Optimal Space Priority Policies for Shared Memory ATM Systems *

Rajarshi Roy and Shivendra S. Panwar †
Center for Advanced Technology in Telecommunications
Polytechnic University
6 Metrotech Center
Brooklyn, NY 11201

July 10th, 1997

Abstract

In this paper we study the problem of the optimal design of buffer management policies for a shared memory ATM switch or demultiplexer. A system with cells of two different space priorities is considered. Our objective is to determine the optimal policy that minimizes the total weighted cell loss. The problem of finding the optimal policy within the class of *pushout* and *expelling* policies is considered. Using sample path techniques the search space for the optimal policy is reduced to a subset of the entire policy set for each policy class. A numerical study based on value iteration technique is used to investigate the structure of the optimal policy. This technique is also used to calculate the loss probabilities for different classes of cells for a system with small buffer size.

Keywords: Markov Decision Theory, Sample path techniques, Shared Memory Switch, Buffer Management, ATM.

1 Introduction

Most of the ATM switch architectures that have been proposed in the literature use some buffering to queue cells whose service has been delayed due to contention for resources within the switch. The location of these buffers and the buffer management policy affects the switch performance i.e., switch throughput and delay of cells in the switch to a great extent. Performance studies of different types of switches were done previously and it was shown that output queueing with completely shared buffering achieves the optimal throughput-delay performance.[8]

Some buffer management is necessary to ensure that an output-queued shared-memory switch or demultiplexer does not perform poorly when some of the output queues get overloaded. In such a case, without any control the overloaded queue will consume most of the buffer spaces forcing the other lightly loaded queues to incur loss. To ensure efficient sharing of the buffer one needs to develop buffer sharing schemes to ensure that buffer space is shared properly among

*This research was supported by the New York State Center for Advanced Technology in Telecommunications, Polytechnic University.

†Tel: 718 260 3740, Fax: 718 260 3074, e-mail: panwar@kanchi.poly.edu

different logical queues and high priority cells are selectively favored over low priority cells in case of contention.

Let us describe the system model and the two different classes of buffer management schemes we will consider. The shared memory switch is modeled by a multi server queue with bounded buffer. The entire buffer is partitioned into two parts: the main buffer of size B_M and the temporary buffer of size B_T. Time is slotted and transmission of a cell takes one time slot. During one time slot at most B_T cells may arrive to the system and they are placed in the temporary buffer which has size B_T. The cells may belong to different traffic types. This assumption is consistent with the fact that a leaky bucket policed source may inject cells with different priority levels even if they belong to the same virtual circuit. A cost is incurred for each cell that is dropped from the system. The cost is higher for high priority classes. The objective is to minimize the long run average cost that is incurred from the lost packets. Depending on the available control we have over the dropping of cells from the temporary or main buffer and the placement of cells in the main buffer, we distinguish two classes of policies.

The first class of policies considered are *pushout* policies. A pushout policy is allowed to discard cells from the main buffer in order to make space available for cells in the temporary buffer which cannot enter the main buffer because it is full. A cell from temporary buffer cannot be blocked from admission to the main buffer if there is space in the main buffer. We narrow down the search space for an optimal pushout policy in a system with two priority classes, with an arbitrary discrete arrival process and with an arbitrary number of logical queues. We prove by sample path techniques that the optimal policy within the pushout class G_P has to lie within the class of *squeeze-out policies* G_{P_1}, which is defined later.

The second class of policies considered as *expelling* policies. Expelling policies are allowed to discard cells from the main buffer or block cells in the temporary buffer from admission into the main buffer irrespective of system state. Properties of the optimal expelling policy are obtained that narrow down the the class of candidate policies considerably in a system with two classes. More specifically speaking, it is shown that according to the optimal *expelling* policy one should apply the *squeeze-out* rule for accommodating cells of each logical queue in the portion of the main buffer allocated to it. While serving the queue, if the cell to be served next is of high priority then it is always served, and if the cell to be served is of low priority then that is either served or all the low priority cells residing at the head of that logical queue are dropped and the first available high priority cell is served.

In both type of policies it is assumed that the cells which enter the main buffer in every slot should join the end of their respective logical queues and rearrangement of cells that violate FIFO service is not allowed.

There has been considerable amount of prior work in this area. The queueing analysis of different buffer sharing schemes and their relative merits was analyzed in [1]. The problem of designing optimal policies for the purpose of optimizing certain performance criteria is considered in [2]. They only consider the problem of searching for optimal policies within the class where cells can only be blocked at the entry point of the buffer; dropping cells once it is accepted in the switch is not allowed. In [3] *pushout* is allowed and new arriving cells can push out cells from the longest logical queue. This policy turns out to be optimal only for a system with symmetric arrival and service processes. Guerin, Cidon, Georgiadis and Khamisy [4] considered a Poisson arrival, single loss priority and exponential service model for sharing memory in a switch with two output ports. They have established the optimality of a threshold based pushout scheme using Markov decision theory.They also calculated the values of those thresholds. For a N ported system, $N \geq 3$, they have results only for balanced input rates and equal service rates at all the output ports. The non-optimality of [3] in the case of unequal service rates in a two ported system is shown in their work. They do not consider packets with different loss priorities. For a N-ported system they have results only for the symmetric case.

Hung et al in [5] has provided optimal policies within the class of *discarding*, *pushout* and *expelling* policies using dynamic programming and sample path based techniques. They consider the case of general arrival models for the case of a single output port.

Hahne and Choudhury [6] have proposed a set of pushout and buffer sharing rules. They have proposed a backpressure mechanism for sharing buffer across stages and a pushout scheme for sharing buffer among different logical queues sharing a common memory. They have also applied their policy on traffic with multiple priorities. The lowest priority cell at the head of the queue is selected for pushout. They have applied this work for a hierarchical switch system and Banyan based switch system [6], [7]. Their simulation study reveals that "Delayed Pushout" scheme works well under all load conditions.

Here we model a buffer of finite capacity which is being shared by several logical queues, each of which may contain high and low priority cells. Our model is described below.

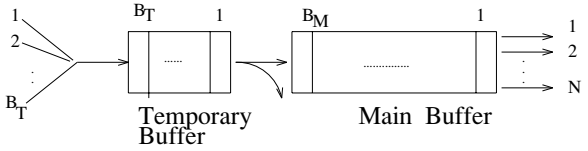

Figure 1: The system model

2 Model description

In our model, the two parts into which total memory of the switch/demultiplexer B is subdivided are the main buffer with capacity B_M and the temporary buffer with capacity B_T. It is a N ported, slotted system with packets destined to each output port or server constituting a logical queue. FIFO order is to be maintained within each of the logical queues once the cells are accepted in the main buffer, each of which contain high and low priority cells. Service time is deterministic and the servers serve one cell in every time slot if such a cell is available. In one slot at most B_T cells can arrive. Clearly for a switch $N = B_T$, if each input line can inject at most one cell in every time slot. The value of B_T in case of a demultiplexer will be dependent on the input line speed. At every decision epoch cells from the temporary buffer are either admitted to the main buffer, possibly by pushing out some cells from the main buffer, or dropped.

3 Pushout Policies

The class of *pushout* policies G_P is defined by the following rules: (a) A cell can be expelled from main buffer only if it is pushed out by another cell in the temporary buffer and the main buffer is full, (b) A cell from the temporary buffer can be discarded only if the main buffer is full, (c) A cell can never be dropped if there is room for it in the main buffer.

The class of policies G_{P_1} is defined as the following: Suppose we are given the number of buffer positions that will be allocated to a particular logical queue after the drop/pushout decision at a particular integer time in the main buffer is strictly less than the number of cells it had in the entire system just before the decision epoch t, i.e. at t^- a given number of cells are to be dropped/pushed out from each logical queue. Then the rules to be followed are (a) Append the cells that are in the temporary buffer and belong to logical queue n to the end of the main buffer in the allocated position, high priority cells first (if switch model) or in FIFO

order (if demultiplexer model). In switches input line speed is such that at every time slot only one cell can come at one input line. This implies that not more than one cell of the same virtual circuit can arrive in the same time slot. In demultiplexer model, it is possible that there is some high speed source that sends more than one cell in the same time slot. In ATM we are required to preserve the cell sequence. That is why reordering of cells can be allowed as they are placed from the temporary to the main buffer in the case of a switch model but cannot be done in the case of a demultiplexer model. The same reason dictates that reordering of cells is not allowed in the logical queues maintained in the main buffer. (b) If the amount of buffer allocated for logical queue n in the main buffer is full, and there are cells of that logical queue in the temporary buffer, push out the low-priority cells starting from those closest to the head of that logical queue, (c) If all low priority cells of that logical queue which are in the main buffer are pushed out, discard all remaining cells of that logical queue which are in temporary buffer. This entire procedure defines the *squeeze-out* class of policies policy.

Clearly, $B = B_M + B_T$. The amount of buffering allocated to one logical queue i at time t is $B_i(t)$. It is evident that $B_i(t) \leq B_M$.

3.1 Problem formulation

Cells are classified into two priority classes. The high priority class is more sensitive to cell losses. Without loss of generality we assume that the priority of class 1 is higher than the priority of class 2. The priority of a class is reflected by the cost that is incurred by the dropping of a cell of that class. We denote by $X_{n,i}(t)$ the class of the cell residing at the i-th place of the n-th logical queue by the end of the slot $[t-1, t]$. Here $i = 1, 2, ..., L_n(t)$, where $L_n(t)$ is the number of buffer places the logical queue n occupies at t^-. Clearly, $\sum_{n=1}^{N} L_n(t) = B$. $X_1(t), X_2(t), ..., X_N(t)$ is the vector that can be taken as a state, where $X_n(t) = (X_{n,i}(t) : i = 1, 2, ..., L_n(t))$. If there is any empty buffer places, they can be arbitrarily assumed to be part of a particular fixed logical queue and the value of $X_{n,i}(t)$ corresponding to them is 0.

We want to find out the policy π_1 amongst the class of *pushout* policies G_P so that $E_{\pi_1}(\sum_{t=0}^{\infty}(C_h \cdot D_h(t) + C_l \cdot D_l(t))|\text{any initial state } = x) \leq E_{\pi_2}(\sum_{t=0}^{\infty}(C_h \cdot D_h(t) + C_l \cdot D_l(t))|\text{any in initial state} = x)$, $\forall \pi_2 \in G_P$. Here, $D_h(t)$ is the number of high priority cells dropped from the system up to the end of slot $[t-1, t]$. $D_l(t)$ is the number of low priority cells dropped from the system up to the end of the slot $[t-1, t]$. C_h and C_l are positive real numbers, $C_h \geq C_l$. E_{π_i} the expectation when policy π_i is used. One can write with this value function the dynamic programming equations if the arrival statistics are completely characterized.

The objective of theorem 1, which is stated below is to show that the policy π^{p_0} which is optimal in G_P in the sense that $D_h^{\pi^{p_0}}(t) + D_l^{\pi^{p_0}}(t) \leq D_h^{\pi}(t) + D_l^{\pi}(t)$, $\forall$ integer time $t \in \{0, 1, 2, 3, ...\}$ and $\forall \pi \in G_P$ such that $\pi \neq \pi_{p_0}$ and also $D_h^{\pi^{p_0}}(t) \leq D_h^{\pi}(t)$; lies in $G_{P_1} \subset G_P$. Here, $D_h^{\pi}(t)$ is the number of high priority cells dropped upto time t under policy π and $D_l^{\pi}(t)$ is the number of low priority cells dropped up to time t under that policy.

Theorem 1: When the systems start with the same initial conditions and under policies π_1 and π_2 where $\pi_1 \in G_{P_1}$ and $\pi_2 \in G_{P_1}^c \cap G_P$ and the arrivals are identical then we have $D_h^{\pi_1}(t) \leq D_h^{\pi_2}(t)$ and $D_h^{\pi_1}(t) + D_l^{\pi_1}(t) \leq D_h^{\pi_2}(t) + D_l^{\pi_2}(t)$, $\forall t \in \{1, 2, ...\}$.

The proof of this theorem depends on lemma 1.1 which is stated below. Here, we tag the n-th logical queue.

Lemma 1.1: For any policy $\pi_2 \in G_P$ but $\pi_2 \in G_{P_1}^c$ there exists a policy π_3, which acts similarly as a policy that belongs to G_{P_1} at $t = 1$ and if appropriately defined for all inte-

ger time $t \geq 1$, under same initial condition and same arrival statistics $D_h^{\pi_3}(t) \leq D_h^{\pi_2}(t)$ and $D_h^{\pi_3}(t) + D_l^{\pi_3}(t) \leq D_h^{\pi_2}(t) + D_l^{\pi_2}(t), \forall t \in \{1, 2, ...\}$.

Proof of the lemma and theorem given above is not given here for space restrictions.

3.2 Some Counterexamples

These are counterexamples to two sample-path dominance conjectures. These conjectures are true in expected value sense at least for certain values of the load parameters. That fact is revealed by the numerical results presented in the next section.

3.2.1 Counterexample 1

This is a counterexample to the conjecture that a high priority cell should always pushout a low priority cell, if there is any. Here it will be shown that a performance criteria such as given the previous section cannot be satisfied for every sample path of the input process if such a policy is followed. We consider a 2x2 switch with four main buffer places and two temporary buffer places. We refer to the policy that recommends pushout from the longest queue with low priority cell as the conjectured policy and the other one as an alternative policy. We start with the same initial state with one low priority cell destined to output 1, three high priority cells destined to output 2 and two high priority cells in the temporary buffer destined to output 2. Under the conjectured policy the low priority cell is dropped and one of the high priority cells are accepted from the temporary buffer. Under the alternative policy both the high priority cells are dropped from the temporary buffer. Now if two high priority cells come in the next time slot then at the end of slot two under both the policies we have the same state. If in all the future decision epochs we follow same actions for both the policies, then under the alternative policy we incur less total packet loss.

3.2.2 Counterexample 2

This is a counterexample of the conjecture that a low priority cell can never pushout a high priority cell under optimal policy and this is true for every sample path. Consider a system with main buffer size of four and temporary buffer size of two. The initial state is such that there are four high priority cells destined to output 2 in the main buffer and one low priority cell destined to output 1 in the temporary buffer. Under the conjectured policy we drop the low priority cell and under the alternative policy we pushout one high priority cell from the main buffer and admit the low priority cell. Under the conjectured policy we serve one high priority cell destined to output 2 and under the alternative policy we serve the low priority cell destined to output 1 and one high priority cell destined to output 2. If in the next slot two high priority cells destined to output 2 comes then at the end of that slot we come to the same state under both the policies and then in the future decision epochs both the policies can take the same actions at every slot. So the total cell loss under the alternative policy will be less.

These counterexamples does show that inter queue pushout policy investigation is more complex and the results are not sample path wise true but may be true in the expected value sense. So we did a numerical study using value iteration and the goal of the ongoing work is to find out analytical structures of the optimal policy using the numerical study as a supporting tool.

3.3 Numerical study for Pushout Policy

We considered a two-ported shared memory switch modeled as a queue with bounded buffer and two servers each with a constant service rate of one time slot/cell. At any time slot at most two cells can come into the system. The arrival process is i.i.d. and the cells can belong to either of the two logical queues with probabilities b_1 and b_2. Given the event that a cell belongs to logical queue 1, it can be of high or low priority with probability c_1 and c_2 and given the event that it belongs to logical queue 2, it can be of high or low priority with probability d_1 and d_2. We have here $b_1 + b_2 = 1$, $c_1 + c_2 = 1$ and $d_1 + d_2 = 1$. If we drop a high priority cell we incur a loss of C_h and if we drop a low priority packet we incur a cost of C_l. Our system has six buffer places of which two are temporary buffer places. At every decision epoch we have to make sure that after the cell dropping/pushout decision is taken we should not be left with more than four cells in the buffer because at most two cells can come between a decision epoch and the next one (they are one time slot apart). The cells which belong to one logical queue may or may not belong to the same virtual circuit, so we serve each logical queue in FCFS order.

For our numerical study we have assumed $B = 4$ and our arrival process is i.i.d. with the statistics that at any time slot 0, 1 or 2 packets can come into the system with probabilities a_0, a_1 and a_2 respectively. The sum of these probabilities is 1. We formulate the problem in the framework of Markov decision theory and seek to find out the optimal policy numerically that minimizes the total undiscounted expected cost over any finite horizon.

The numerical results reveal that pushout from the longest queue policy when we have only either high or low priority cells in the buffer is not optimal for an unbalanced load. Rather, the logical queues have thresholds k_1 and k_2 with their sum at least B. Thus if we need to drop cells we should check which logical queue has exceeded its threshold and if the new cells belong to that one then they get dropped. If it is the other one then these new cells are accepted and cells are pushed out from the other logical queue. For a balanced load when there are 3 cells of high priority for both the logical queues numerical computation shows that we should drop one from each logical queue. But for the case of 70 percent of the incoming traffic going to output port 1 we see that we should drop 2 cells from logical queue 1. This clearly shows the change of value of thresholds with the offered load.

4 Expelling Policies

The class of *expelling* policies G^E has as members all policies that append the new cells from the temporary buffer and do not rearrange them. This class of policies is even allowed to drop cells from the main buffer when it is not full. The only constraint is that FIFO service order must be maintained.

In the following we will prove that the optimal policy within the class G^E belongs to a subset of that class G^{EO}.

The class G^{EO} is defined as the following:

1. The cells of each logical queue are placed from temporary buffer to the portion of the main buffer allocated for it in that slot, high priority cells first. If they do not fit then low priority cells are expelled starting from those closest to the head of the queue.

2a. If the cell at the head of the queue is of high priority then it is served.

2b. If the cell at the head of the queue is of low priority then either that cell is served or all

the low priority cells from the head of the queue until the high priority cell closest to the head of the queue are expelled and that high priority cell is served.

Theorem 2: For every policy $\pi \in G^E \cap G^{EO^c}$ there exists a policy $\pi_1 \in G^{EO}$ such that if the system starts from the same initial state under the two policies and the arrival process is identical under the two policies we have
$D^{1h}(t) \leq D^h(t)$
$D^{1h}(t) + D^{1l}(t) \leq D^h(t) + D^l(t)$, $t = 1, 2, ...$
where $D^{1h}(t)$,$D^{1l}(t)$ is the number of lost cells of high and low priority respectively under π_1 and similarly for $D^h(t)$, $D^l(t)$ under π.

The proof of this theorem requires the following lemma.

Lemma 2.1: For every policy $\pi \in G^E$ there exists a policy π_2, which acts according to rules 1 and 2 and is defined appropriately for $t = 2, 3, ...,$, based on π, so that if the system starts from the same initial state and the arrival process is same under the two policies, then we have

$D^{2h}(t) \leq D^h(t)$
$D^{2h}(t) + D^{2l}(t) \leq D^h(t) + D^l(t)$, $t = 1, 2, ...$
where $D^{2h}(t)$,$D^{2l}(t)$ is the number of lost cells of high and low priority respectively under π_2 and similarly for $D^h(t)$, $D^l(t)$ under π.

Proof of the lemma and theorem is not given here due to space restrictions.

4.1 Numerical study for Expelling Policy

We used the system described in the numerical study for pushout policies and did see that at certain states all the low priority cells at the head of a logical queue are dropped and the high priority cell is served. The decision of whether to serve the low priority cell or to drop them depends on the number of high priority cells of that logical queue in the system for most of the cases. However, we did see that there are some pair of states such that both have more than one low priority cell at the head of the queue and with same number of high and low priority cells in the system, but the action chosen are different.

Blocking probability calculations using value iteration shows that for the same loss probability expelling policies can handle a higher load region. Expelling policy does allow us to improve the performance of high priority class compared to pushout policies at the cost of some degradation in the performance of low priority class. Here the gain of the Markov decision process is calculated, which gives us the loss probabilities of different class of cells. In the results presented in the tables $P(H_i)$ indicates the loss probability of high priority cells of the i-th logical queue and $P(L_i)$ denotes the loss probability of the low priority cells of the i-th logical queue. C_h is the cost of dropping a high priority cell and this quantity is varied to achieve different degrees of expelling low priority cells in favor of high priority cells. C_l, the cost of dropping a low priority cell is always 1.

References

[1] F. Kamoun and L. Kleinrock, "Analysis of Shared Finite Storage in a Computer Network Node Environment Under General Traffic Conditions," *IEEE Trans. on Communications*, Vol. COM-28, No. 7, July 1980.

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$3.51\text{x}10^{-4}$	$2.20\text{x}10^{-4}$	$2.20\text{x}10^{-4}$	$2.20\text{x}10^{-4}$
$P(L_1)$	$3.0\text{x}10^{-3}$	$1.01\text{x}10^{-2}$	$1.01\text{x}10^{-2}$	$1.01\text{x}10^{-2}$
$P(H_2)$	$3.51\text{x}10^{-4}$	$2.20\text{x}10^{-4}$	$2.20\text{x}10^{-4}$	$2.20\text{x}10^{-4}$
$P(L_2)$	$3.0\text{x}10^{-3}$	$1.01\text{x}10^{-2}$	$1.01\text{x}10^{-2}$	$1.01\text{x}10^{-2}$

Table 1: Offered Load=0.5; Splitting of Traffic=1:1; High:Low=9:1

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$1.02\text{x}10^{-5}$	$4.0\text{x}10^{-6}$	$1.13\text{x}10^{-6}$	$1.13\text{x}10^{-6}$
$P(L_1)$	$5.10\text{x}10^{-4}$	$5.0\text{x}10^{-2}$	$3.09\text{x}10^{-2}$	$3.09\text{x}10^{-2}$
$P(H_2)$	$1.02\text{x}10^{-5}$	$4.0\text{x}10^{-6}$	$1.13\text{x}10^{-6}$	$1.13\text{x}10^{-6}$
$P(L_2)$	$5.10\text{x}10^{-4}$	$5.0\text{x}10^{-2}$	$3.09\text{x}10^{-2}$	$3.09\text{x}10^{-2}$

Table 2: Offered Load=0.5; Splitting of Traffic=1:1; High:Low=1:1

[2] G. J. Foschini and B. Gopinath, "Sharing Memory Optimally," *IEEE Trans. on Communications*, Vol. COM-31, No. 3, March 1983.

[3] S. X. Wei, E. J. Coyle and M. T. Hsiao, "An Optimal Buffer Management Policy for High Performance Packet Switching," *IEEE GLOBECOM'91*, Vol. 2, pp. 924-928, December, 1991.

[4] I. Cidon, L.Georgiadis, R. Guerin, A. Khamisy, "Optimal buffer sharing," *IEEE JSAC*, september 1995, vol 13, No. 7, pp. 1229-1240.

[5] L. Tassiulas, Y. C. Hung and S. S. Panwar, "Optimal Buffer Control During Congestion in an ATM Network Node," *IEEE/ACM Transactions on Networking*, August 1994, Vol. 2, No. 4, pp. 374-386.

[6] Abhijit Choudhury, Ellen L. Hahne, "Buffer Management in a Hierarchical Shared Memory Switch," *IEEE INFOCOM'94*, Vol. 3, pp. 1410-1419, June, 1994.

[7] Debasis Basak, Abhijit K. Choudhury, Ellen L. Hahne, "Sharing Memory in Banyan-based ATM Switches," Preprint.

[8] Mark J. Karol, Michael J. Hluchyj and Samuel P. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, December 1987, Vol. 35, No. 12, pp. 1347-1356.

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$3.0\text{x}10^{-9}$	$6.0\text{x}10^{-10}$	$3.10\text{x}10^{-11}$	$2.0\text{x}10^{-12}$
$P(L_1)$	$6.0\text{x}10^{-4}$	$6.10\text{x}10^{-4}$	$7.02\text{x}10^{-4}$	$1.0\text{x}10^{-2}$
$P(H_2)$	$3.0\text{x}10^{-9}$	$6.0\text{x}10^{-10}$	$3.10\text{x}10^{-11}$	$2.0\text{x}10^{-12}$
$P(L_2)$	$6.0\text{x}10^{-4}$	$6.10\text{x}10^{-4}$	$7.02\text{x}10^{-4}$	$1.0\text{x}10^{-2}$

Table 3: Offered Load=0.5; Splitting of Traffic=1:1; High:Low=1:9

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$5.0\text{x}10^{-4}$	$1.01\text{x}10^{-4}$	$4.0\text{x}10^{-5}$	$4.0\text{x}10^{-5}$
$P(L_1)$	$1.7\text{x}10^{-2}$	$4.7\text{x}10^{-2}$	$1.0\text{x}10^{-1}$	$1.0\text{x}10^{-1}$
$P(H_2)$	$3.0\text{x}10^{-7}$	$7.0\text{x}10^{-9}$	$1.0\text{x}10^{-9}$	$1.0\text{x}10^{-9}$
$P(L_2)$	$2.4\text{x}10^{-4}$	$4.1\text{x}10^{-5}$	$4.0\text{x}10^{-3}$	$4.0\text{x}10^{-3}$

Table 4: Offered Load=0.5; Splitting of Traffic=4:1; High:Low=1:1

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$1.0\text{x}10^{-7}$	$2.0\text{x}10^{-8}$	$1.1\text{x}10^{-8}$	$4.0\text{x}10^{-11}$
$P(L_1)$	$1.8\text{x}10^{-2}$	$1.9\text{x}10^{-2}$	$2.0\text{x}10^{-2}$	$6.0\text{x}10^{-2}$
$P(H_2)$	$7.0\text{x}10^{-11}$	$4.0\text{x}10^{-12}$	$1.0\text{x}10^{-13}$	$8.01\text{x}10^{-15}$
$P(L_2)$	$7.6\text{x}10^{-6}$	$6.0\text{x}10^{-7}$	$4.0\text{x}10^{-6}$	$1.5\text{x}10^{-3}$

Table 5: Offered Load=0.5; Splitting of Traffic=4:1; High:Low=1:9

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$5.3\text{x}10^{-3}$	$3.6\text{x}10^{-3}$	$3.6\text{x}10^{-3}$	$3.6\text{x}10^{-3}$
$P(L_1)$	$3.51\text{x}10^{-3}$	$2.82\text{x}10^{-2}$	$2.82\text{x}10^{-2}$	$2.82\text{x}10^{-2}$
$P(H_2)$	$5.3\text{x}10^{-3}$	$3.6\text{x}10^{-3}$	$3.6\text{x}10^{-3}$	$3.6\text{x}10^{-3}$
$P(L_2)$	$3.51\text{x}10^{-3}$	$2.82\text{x}10^{-2}$	$2.82\text{x}10^{-2}$	$2.82\text{x}10^{-2}$

Table 6: Offered Load=0.7; Splitting of Traffic=1:1; High:Low=9:1

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$2.54\text{x}10^{-4}$	$6.32\text{x}10^{-5}$	$2.0\text{x}10^{-5}$	$2.0\text{x}10^{-5}$
$P(L_1)$	$8.45\text{x}10^{-3}$	$1.60\text{x}10^{-2}$	$7.58\text{x}10^{-2}$	$7.58\text{x}10^{-2}$
$P(H_2)$	$2.54\text{x}10^{-4}$	$6.32\text{x}10^{-5}$	$2.0\text{x}10^{-5}$	$2.0\text{x}10^{-5}$
$P(L_2)$	$8.45\text{x}10^{-3}$	$1.60\text{x}10^{-2}$	$7.58\text{x}10^{-2}$	$7.58\text{x}10^{-2}$

Table 7: Offered Load=0.7; Splitting of Traffic=1:1; High:Low=1:1

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$4.75\text{x}10^{-8}$	$1.11\text{x}10^{-8}$	$6.23\text{x}10^{-10}$	$2.49\text{x}10^{-11}$
$P(L_1)$	$8.58\text{x}10^{-3}$	$8.81\text{x}10^{-3}$	$9.40\text{x}10^{-3}$	$3.82\text{x}10^{-2}$
$P(H_2)$	$4.75\text{x}10^{-8}$	$1.11\text{x}10^{-8}$	$6.23\text{x}10^{-10}$	$2.49\text{x}10^{-11}$
$P(L_2)$	$8.58\text{x}10^{-3}$	$8.81\text{x}10^{-3}$	$9.40\text{x}10^{-3}$	$3.82\text{x}10^{-2}$

Table 8: Offered Load=0.7; Splitting of Traffic=1:1; High:Low=1:9

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$6.40\text{x}10^{-3}$	$6.70\text{x}10^{-4}$	$6.70\text{x}10^{-4}$	$6.70\text{x}10^{-4}$
$P(L_1)$	$1.56\text{x}10^{-1}$	$2.68\text{x}10^{-1}$	$2.68\text{x}10^{-1}$	$2.68\text{x}10^{-1}$
$P(H_2)$	$1.82\text{x}10^{-7}$	$1.68\text{x}10^{-8}$	$1.68\text{x}10^{-8}$	$1.68\text{x}10^{-8}$
$P(L_2)$	$2.81\text{x}10^{-3}$	$8.24\text{x}10^{-3}$	$8.24\text{x}10^{-3}$	$8.24\text{x}10^{-3}$

Table 9: Offered Load=0.7; Splitting of Traffic=4:1; High:Low=1:1

Loss Probabilities	Pushout Policy	Expelling Policy		
		$C_h = 10^3$	$C_h = 10^6$	$C_h = 10^{11}$
$P(H_1)$	$9.63\text{x}10^{-7}$	$2.85\text{x}10^{-7}$	$1.44\text{x}10^{-7}$	$4.75\text{x}10^{-10}$
$P(L_1)$	$1.63\text{x}10^{-1}$	$1.64\text{x}10^{-1}$	$1.68\text{x}10^{-1}$	$2.15\text{x}10^{-1}$
$P(H_2)$	$5.76\text{x}10^{-11}$	$5.73\text{x}10^{-12}$	$1.20\text{x}10^{-12}$	$9.0\text{x}10^{-14}$
$P(L_2)$	$3.01\text{x}10^{-6}$	$7.64\text{x}10^{-6}$	$1.53\text{x}10^{-5}$	$2.96\text{x}10^{-3}$

Table 10: Offered Load=0.7; Splitting of Traffic=4:1; High:Low=1:9