer-Verlag, Berlin 1988, 147-186.
12. C.N. Laws and G.M. Louth, "Dynamic Scheduling of a Four Station Queueing Network," **Probability in Engineering and Information Sciences,** 4, 1990, 131-156.
13. L.M. Wein, "Optimal Control of a Two-Station Brownian Network," **Math. Oper. Res.,** 15, 2, 1990, 215-242.
14. P. Yang, "Pathwise Solutions for a Class of Linear Stochastic Systems," *Doctoral Dissertation, Stanford University,* 1988.
15. E. Gelenbe and I. Mitrani, *"Analysis and Synthesis of Computer Systems,"* Academic Press, London, 1980.
16. K.L. Chung, *"A Course in Probability Theory,"* Academic Press, New York, 1974.
17. J.A. Buzacott, and J.G. Shanthikumar, *"Stochastic Models of Manufacturing Systems,"* Prentice Hall, Englewood Cliffs, N.J, 1993.
18. A. Federgruen and H. Groenvelt, "Characterization and Optimization of Achievable Performance in Queueing Systems," **Operations Research,** 36, 733-741.
19. S. Ross, *"Introduction to Stochastic Dynamic Programming,"* Academic Press, 1983.