# Optimal Buffer Sharing

**Israel Cidon**[*] [†]
Sun Microsystems Labs Inc.
Mountain View, CA 94043-1100

**Leonidas Georgiadis**
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

**Roch Guérin**
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

**Asad Khamisy**[‡]
Department of Electrical Engineering
Technion, Haifa 32000, Israel

**Abstract**

We address the problem of designing optimal buffer management policies in shared memory switches when packets already accepted in the switch can be dropped (pushed-out). Our goal is to maximize the overall throughput, or equivalently to minimize the overall loss probability in the system. For a system with two output ports, we prove that the optimal policy is of *push-out with threshold* type (POT). The same result holds if the optimality criterion is the weighted sum of the port loss probabilities. For this system, we also give an approximate method for the calculation of the optimal threshold, which we conjecture to be asymptotically correct. For the $N$-ported system, the optimal policy is not known in general, but we show that for a symmetric system (equal traffic on all ports) it consists of always accepting arrivals when the buffer is not full, and dropping one from the longest queue to accommodate the new arrival when the buffer is full. Numerical results are provided which reveal an interesting and somewhat unexpected phenomenon. While the overall improvement in loss probability of the optimal POT policy over the optimal coordinate-convex policy is not very significant, the loss probability of an *individual* output port remains approximately constant as the load on the other port varies and the optimal POT policy is applied, a property not shared by the optimal coordinate-convex policy.

[*]This work was done while at the IBM T.J. Watson Research Center.
[†]And Department of Electrical Engineering, Technion, Haifa 32000, Israel
[‡]Part of the work was done while visiting the IBM T.J. Watson Research Center.

# 1 Introduction

Shared-memory fast packet switches are widely used in high-speed, wide-area networks [2, 6]. Such switches consist of a single large memory where packets arriving from all inputs are stored while they wait before being transmitted on their respective output(s). While this design presents a number of technical challenges, in particular memory access and speed, the sharing of a single memory by all input and output ports offers numerous advantages. One of them is improved buffer efficiency, which translates into smaller memory sizes to satisfy a given loss probability requirement. However, despite this greater efficiency, losses remain unavoidable and it is, therefore, still of interest to understand how they can be minimized. Furthermore, sharing of the memory also introduces new potential problems as individual inputs can now affect the performance seen by others. In this paper, we focus on identifying how to best share the memory between the system ports, so that overall system throughput is maximized.

There has been a number of prior works which have addressed this problem. In particular, Kamoun and Kleinrock [4] analyzed several sharing schemes, namely, Complete Sharing (CS) in which an arriving packet is accepted if any storage space is available, Complete Partitioning (CP) in which the entire storage is permanently partitioned among the output ports, Sharing with Maximum Queue Lengths (SMXQ) in which a limit on the number of buffers allocated to each output port is imposed, Sharing with a Minimum Allocation (SMA) in which a minimum number of buffers is always reserved for each output port and the remaining buffers are shared between all output ports, and Sharing with a Maximum Queue and Minimum Allocation (SMQMA) which is a combination of the SMXQ and SMA schemes. Their study assumed independent Poisson arrivals and exponential service times and they obtained closed form expressions for the probability distribution of the buffer occupancy, based on the fact that it has a well-known product form solution. From their numerical examples, they showed that sharing can improve performance especially when little storage is available, but that some restrictions should be imposed to avoid throughput degradation in asymmetric systems. Additional numerical results [10] for a CS policy but with bursty arrivals, further supported this conclusion by showing that some outputs can become temporarily congested and monopolize the use of the shared memory.

The existence and the structure of an optimal sharing policy (in the sense of minimum packet loss or maximum throughput) was then first investigated by Foschini and Gopinath [3]. They considered optimality within the class of policies that never drop a packet once they admit it in the buffer, and hence have coordinate-convex state space $\Omega$ (if $x \in \Omega$, then $(x_{1,}x_2, \ldots, (x_i - 1)^+, \ldots, x_m) \in \Omega$ for all $i = 1, \ldots, N$). These policies, referred to as coordinate-convex policies, include the policies of [4]. For a switch with two output ports they proved that the optimal coordinate-convex policy is to limit the queue length of output port $i$, $i = 1, 2$ to some fixed level $m_i$, such that $m_1 + m_2 \geq B$, where $B$ is the buffer size. For more than two ports they conjectured that the optimal policy is *simple* (see definition in [3]). Their proofs were based on the fact that the probability distribution of the buffer occupancy has a product form solution.

Wei, *et al.,* [12], suggested a sharing policy which allows for the dropping of accepted packets, and therefore does not belong to the class of coordinate-convex policies. According to this policy (named, Drop-on-Demand or DoD), an arriving packet is always accepted if there is an empty buffer. If a packet destined for output port $i$ arrives and finds the buffer full and output port $l$ has more packets in the shared-memory than any other ports, the following action is taken: if $i = l$, the arriving packet is dropped; if $i \neq l$, the arriving packet joins the buffer and one port $l$ packet is dropped. In general, policies which can accept an arriving packet by dropping another packet from the system are known as *push-out* policies (see, e.g., [5]). Push-out policies include coordinate-convex policies (never push-out a packet) as well as the DoD policy. In [12], numerical examples were provided showing that the DoD policy yields better throughput and lower packet losses than either the CS and CP policies. However, as we shall show, this policy is optimal only for symmetric systems.

In this paper we consider a model similar to the one of [4]. The buffer size is denoted by $B$, and the arrival and service processes of type $i$ (destined to output $i$) packets are Poisson and exponential with rates $\lambda_i$ and $\mu_i$, respectively. Upon arrival of a packet the system can decide to either accept the packet, or reject it, or accept it and drop another packet from the system. In other words, we include pushout policies and our goal is to determine the policy which maximizes the overall throughput, or equivalently minimize the overall loss probability.

For a two-ported switch, we prove that the optimal policy is of *push-out with threshold* type (POT), i.e., whenever the buffer is non-full, the arrival should be accepted, and whenever it is full, an arrival from type $i$, $i = 1, 2$, is accepted and a type $\bar{i}$ (the other type) packet is pushed-out if the number of type $i$ packets is below some threshold $k_i^*$ (where $k_1^* + k_2^* = B$). The same result is true if the optimality criterion is the weighted sum of the port loss probabilities. For $\mu_1 = \mu_2$ and $\lambda_1 \geq \lambda_2$ we also show that $k_1^* \leq B/2$. In general, the determination of the threshold $k_1^*$ is computationally intensive, but for the two-ported system we develop a simple and yet reasonably accurate heuristic to obtain its value. The results for the two-ported system establish the non-optimality of DoD for asymmetric systems.

For the symmetric $N$-ported system with identical arrival rates and identical transmission rates, we show that the optimal policy is to accept an arrival whenever the buffer is non-full, or the queue corresponding to the type of the arriving packet is not the largest; in the second case a packet from the longest queue is dropped. This establishes the optimality of DoD for the $N$-ported symmetric system. The proofs of the results for both the two-ported and $N$-ported systems are based on the theory of Markov decision processes.

The behavior of the optimal policies are then investigated for the two-ported case by means of numerical examples, which reveal an interesting and somewhat unexpected phenomenon. While the overall improvement in loss probability of the optimal POT policy over the optimal coordinate-convex policy is found to be relatively minor, a significant difference is observed when focusing on the loss probability of an *individual* output port. The use of the optimal POT policy results in an approximately constant loss probability on a given port as the load on the other varies. In contrast, significant variations can be observed with the optimal coordinate-convex policy. The insensitivity of individual losses is clearly a desirable feature, but nevertheless surprising given the global nature (overall throughput) of our optimization. For the two-ported system, we also investigate the heuristic method for determining the threshold of the optimal POT policy, which based on the numerical results obtained is conjectured to be asymptotically correct as the buffer size $B$ increases. Numerical comparisons further show that the approximation is very good for most practical scenarios.

Finally, we note that the structure of the Markov process arising from a POT policy permits the development of an efficient method for the computation of the state probabilities. The method consists in reducing the $(B+1)^2/2$ system of equations to the solution of a system of $(B+1)$ equations.

The paper is organized as follows. In section 2 we introduce the system model and provide the formulation of the optimization problem. In section 3 we investigate the structure of the optimal policy. We first focus on the two-ported system for which we prove that the optimal policy is of POT type. We also derive a number of interesting properties of the optimal policy, and describe the approximation we propose to compute the optimal threshold. Next we consider the more general $N$-ported system and also identify the optimal policy, but only for the symmetric case (arrival and service rates are the same at all ports). Section 4 is devoted to numerical comparisons between the performance of the optimal POT and coordinate-convex policies. We concentrate again on the two-ported system, for which we also study the accuracy of the proposed threshold approximation. Section 5 briefly summarizes the findings of the paper and suggests some open problems. Finally, appendix A provides proofs of lemmas used in section 3, and appendix B outlines an efficient method for computing state probabilities in a two-ported system operating under the POT policy.

## 2 The Model and Problem Formulation

The system consists of a buffer shared by packets destined to any of $N$ output ports. Packets are said to be of type $i$, $1 \leq i \leq N$, if they are destined to port $i$. Type $i$, $1 \leq i \leq N$, packets arrive to the buffer according to a Poisson process with rate $\lambda_i$, and are transmitted by output port $i$ with a transmission time which is exponentially distributed with rate $\mu_i$. We assume that $\lambda_i, \mu_i < \infty$ so that only a finite number of transitions can occur in any finite interval of time, and that packet inter-arrival and transmission times from all sources are mutually independent. The total buffer size is taken to be $B$ packets, and a packet occupies its buffer until it has been completely transmitted.

Our goal is to determine how the $B$ buffers are "best" shared among packets of different types, so that the overall system throughput is maximized. This amounts to identifying rules that specify when and how packets of different types are allowed to occupy a space in the shared buffer. In this paper, acceptable rules include accepting or rejecting an arriving packet as well as discarding (pushing-out) an already stored packet to accommodate an arriving one. Because the state of the system can be represented by a Markov chain, the rules or policy governing the sharing of the buffer can be expressed as a continuous time Markov decision process. Decision epochs correspond to arrivals and departures from the system, where at each epoch, a decision is made as to whether a current or future packet should be accepted, rejected, or accepted by pushing-out another packet from the system. Next, we proceed with a precise formulation of this process.

Let $x(n) = (x_1(n), x_2(n), ..., x_N(n))$ be the state of the system at decision epoch $n = 0, 1, \ldots$, where $x_i(n)$, $1 \le i \le N$, denotes the number of type $i$ packets in the system at decision epoch $n$. Let $X = \{0, 1, ..., B\}^N$ be the state space of the system. Define the following operators to denote the rejection, acceptance and push-out of packets, respectively (upon arrival epoch):

$$\begin{aligned} P_r(x) &= x \\ P_{a_i}(x) &= x + e_i \qquad x \in \mathrm{Dom}(P_{a_i}) \qquad 1 \le i \le N \\ P_{p_i^j}(x) &= x + e_i - e_j \qquad x \in \mathrm{Dom}(P_{p_i^j}) \qquad 1 \le i \ne j \le N \end{aligned} \tag{1}$$

where, $e_i$, $1 \le i \le N$, denotes the vector with all components zero except the $i$th which is equal 1, $\mathrm{Dom}(P_{a_i}) \triangleq \{\sum_{j=1}^N x_j < B\}$ and $\mathrm{Dom}(P_{p_i^j}) \triangleq \{x_j > 0\}$.

Let $U = \{u = (u_1, u_2, ..., u_N) : u_i \in \{r, a_i, p_i^j, 1 \le j \ne i \le N\}\}$ be the set of possible decisions, and $U(x) = \{u \in U : x \in \cap_{i=1}^N \mathrm{Dom}(P_{u_i})\}$ be the set of all admissible actions when the system state is $x$. The specification of the continuous time Markov decision process is then completed once we have defined the length of time between successive decision epochs and the transition probability function. When the system is in state $x$, the length of time until the next decision epoch is an exponential random variable with transition rate $\sum_{i=1}^N (\lambda_i + \mu_i \mathbf{1}\{x_i > 0\})$, and the transition probability to the next state is given by

$$\Pr[x(n+1) = P_{u_i}(x(n)) \mid x(n) = x,\ u(n) = u] = \frac{\lambda_i}{\sum_{i=1}^N (\lambda_i + \mu_i \mathbf{1}\{x_i > 0\})} \qquad 1 \le i \le N$$

$$\Pr[x(n+1) = D_i(x(n)) \mid x(n) = x,\ u(n) = u] \;=\; \frac{\mu_i \mathbf{1}\{x_i > 0\}}{\sum_{i=1}^N (\lambda_i + \mu_i \mathbf{1}\{x_i > 0\})} \quad 1 \le i \le N, \qquad (2)$$

where $D_i(x(n))$ indicates a departure of a type $i$ packet when the system is in state $x$ at time $n$.

As mentioned earlier, the objective of the optimization is to minimize the overall loss probability, or equivalently to maximize the overall throughput of the system. That is, denoting by $n(T)$ the number of decision epochs up to time $T$, we are interested in the policy that for any initial state $x(0) = x$, maximizes

$$\lim_{T\to\infty} \inf \frac{1}{T} E \left( \sum_{k=1}^{n(T)} \sum_{i=1}^{N} d_i(k) \,|x(0) = x \right), \qquad (3)$$

where $d_i(k) = 1$ if at the $k$th decision instant a departure of type $i$ occurs and $d_i(k) = 0$ otherwise. The above expression can be generalized to a *weighted* average reward of the form

$$\lim_{T\to\infty} \inf \frac{1}{T} E \left( \sum_{k=1}^{n(T)} \sum_{i=1}^{N} w_i d_i(k) \,|x(0) = x \right), \quad w_i \ge 0, \quad 1 \le i \le N,$$

and the proof of Section 3 on the structure of the optimal policy actually goes through for this more general case. However, in order to keep notations simple we will assume that $w_i = 1,\ 1 \le i \le N$ in the rest of the paper.

The above continuous time problem needs to be translated into an equivalent discrete time optimization problem, and we use uniformization (see [7], [9], [1, section 6.7]) for that purpose. In our case, uniformization amounts to introducing, when the system is in one of the states $x \in \{x_i = 0,$ for at least one $i\}$, *fictitious* departure epochs of type $i$ which occur with rate $\mu_i \mathbf{1}\{x_i = 0\}$. The reward of a fictitious departure is taken to be zero. This modification does not alter the stochastic behavior of the process and results in identical transition rates, equal to $\sum_{i=1}^N (\lambda_i + \mu_i)$, independent of the state of the system and the decision taken. The transition probabilities in the new system are now given by

$$\begin{aligned} \Pr[x(n+1) = P_{u_i}(x(n)) \mid x(n) = x,\ u(n) = u] &= \frac{\lambda_i}{\sum_{i=1}^N (\lambda_i + \mu_i)} \quad 1 \le i \le N \\ \Pr[x(n+1) = D_i(x(n)) \mid x(n) = x,\ u(n) = u] &= \frac{\mu_i}{\sum_{i=1}^N (\lambda_i + \mu_i)} \quad 1 \le i \le N, \end{aligned} \qquad (4)$$

and the equivalent discrete time optimization objective is to find a policy that for any initial state $x(0) = x$, maximizes

$$\lim_{n\to\infty} \inf \frac{1}{n} E \left( \sum_{k=1}^{n} \sum_{i=1}^{N} d_i(k) \,|x(0) = x \right),$$

where $d_i(n) = 1$ if at the $n$th decision instant a *real* (non-fictitious) departure of type $i$ occurs and $d_i(n) = 0$ otherwise. In the next section, we consider the problem of identifying the policy that maximizes this cost function.

# 3 The Optimal Policy

To avoid cumbersome notation, in the following we assume without loss of generality that the arrival and departure rates are normalized so that, $\sum_{i=1}^{N}(\lambda_i + \mu_i) = 1$. A standard approach to solving the kind of optimization problem we consider, is to first work with the discounted cost criterion (see [8]),

$$E \sum_{n=1}^{\infty} \beta^n \sum_{i=1}^{N} d_i(n), \tag{5}$$

where $0 < \beta < 1$ is a discount factor.

In our system, since the rewards $d_i(n)$ are bounded, it is known [8] that there is always an optimal stationary policy to the discounted problem. Therefore, we limit our investigations to this class of policies. A stationary policy $\pi$ is a function $\pi : X \to U$ with $\pi(x) \in U(x)$ for every $x \in X$, and such that under $\pi$ the decision $u = \pi(x)$ is always taken whenever the system is in state $x$. In order to carry out our investigation of the optimal policy, we need to introduce the Banach space $\mathcal{F}$ of all bounded real functions $f : X \to R$ with norm $||\cdot||$ given by $||f|| = \max_{x \in X} |f(x)|$.

For any stationary policy $\pi$ we then define $T_\pi : \mathcal{F} \to \mathcal{F}$ (the dynamic programming operator) by

$$(T_\pi f)(x) = \sum_{i=1}^{N} \mu_i 1\{x_i > 0\} + \sum_{i=1}^{N} \beta \lambda_i f\left(P_{u_i}(x)\right) + \sum_{i=1}^{N} \beta \mu_i f\left(D_i(x)\right) \tag{6}$$

where, $\pi(x) = (u_1, u_2, ..., u_N)$. The first term in the right hand expression is the one-step cost for the system under policy $\pi$, while the rest of the terms correspond to the cost that incurs after the first step. Based on equation (6) we then define the operator $T$ by $(Tf)(x) = \max_{\pi}(T_\pi f)(x)$ for all $x$. If we denote the optimal cost starting from state $x$ by $J_\beta(x)$, the following results are well-known (see, e.g., [8]).

- For every $x \in X$, $J_\beta(x) = TJ_\beta(x)$.
- For any $f \in \mathcal{F}$, $\lim_{n\to\infty} T^{(n)} f(x) = J_\beta(x)$ for every $x \in X$, where $T^{(n)}$ is the $n-$fold composition of the operator $T$.
- A stationary policy $\pi$ is optimal iff $J_\beta(x) = T_\pi J_\beta(x)$ for every $x \in X$.

In the next sections, we rely on these results to identify the optimal policy for our system. We focus first on the two-ported system for which we show that the optimal policy is of POT type in the general case of different arrival and service rates on each port. We also prove some interesting properties of this policy in some cases and propose a simple heuristic to compute the optimal threshold value. The accuracy of this approximation is later evaluated in section 4. The $N$-ported system is treated next, but only for the symmetric case for which the optimal policy is identified.

## 3.1 The Two-Ported System ($N = 2$)

### 3.1.1 Optimal Policy Derivation

In order to determine the optimal policy for the two-ported system, it is necessary to specify enough of its properties so that it is fully characterized. The basic approach we employ to identify these properties is the value iteration method (see [8]). It is based on the fact that the optimal value function can be shown to obey a certain property simply by showing that if this property holds for a function $f \in \mathcal{F}$, then it continues to hold for the function $Tf$ (which also belongs to $\mathcal{F}$). The main difficulties in using this approach are in initially "guessing" the properties of the optimal policy, and in selecting appropriate auxiliary properties for the function $f$ which are usually necessary to prove that the desired properties hold.

The dynamic programming operator for the two-ported system is:

$$\begin{aligned} Tf(k_1,k_2) &= \mu_1 1\{k_1>0\} + \mu_2 1\{k_2>0\} \\ &+ \beta\mu_1 f((k_1-1)^+,k_2) + \beta\mu_2 f(k_1,(k_2-1)^+) \\ &+ \beta\lambda_1\Phi_1(k_1,k_2) + \beta\lambda_2\Phi_2(k_1,k_2) \end{aligned} \tag{7}$$

where,

$$\begin{aligned} \Phi_1(k_1,k_2) &= \max\{\ f(k_1,k_2), f(k_1+1,k_2-1)\ \} \quad if \quad k_1+k_2=B \\ \Phi_1(k_1,k_2) &= \max\{\ f(k_1,k_2), f(k_1+1,k_2), f(k_1+1,k_2-1)\ \} \quad if \quad k_1+k_2<B \end{aligned} \tag{8}$$

and,

$$\begin{aligned} \Phi_2(k_1,k_2) &= \max\{\ f(k_1,k_2), f(k_1-1,k_2+1)\ \} \quad if \quad k_1+k_2=B \\ \Phi_2(k_1,k_2) &= \max\{\ f(k_1,k_2), f(k_1,k_2+1), f(k_1-1,k_2+1)\ \} \quad if \quad k_1+k_2<B, \end{aligned} \tag{9}$$

where we define $\Phi_i(k_1,k_2)\stackrel{\Delta}{=}0$ for $k_1,k_2<0$ or $k_1,k_2>B$.

**Lemma 1** *The optimal value function $J_\beta(k_1,k_2)$ has the following properties:*

1. *Monotonicity and boundedness in* $k_1$*:* $0 \leq J_\beta(k_1+1, k_2) - J_\beta(k_1, k_2) \leq 1,\ 0 \leq k_1 \leq B-1$.

2. *Monotonicity and boundedness in* $k_2$*:* $0 \leq J_\beta(k_1, k_2+1) - J_\beta(k_1, k_2) \leq 1,\ 0 \leq k_2 \leq B-1$.

3. *Concavity along* $k_1$*:* $J_\beta(k_1+1, k_2) - J_\beta(k_1, k_2) \leq J_\beta(k_1, k_2) - J_\beta(k_1-1, k_2),\ 1 \leq k_1 \leq B-1$.

4. *Concavity along* $k_2$*:* $J_\beta(k_1, k_2+1) - J_\beta(k_1, k_2) \leq J_\beta(k_1, k_2) - J_\beta(k_1, k_2-1),\ 1 \leq k_2 \leq B-1$.

5. *Concavity along the line* $k_1 + k_2 = b,\ 2 \leq b \leq B$*:* $J_\beta(k_1+1, k_2-1) - J_\beta(k_1, k_2) \leq J_\beta(k_1, k_2) - J_\beta(k_1-1, k_2+1),\ 1 \leq k_1 \leq B-1,\ 1 \leq k_2 \leq B-1$.

**Proof:** Consider a function $f \in \mathcal{F}$ which satisfies properties 1-5 of Lemma 1 (for example the function "0"). In appendix A, we show that the function $Tf$ also satisfies these properties. Using the fact that $\lim_{n\to\infty} T^{(n)} f(x) = J_\beta(x)$ for every $x \in X$, the function $J_\beta(\cdot)$ then satisfies these properties as well. ■

Properties 3-4 are only auxiliary properties needed for the proof of property 5, and the combination of properties 1, 2 and 5 of Lemma 1 allows to directly establish the following proposition.

**Proposition 1** *The optimal policy for the discounted cost problem is of type POT.*

**Proof:** Properties 1-2 of Lemma 1 imply that an arrival from either type should be accepted if an empty buffer is available. From property 5, the optimal value function is concave on the line $k_1 + k_2 = B$ and hence it has either a unique maximum or two maxima at consecutive points on this line. The fact that $J_\beta(x)$ satisfies the dynamic programming equation $J_\beta(x) = TJ_\beta(x)$, implies that the value of $k_1$ (or $k_2$) corresponding to the maximum (in case two maxima exist either one can be chosen) is the optimal threshold in the POT policy. ■

Lemma 1 and Proposition 1 can be shown to hold for the weighted average reward criterion with weights $0 \leq w_1,\ w_2 \leq 1,\ w_1 + w_2 = 1$. In this case, the differences in properties 1 and 2 of Lemma 1 should be upper bounded by $w_1$ and $w_2$ instead of 1, respectively, and the proofs follow as in appendix A.

### 3.1.2 The Two-Ported System with Equal Transmission Rates ($\mu_1 = \mu_2$)

In this section we establish an interesting albeit intuitive property of the optimal policy for the two-ported system when the transmission rate on both output ports are equal, $\mu_1 = \mu_2 \stackrel{\Delta}{=} \mu$. For that system, we show that when $\lambda_1 \geq \lambda_2$, the optimal threshold satisfies, $k_1^* \leq B/2$. The proof of this property relies again on the value iteration method.

The dynamic programming operator for this system is now:

$$\begin{aligned} Tf(k_1, k_2) &= \mu[\ 1\{k_1 > 0\} + 1\{k_2 > 0\}\ ] \\ &+ \beta\mu[\ f((k_1 - 1)^+, k_2) + f(k_1, (k_2 - 1)^+)\ ] \\ &+ \beta\lambda_1\Phi_1(k_1, k_2) + \beta\lambda_2\Phi_2(k_1, k_2) \end{aligned} \tag{10}$$

where, $\Phi_i(\cdot)$, $i = 1, 2$, were defined in (8)-(9).

**Lemma 2** *If $\lambda_1 \geq \lambda_2$, the optimal value function $J_\beta(k_1, k_2)$ has the following property:*

$$J_\beta(k_1, k_2) \geq J_\beta(k_2, k_1), \qquad k_2 \geq k_1$$

**Proof:** See appendix A. ■

The above lemma essentially states that when $\lambda_1 \geq \lambda_2$, it is preferable to have more packets of type 2 than of type 1 in the system. When combined with Proposition 1, it directly gives the desired result stated in the following Proposition.

**Proposition 2** *If $\lambda_1 \geq \lambda_2$, the optimal policy for the discounted cost is POT with threshold $k_1^* \leq B/2$* .

**Proof:** From Proposition 1, the optimal policy for the discounted cost problem is POT. From Lemma 2, we know that

$$J_\beta(k_1, B - k_1) \geq J_\beta(B - k_1, k_1), \qquad k_1 \leq \frac{B}{2}$$

which, together with the concavity of the optimal value function $J_\beta(k_1, k_2)$ on the line $k_1 + k_2 = B$ (property 5 of Lemma 1), implies that the maximum of the optimal value function on this line occurs in the range $k_1 \leq B/2$. ■

This result confirms the intuition that the higher arrival rate of type 1 packets implies that they can be pushed out more often.

### 3.1.3 Threshold Design for the Two-Ported System

The two previous sections established that the optimal policy for the two-ported system is of type POT. However, the value of the optimal threshold was not explicitly identified. There are a number of possible approaches to obtain the value of the optimal threshold, but typically they are computationally intensive and for large values of the buffer size may even be infeasible. In this section we propose a heuristic approximation to compute the value of the optimal threshold $k_1^*$. The accuracy of this approximation will be assessed numerically in section 4.2.

Using the value iteration method, it is easy to check that an upper bound to the solution of the equation $J_\beta(k_1, k_2) = TJ_\beta(k_1, k_2)$ is the function

$$f_\beta(k_1, k_2) = d - c_1\alpha_1^{k_1} - c_2\alpha_2^{k_2},$$

where $d = (\mu_1 + \mu_2)/(1-\beta)$, and $\alpha_i,\ i = 1, 2$, is the unique solution in $(0, 1)$ of the quadratic equation

$$(1 - \beta + \mu_i\beta + \lambda_i\beta)\alpha_i = \lambda_i\beta\alpha_i^2 + \mu_i\beta \tag{11}$$

and

$$c_i = \frac{\mu_i}{1 - \beta(1 - \lambda_i(1 - \alpha_i))} \tag{12}$$

In fact, the function $f_\beta(k_1, k_2)$ corresponds to the discounted long term throughput when the buffer size is infinite. We are interested in determining the value $k_\beta^*$ that maximizes $J_\beta(k, B - k)$ for $k = 0, \ldots, B$. Our proposed approximation consists of assuming that $k_\beta^*$ approximately maximizes $f_\beta(k, B - k),\ k = 0, \ldots, B$ as well. We expect this to be especially true for large buffers since the optimal cost is then closer to $f_\beta(\cdot)$.

The maximum of $f_\beta(k, B - k),\ \ k = 0, \ldots, B$ is easily found to be either $\max(\lfloor x^* \rfloor, 0)$ or $\max(\lfloor x^* \rfloor + 1, 0)$, where

$$x^* = \frac{\ln(c_2 \ln \alpha_2/(c_1 \ln \alpha_1))}{\ln \alpha_1 + \ln \alpha_2} + \frac{\ln \alpha_2}{\ln \alpha_1 + \ln \alpha_2} B = C_1 + C_2 B \tag{13}$$

To find the form of the approximation for the average cost criterion (see section 3.3 for details), we have to take the limit as $\beta \to 1$. Let $\rho_i \stackrel{\Delta}{=} \lambda_i/\mu_i,\ i = 1, 2$. It can be seen that

$$\lim_{\beta\to 1} \alpha_i = \begin{cases} 1 & \text{if} \quad \rho_i \le 1 \\ \rho_i^{-1} & \text{if} \quad \rho_i > 1 \end{cases}$$

and

$$\lim_{\beta\to 1} c_i = \begin{cases} \infty & \text{if} \quad \rho_i \le 1 \\ \frac{\mu_i}{\lambda_i - \mu_i} & \text{if} \quad \rho_i > 1 \end{cases}$$

Also, when $\rho_i \le 1$,

$$\alpha_i'(1) \stackrel{\Delta}{=} \lim_{\beta\to 1} \frac{d\alpha_i}{d\beta} = \frac{1}{\mu_i - \lambda_i}$$

We can, therefore, compute (except for the case $\rho_1 = \rho_2 = 1$ which requires that we apply L'Hospital's rule one more time)

$$\bar{C}_2 = \lim_{\beta \to 1} C_2 = \lim_{\beta \to 1} \frac{\ln \alpha_2}{\ln \alpha_1 + \ln \alpha_2} = \begin{cases} 1 & \text{if} \quad \rho_1 \leq 1,\ \rho_2 > 1 \\ \frac{\ln \rho_2}{\ln \rho_1 + \ln \rho_2} & \text{if} \quad \rho_1 > 1,\ \rho_2 > 1 \\ \frac{\mu_1 - \lambda_1}{\mu_1 - \lambda_1 + \mu_2 - \lambda_2} & \text{if} \quad \rho_1 < 1,\ \rho_2 < 1 \\ 0 & \text{if} \quad \rho_1 > 1,\ \rho_2 \leq 1 \end{cases}$$

and similarly for $\rho_1 \leq 1,\ \rho_2 \leq 1$ (other cases are again straightforward),

$$\begin{aligned} \bar{C}_1 = \lim_{\beta \to 1} C_1 &= \lim_{\beta \to 1} \frac{d \ln(c_2 \ln \alpha_2)/d\beta - d \ln(c_1 \ln \alpha_1)/d\beta}{\alpha_1'(1) + \alpha_2'(1)} \\ &= \frac{(\rho_2 - 1)\alpha_2''(1) + (\alpha_2'(1))^2 + 2\rho_2 \alpha_2'(1)}{\alpha_2'(1)(\alpha_1'(1) + \alpha_2'(1))} \\ &- \frac{(\rho_1 - 1)\alpha_1''(1) + (\alpha_1'(1))^2 + 2\rho_1 \alpha_1'(1)}{\alpha_1'(1)(\alpha_1'(1) + \alpha_2'(1))} \end{aligned} \tag{14}$$

where

$$\alpha_i''(1) \stackrel{\Delta}{=} \lim_{\beta \to 1} \frac{d^2 \alpha_i}{d\beta^2} = 2\alpha_i'(1)[\lambda_i(\alpha_i'(1))^2 + \alpha_i'(1) - 1]$$

The proposed approximation for the optimal threshold $k_1^*$ consists then of selecting the closest integer to $x^* \stackrel{\Delta}{=} \bar{C}_1 + \bar{C}_2 B$. Note that the main contribution for large $B$ comes from $\overline{C}_2$. In fact, as will be seen in section 4.2, numerical evidences suggest the following conjecture which, however, we were not able to prove

$$\lim_{B \to \infty} \frac{k_1^*}{B} = \bar{C}_2. \tag{15}$$

### 3.2 The Symmetric $N$-Ported System

In this section, we consider a system with $N$ identical output ports, i.e., the arrival and transmission rates are the same on all ports and denoted by $\lambda$ and $\mu$, respectively. For this special case, we show that the optimal policy is to accept all packets whenever the buffer is non-full, and when the buffer is full to accept a packet only if the queue corresponding to its destination output port is not the largest. In the latter case, a packet from the largest queue is pushed-out to accommodate the new arrival.

The dynamic programming equation for this system becomes:

$$Tf(\overline{k}) = \mu \sum_{i=1}^{N} 1\{k_i > 0\} + \beta\mu \sum_{i=1}^{N} f(\ (\overline{k} - e_i)^+\ ) + \beta\lambda \sum_{i=1}^{N} \Phi_i(\bar{k}) \tag{16}$$

where,

$$\begin{aligned} \Phi_i(\bar{k}) &= \max_{1\le j\le N} \{\ f(\overline{k} + e_i - e_j)\ \} \quad \text{if} \ \sum_{i=1}^{N} k_i = B \\ \Phi_i(\bar{k}) &= \max_{1\le j\le N} \{\ f(\overline{k} + e_i), f(\overline{k} + e_i - e_j)\ \} \quad \text{if} \ \sum_{i=1}^{N} k_i < B \end{aligned} \tag{17}$$

where, we define $\overline{k} \triangleq (k_1, \ldots, k_N)$, $(\overline{k} - e_i)^+ \triangleq (k_1, \ldots, (k_i - 1)^+, \ldots, k_N)$, and $\Phi_i(\overline{k}) \triangleq 0$ if $k_j < 0$ or $k_j > B$ for some $1 \le j \le N$.

As for the two-ported case, we first proceed to establish a number of key properties of the optimal value function which will enable us to characterize the optimal policy.

**Lemma 3** *The optimal value function $J_\beta(\overline{k})$ has the following properties:*

1. *Monotonicity and boundedness in $k_i$: $0 \le J_\beta(\overline{k} + e_i) - J_\beta(\overline{k}) \le 1$, $0 \le k_i \le B - 1$ , $1 \le i \le N$.*

2. *Symmetry: $J_\beta(\overline{k}) = J_\beta(\pi(\overline{k}))$ for any permutation $\pi(\overline{k})$ of the vector $\overline{k}$.*

3. *Balancing: For $1 \le i, j \le N$, if $k_i \ge k_j$ then $J_\beta(\overline{k}) \ge J_\beta(\overline{k} + e_i - e_j)$, otherwise $J_\beta(\overline{k}) \le J_\beta(\overline{k} + e_i - e_j)$.*

4. *Drop from the longest queue: For $1 \le i, j, l \le N$, if $k_i < k_j < k_l$ then $J_\beta(\overline{k} + e_i - e_l) \ge J_\beta(\overline{k} + e_i - e_j)$.*

**Proof:** See appendix A. ■

As before, the above properties can now be applied to characterize the optimal policy which we state in the following proposition.

**Proposition 3** *The optimal policy for the discounted cost problem is to accept a packet whenever the buffer is non-full. When the buffer is full, a packet is accepted only if the queue corresponding to its destination output port is not the largest among all queues. The arriving packet is then accommodated by pushing-out a packet from the largest queue.*

**Proof:** Property 1 of Lemma 3 implies that an arriving packet should be accepted if an empty buffer is available. Property 3 implies that an arrival to a full buffer should be accepted only if it is not destined to the longest output queue; in this case property 4 implies that a packet from the longest output queue should be dropped in order to accommodate the arriving packet. ■

### 3.3 The Average Cost Problem

In this section we establish that the usual conditions required to extend the solution of the discounted cost problem to the average cost problem are indeed satisfied. The state space of our system is finite, and for $\mu_i > 0,\ 1 \leq i \leq N$, the state $x = 0$ is accessible from every other state regardless of which stationary policy is used. Hence, the conditions of Corollary 2.5, section V, of [8] are satisfied and the following properties hold:

1. There exists a bounded function $h(i)$ and a constant $J$ (the optimal value of the average cost problem which doesn't depend on the initial state) which satisfy the average cost version of the optimality equation,

$$J + h(\bar{k}) = \sum_{i=1}^{N} \mu_i \mathbf{1}\{k_i > 0\} + \sum_{i=1}^{N} \mu_i h((\bar{k} - e_i)^+) + \sum_{i=1}^{N} \lambda_i \Phi_i(\bar{k}), \tag{18}$$

   where $\Phi_i(\bar{k})$ is defined in (17) with $h(\cdot)$ replacing $f(\cdot)$.

2. For some sequence $\beta_n \to 1,\ h(\bar{k}) = \lim_{n\to\infty}[J_{\beta_n}(\bar{k}) - J_{\beta_n}(0)]$.

3. $J = \lim_{\beta\to 1}(1-\beta)J_\beta(0)$.

From property 2 it follows that the function $h(\bar{k})$ has the same properties as the function $J_{\beta_n}(\bar{k})$ listed in Lemmas 1-3 for some sequence $\beta_n \to 1$. Equation (18) has a form similar to the dynamic programming equation of the discounted cost problem and Corollaries 1-3 can, therefore, be shown to also hold for the average cost problem.

## 4 Numerical Results

In this section we investigate the performance of a shared memory switch operated under the optimal policy identified in this paper. Our focus is on the two-ported case for which more general results are available. For this system, we conduct two kinds of investigations: A comparison with the performance of the optimal coordinate-convex policy; An evaluation of the accuracy of the heuristic approximation proposed to compute the optimal threshold.

In order to determine the performance of the optimal policy, we need to compute the packet loss probabilities of the system under this policy. The main problem is obtaining these probabilities is that the state space of the underlying Markov chain increases very fast as the buffer size increases. Specifically, the number of states is $(B+1)^2/2$, where $B$ is the buffer size, which makes direct state probabilities computations difficult when $B$ is large.

We investigated two possible approaches to compute packet loss probabilities. The first one takes advantage of the special structure of the Markov chain to reduce to $B+1$ the number of equations needed to solve for the state probabilities. The method for achieving this reduction is presented in appendix B. While this method was found to drastically improve computation time, it requires very high precision in order to avoid numerical instabilities. Therefore, we investigated another approach for computing loss probabilities. It is based on the successive approximation technique for computing average costs presented in [1, Section 7.2]. This technique requires certain properties from the Markov chain, but they are easily seen to be satisfied in our case. Since we are interested mainly in computing loss probabilities rather than the probabilities of every state, we can then use the method with the number of lost packets as our cost function. The main issue with the technique is convergence speed, and it is typically much slower than the first approach. However, the main advantage is numerical stability, especially for large buffers. Because of that, it has been the technique of choice in most of our numerical examples.

### 4.1 Comparison with Coordinate-Convex Policies

Since a coordinate-convex policy is a special case of a pushout policy, considering as cost the overall switch loss probability, the optimal pushout policy will have a smaller loss probability than the optimal coordinate convex policy. The price of this improvement is, however, a more complex implementation, and it is of interest to evaluate the trade-off between the gain in performance and the higher cost. In this section, we provide numerical results to help us compare the relative performance of the optimal coordinate-convex, $\pi_c^*$, and the optimal pushout, $\pi_p^*$ policies. Figure 1 gives the performance of the two policies for a buffer size of 50. In figures 1 (a) and (b), the utilization ($\rho = \lambda/\mu$) of port 1 is kept constant at .6 while the utilization of port 2 varies from .6 to 1.9. In figures 1 (c)and (d), the utilization of port 1 is kept constant at .8 while the utilization of port 2 varies again between .6 and 1.9.

We observe that the difference in overall loss probability between the two policies is not very significant. For the specific examples we consider, the maximum value of the ratio

$$\frac{\text{Loss probability of } \pi_c^*}{\text{Loss probability of } \pi_p^*}$$

is 1.18. However, when individual loss probabilities are considered, significant differences between the two policies are observed. Specifically, the loss probability of port 1 whose utilization is kept constant, is affected significantly by the variation of the utilization of port 2 under policy $\pi_c^*$ and varies by up to seven orders of magnitude. Under policy $\pi_p^*$, however, the loss probability of port 1 never increases above what it experiences in the reference balanced case (equal port loads) by more than one order of magnitude as the utilization of port 2 varies. Furthermore, the better overall performance of $\pi_p^*$ often also results in lower loss probabilities for both ports. Even in the few cases where the loss probabilitiy for the heavier loaded port is lower under $\pi_c^*$ than under $\pi_p^*$, the difference remains minimal. Specifically, for the examples of figure 1, the loss probabilitiy for the heavier loaded port under $\pi_c^*$ is never less than 65% of its value under $\pi_p^*$. Similar experiments were conducted for many other values of port utilizations and for buffer sizes up to 80, and similar behaviors were observed. Therefore, we concluded that an important advantage of $\pi_p^*$ compared to $\pi_c^*$ is that it effectively isolates the performance of a port with constant utilization from fluctuations in the utilization of the other port (assuming, the optimal policy is used in each case).

Figure 2 illustrates another noteworthy feature of the pushout policies. In this figure we plot the total loss probability and the loss probabilities of the two ports when a POT policy, not necessarily the optimal, is employed. The buffer size in this case is 30 and the $k_1$ thresholds used by the POT policy are varied between 0 and 30. We observe that the overall loss varies little as the threshold changes. This implies that the performance of the optimal policy is not critically dependent on the choice of the optimal threshold. However, this is not true for individual loss probabilities at each port. As we have seen above it may be desirable, in addition to achieving a throughput close to the optimal, to also provide appropriate loss probabilities to each port. In this case a good approximation to the optimal threshold $k_1^*$ is still required. As we will see in the next subsection, the approximation proposed in section 3.1.3 for the optimal threshold is quite good, especially when the port utilizations are smaller than 1.

## 4.2 Accuracy of the Approximation

Figure 3 compares the values of the optimal threshold and the proposed approximate threshold for the parameters of figure 1.

We see that the approximate values are very close to the optimal when utilization of both ports is smaller than one. Numerical experimentation has also shown, that the approximation while somewhat less accurate, is still good when the utilization of both ports is larger than one. The approximation is less accurate when the utilization of one port is smaller than but close to 1, while the utilization of the other port is larger than but close to 1. Even in this case, as the buffer size increases, the approximation does become more accurate. This is illustrated in figure 4, where we plotted the ratio of the optimal to the approximate threshold as the buffer size increases. This was done for two sets of port utilizations. In the first set $(\rho_1, \rho_2) = (.8, 1.1)$, while in the second, $(\rho_1, \rho_2) = (.9, 1.1)$. We see that in both cases, while the approximation is poor for small buffers, it continually improves as the buffer size increases. This leads us to conjecture the correctness of the asymptotic behavior of the approximation expressed in (15).

# 5 Conclusion

In this paper, we have studied optimal policies for shared memory switches when packets already stored in memory can be pushed-out to accommodate another packet. We have identified the optimal policy for a general two-ported system as well as for a symmetric $n$-ported system. Our investigation of the performance of the optimal policy has shown, that if the only objective is to minimize the overall loss probability, the optimal push-out policy does not provide significant advantages over the optimal coordinate-convex policy. However, if it is also desired that the loss probability of one port be isolated from the traffic fluctuations of the other port, then the optimal push-out policy offers some advantages.

Another interesting property of the optimal POT policy is the insensitivity of the overall loss probability to the exact choice of the threshold, which facilitates its design. In cases where individual port loss probabilities are also of importance, a more accurate calculation of the threshold is required and we suggested an approximation that provides a quick estimate which is good for most traffic values of interest.

While we provided the optimal push-out policy for $N > 2$ and symmetric traffic, the structure of the optimal policy for general input traffic and $N > 2$ is not known at this time.

APPENDICES

# A Proofs of Lemmas

**Proof of Lemma 1:** Assume that properties 1-5 of Lemma 1 hold for a function $f \in \mathcal{F}$.

*Proof of property 1:*

**case 1** ($k_1 + k_2 \leq B - 2$): From properties 1-2 of the function $f$, we have

$$\begin{aligned} Tf(k_1, k_2) &= \mu_1 1\{k_1 > 0\} + \mu_2 1\{k_2 > 0\} \\ &+ \beta\mu_1 f((k_1-1)^+, k_2) + \beta\mu_2 f(k_1, (k_2-1)^+) \\ &+ \beta\lambda_1 f(k_1+1, k_2) + \beta\lambda_2 f(k_1, k_2+1) \end{aligned} \tag{19}$$

$$\begin{aligned} Tf(k_1+1, k_2) &= \mu_1 + \mu_2 1\{k_2 > 0\} \\ &+ \beta\mu_1 f(k_1, k_2) + \beta\mu_2 f(k_1+1, (k_2-1)^+) \\ &+ \beta\lambda_1 f(k_1+2, k_2) + \beta\lambda_2 f(k_1+1, k_2+1) \end{aligned} \tag{20}$$

Comparing (19) and (20) term by term and using properties 1-2 of the function $f$, we have that

$$\mu_1 1\{k_1 = 0\} \leq Tf(k_1+1, k_2) - Tf(k_1, k_2) \leq \mu_1 1\{k_1 = 0\} + \beta(1 - \mu_1 1\{k_1 = 0\})$$

from which we have $0 \leq Tf(k_1+1, k_2) - Tf(k_1, k_2) \leq 1$.

**case 2** ($k_1 + k_2 = B - 1$): From properties 1-2 of the function $f$, we have

$$\begin{aligned} Tf(k_1+1, k_2) &= \mu_1 + \mu_2 1\{k_2 > 0\} + \beta\mu_1 f(k_1, k_2) + \beta\mu_2 f(k_1+1, (k_2-1)^+) \\ &+ \beta\lambda_1 \max\{f(k_1+1, k_2), f(k_1+2, k_2-1)\} \\ &+ \beta\lambda_2 \max\{f(k_1+1, k_2), f(k_1, k_2+1)\} \end{aligned} \tag{21}$$

and $Tf(k_1, k_2)$ is the same as (19). From properties 1-2 of the function $f$, we have $0 \leq f(k_1+2, k_2 - 1) - f(k_1+1, k_2-1) \leq 1$ and $-1 \leq f(k_1+1, k_2-1) - f(k_1+1, k_2) \leq 0$, from which we have

$$0 \leq \max\{f(k_1+1, k_2), f(k_1+2, k_2-1)\} - f(k_1+1, k_2) \leq 1 \tag{22}$$

and similarly we have

$$0 \leq \max\{f(k_1+1, k_2), f(k_1, k_2+1)\} - f(k_1, k_2+1) \leq 1 \tag{23}$$

Comparing (19) and (21) term by term and using properties 1-2 of the function $f$ and (22)-(23), we have $0 \leq Tf(k_1+1, k_2) - Tf(k_1, k_2) \leq 1$.

*Proof of property 2:* similar to the proof of property 1.

*Proof of property 3:* follows directly by comparing $Tf(k_1+1,k_2) - Tf(k_1,k_2)$ with $Tf(k_1,k_2) - Tf(k_1-1,k_2)$ term by term as in the proof of property 1. The proof of property 4 follows in a similar way.

*Proof of property 5:*

**case 1** $(k_1+k_2 \leq B-1)$: From properties 1-2 of the function $f$, we have

$$\begin{aligned} & Tf(k_1+1,k_2-1) - Tf(k_1,k_2) = \mu_2(1\{k_2>1\}-1) \\ + \;& \beta\mu_1(f(k_1,k_2-1) - f(k_1-1,k_2)) + \beta\mu_2(f(k_1+1,(k_2-2)^+) - f(k_1,k_2-1)) \\ + \;& \beta\lambda_1(f(k_1+2,k_2-1) - f(k_1+1,k_2)) + \beta\lambda_2(f(k_1+1,k_2) - f(k_1,k_2+1)) \end{aligned} \tag{24}$$

$$\begin{aligned} & Tf(k_1,k_2) - Tf(k_1-1,k_2+1) = \mu_1(1-1\{k_1>1\}) \\ + \;& \beta\mu_1(f(k_1-1,k_2) - f((k_1-2)^+,k_2+1)) + \beta\mu_2(f(k_1,k_2-1) - f(k_1-1,k_2)) \\ + \;& \beta\lambda_1(f(k_1+1,k_2) - f(k_1,k_2+1)) + \beta\lambda_2(f(k_1,k_2+1) - f(k_1-1,k_2+2)) \end{aligned} \tag{25}$$

For $k_1, k_2 \geq 2$ it follows directly by comparing (24) with (25) term by term and using property 5 of the function $f$ that the LHS of (24) is less or equal to the LHS of (25). For $k_1 = 1, k_2 \geq 2$, comparing (24) with (25) term by term we have that, the terms multiplied by $\beta\mu_2$, $\beta\lambda_1$, $\beta\lambda_2$ in (24) are less equal to the corresponding terms in (25) by property 5. We still have to show that

$$\beta\mu_1(f(1,k_2-1) - f(0,k_2)) \leq \mu_1 + \beta\mu_1(f(0,k_2) - f(0,k_2+1)) \tag{26}$$

From property 4 we have $f(0,k_2+1) - f(0,k_2) \leq f(0,k_2) - f(0,k_2-1)$, and from property 1 we have $f(1,k_2-1) \leq 1 + f(0,k_2-1)$. From the last two inequalities we have $f(1,k_2-1) - f(0,k_2) \leq 1 + f(0,k_2) - f(0,k_2+1)$, from which the inequality (26) follows. The proof is similar for the case $k_1 \geq 2, k_2 = 1$.

**case 2** $(k_1+k_2 = B)$: The inequality $Tf(k_1+1,k_2-1) - Tf(k_1,k_2) \leq Tf(k_1,k_2) - Tf(k_1-1,k_2+1)$ is easily shown to hold by comparing these expressions term by term and using property 5 of the function $f$ on the lines $k_1+k_2 = B-1$ and $k_1+k_2 = B$. For the terms involving $\lambda_1$, $\lambda_2$ the above inequality holds for all cases of the location of the maximum of the function $f$ on the line $k_1+k_2 = B$. ■

**Proof of Lemma 2:** Assume that Lemma 2 holds for a function $f \in \mathcal{F}$.

**case 1** $(k_1+k_2 \leq B-1)$: From properties 1-2 of the function $f$, we have

$$\begin{aligned} Tf(k_1,k_2) &= \mu(1\{k_1>0\} + 1\{k_2>0\}) + \beta\mu(f((k_1-1)^+,k_2) + f(k_1,(k_2-1)^+)) \\ &+ \beta\lambda_1 f(k_1+1,k_2) + \beta\lambda_2 f(k_1,k_2+1) \end{aligned} \tag{27}$$

$$\begin{aligned} Tf(k_2,k_1) &= \mu(1\{k_1>0\}+1\{k_2>0\})+\beta\mu(f((k_2-1)^+,k_1)+f(k_2,(k_1-1)^+) \\ &+ \beta\lambda_1 f(k_2+1,k_1)+\beta\lambda_2 f(k_2,k_1+1) \end{aligned} \tag{28}$$

Comparing (27) with (28) term by term and using Lemma 2 for the function $f$, we have that $Tf(k_1,k_2)\geq Tf(k_2,k_1)$.

**case 2** ($k_1+k_2=B$): Here we have,

$$\begin{aligned} &Tf(k_1,k_2)=\mu(1\{k_1>0\}+1\{k_2>0\})+\beta\mu(f((k_1-1)^+,k_2)+f(k_1,(k_2-1)^+) \\ +\;& \beta\lambda_1\max\{f(k_1,k_2),f(k_1+1,k_2-1)\}+\beta\lambda_2\max\{f(k_1,k_2),f(k_1-1,k_2+1)\} \end{aligned} \tag{29}$$

$$\begin{aligned} &Tf(k_2,k_1)=\mu(1\{k_1>0\}+1\{k_2>0\})+\beta\mu(f((k_2-1)^+,k_1)+f(k_2,(k_1-1)^+) \\ +\;& \beta\lambda_1 f(k_2,k_1)+\beta\lambda_2 f(k_2-1,k_1+1) \end{aligned} \tag{30}$$

where in the last two terms of (30) we used Lemma 2 and property 5 of Lemma 1 (concavity on the line $k_1+k_2=B$) for the function $f$. Applying Lemma 2 twice to the function $f$, we obtain the following two inequalities

$$\begin{aligned} &\max\{f(k_1,k_2),f(k_1+1,k_2-1)\}-f(k_2,k_1) \\ &\geq f(k_1+1,k_2-1)-f(k_1,k_2)\geq f(k_2-1,k_1+1)-\max\{f(k_1,k_2),f(k_1-1,k_2+1)\} \end{aligned} \tag{31}$$

From (31) and the inequality $\lambda_1\geq\lambda_2$ we have $Tf(k_1,k_2)\geq Tf(k_2,k_1)$. ∎

**Proof of Lemma 3:** Assume that properties 1-4 of Lemma 3 hold for a function $f\in\mathcal{F}$.

*Proof of property 1:* Similar to the proof of property 1 of Lemma 1.

*Proof of property 2:* Consider a state $\overline{k}$ and define the state $\overline{k}_{ij}$ as the state $\overline{k}$ with $k_i$ and $k_j$ interchanged. It is enough to show that $Tf(\overline{k})=Tf(\overline{k}_{ij})$. Compare the dymanic programming equations, (16), of $Tf(\overline{k})$ and $Tf(\overline{k}_{ij})$ term by term. Clearly, the first terms are equal. Using property 2 ($f((\overline{k}-e_i)^+)=f((\overline{k}_{ij}-e_j)^+)$) it follows that the second terms are equal as well. Using property 2 again one can easily show that the third terms are also equal.

*Proof of property 3:* Consider a state $\overline{k}$ and assume that the buffer is full, i.e., $\sum_{i=1}^N k_i=B$. Fix some $i,j$ ($i\neq j$), and assume that $k_i\geq k_j$ ($\geq 1$). We show that $Tf(\overline{k})-Tf(\overline{k}+e_i-e_j)\geq 0$. From equation (16) we have that

$$\begin{aligned} &Tf(\overline{k})-Tf(\overline{k}+e_i-e_j) \\ =\;& \mu[1-1\{k_j>1\}]+\beta\mu\sum_{\substack{l=1\\ l\neq i,j}}^{N}[f((\overline{k}-e_l)^+)-f((\overline{k}+e_i-e_j-e_l)^+)] \end{aligned}$$

$$
\begin{aligned}
&+ \quad \beta\mu[f(\overline{k}-e_i) - f(\overline{k}-e_j)] + \beta\mu[f(\overline{k}-e_j) - f((\overline{k}+e_i-2e_j)^+)] \\
&+ \quad \beta\lambda \sum_{l=1}^{N} [\max_{1\le p_1 \le N}\{f(\overline{k}+e_l-e_{p_1})\} - \max_{1\le p_2\le N}\{f(\overline{k}+e_i-e_j+e_l-e_{p_2})\}] \qquad (32)
\end{aligned}
$$

Each of the second and the third terms of (32) is greater than or equal to zero (gez) by properties 2 and 3. For $k_j > 1$ the first term of (32) vanishes and the fourth term is gez by property 3. For $k_j = 1$ the first term equals $\mu$ and for the fourth term we have by property 1 (boundedness) that it is greater or equal to $-\beta\mu$, hence the sum of the first and the fourth terms is greater than zero for $\beta < 1$. This shows that the sum of the first four terms is gez.

Then, we show that the last term of (32) is gez. We show that for each $l$ the term that appears in the sum is gez. Fix $l$ and denote by $p_1^*$ and $p_2^*$ the indices that give the maximum value in the first and the second maximum terms, respectively. We break ties in the maximum terms by choosing $p_1^*$ and $p_2^*$ such that $k_{p_1^*}$ and $k_{p_2^*}$ are the maximal components of the vectors $\overline{k}$ and $\overline{k}+e_i-e_j$, respectively (obviously, this convention doesn't affect the validity of the proof). Next, we consider all cases of the location of the maximum indices $p_1^*$ and $p_2^*$ and show that each term that appears in the sum of the last term of equation (32) is gez. First consider the case $p_1^* = j$. Then by property 4 of the function $f$ it must be that $k_j$ is the maximal component of the vector $\overline{k}$ (there may be other components with the same value), and since $k_i \ge k_j$ we have that $k_i = k_j$. Then, applying property 4 of the function $f$ to the second maximum term we have that $p_2^* = i$, which implies that the two maximum terms are equal. Next, consider the case $p_1^* = i$ (and assume that $k_i > k_j$, otherwise we get the previous case). Then, by property 4 of the function $f$ it must be that $k_i$ is the maximal component of the vector $\overline{k}$, and hence $p_2^*$ is equal to $i$ (by property 4 again). Then, the difference between the two maximum terms is gez for $i = l$ or $i, j \ne l$ by property 3 of the function $f$ and for $i \ne l = j$ by properties 2 and 3 of the function $f$. Finally, we consider the case $p_1^* \ne i, j$ (and assume that $k_{p_1^*} > k_i$, otherwise we get the previous case). Then, by property 4 of the function $f$ it must be that $k_{p_1^*}$ is the maximal component of the vector $\overline{k}$, and hence $p_2^*$ is equal to $p_1^*$ (by property 4). The proof then follows exactly as the previous case.

Consider the case $k_i < k_j$ and define $\hat{k} \stackrel{\Delta}{=} \overline{k} + e_i - e_j$. Then, for $k_j > k_i + 1$ we have $Tf(\hat{k}) \ge Tf(\hat{k} + e_i - e_j)$ by the first part of property 3, and for $k_j = k_i + 1$ we have $Tf(\hat{k}) = Tf(\hat{k} + e_i - e_j)$ by property 2. This completes the proof of property 3 for the case of full buffer. The proof for the case of a non-full buffer is simpler and follows in a similar way.

*Proof of property 4:* This property follows directly from property 3. ■

# B Recursive State Probabilities Computations

In this section we outline an efficient method for the computation of the state probabilities in a threshold-based pushout shared memory system with two inputs and two outputs. We let $\lambda_k$ and $\mu_k$, be the arrival and service rates, respectively, for output $k, k = 1, 2$. We assume that the shared memory can hold $B$ packets and denote by $K^*$ the pushout threshold value. Note that a packet occupies its buffer until it finishes being served and leaves the system. In other words, the maximum number of packets in the system is $B$, and the two servers do not correspond to additional buffer space. Under the assumption of Poisson arrivals and exponential service times, the system can be represented by a two-dimensional Markov chain whose state is of the form $(i, j)$, $0 \leq i, j \leq B$, and $i+j \leq B$, with $i$ and $j$ denoting the number of class 1 and class 2 packets in the system, respectively. The total number of states is then $(B+1)(B+2)/2$.

The operation of the system is such that if there are free buffers, an arriving packet is always accepted irrespective of its class. However, when all buffers are full, the handling of a packet depends on its class and the system state:

$i+j = B$ **and** $i < K^*$ : An arriving class 2 packet is dropped and the system state remains at $(i, j)$. An arriving class 1 packet is accepted by pushing out a class 2 packet, and the new state is $(i+1, j-1)$.

$i+j = B$ **and** $i > K^*$ : An arriving class 1 packet is dropped and the system state remains at $(i, j)$. An arriving class 2 packet is accepted by pushing out a class 1 packet, and the new state is $(i-1, j+1)$.

$i+j = B$ **and** $i = K^*$ : Any arriving packet is dropped irrespective of its class, and the system state remains at $(i, j)$.

The $(B+1)(B+2)/2$ state probabilities can be obtained using a number of standard techniques, but the purpose of this section is to describe a simple and efficient recursive procedure for their computation. The key to this approach is to note that all state probabilities can be expressed as functions of the probabilities of boundary states, i.e., states corresponding to a full buffer. In this appendix we describe how these functions can be recursively obtained, and then show how the remaining $B+1$ unknown state probabilities can be computed using additional equations and the normalization condition.

## B.1 Recursive Procedure

### B.1.1 Outline

We denote the probability of state $(i, j)$ by $P_{i,j}$ and want to express all $P_{i,j}$'s in terms of the $B + 1$ unknowns $C_i = P_{i,B-i}$. This can be done recursively using the balance equations for state $(i, j)$ and progressing by decreasing value of $j$ and then increasing value of $i$. When referring to figure 5, we start with the balance equation for state $(0, B)$ which allows us to express $P_{0,B-1}$ as a function of $C_0$. Next we move to $(0, B-1)$ whose balance equations gives $P_{0,B-2}$ in terms of $P_{0,B}$, $P_{1,B-1}$, and $P_{0,B-1}$, and therefore as a function of $C_0$ and $C_1$ based on the previously obtained expression for $P_{0,B-1}$. This process continues row-wise and by decreasing value of $j$ until we reach the second row of the state diagram (the detailed equations for this procedure are provided below). At this point, all state probabilities have been expressed as functions of the unknowns $C_k$, $0 \leq k \leq B$, and it now remains to identify $B + 1$ additional, independent equations from which these unknowns can be computed.

These $B + 1$ additional, independent equations can be obtained from the balance equations for the states $(1, 0), (2, 0) \ldots (B, 0)$ of the last row of the state diagram, plus the normalization condition. The unknowns can then be determined and used to obtain all the state probabilities. The overall complexity of the procedure is $O(B^3)$, while a standard approach based on the full transition matrix would typically be $O(B^6)$. Significant speed improvements can, therefore, be achieved. However, as mentioned earlier, stable operation for large values of $B$ requires very high precision due to the presence of alternating signs in the expressions involved. The rest of this section is devoted to detailing these equations.

### B.1.2 Notation and Detailed Equations

We assume that the state probabilities $P_{i,j}$ can be expressed in terms of the unknowns $C_k$:

$$P_{i,j} = \sum_{k=0}^{B} \alpha_{i,j}(k) C_k, \quad 0 \leq i \leq B \text{ and } 0 \leq j \tag{33}$$

From our definition of the unknowns $C_i = P_{i,B-i}$, we have:

$$\alpha_{i,B-i}(k) = \begin{cases} 1 & \text{if} \quad k = i \\ 0 & \text{if} \quad k \neq i \end{cases}$$

Our recursion to obtain the $P_{i,j}$ in the form of equation (33) proceeds row-by-row and increasing value of the index $i$ identifying the number of class 1 packets in the full buffer state of that row, i.e., the state $(i, B-i)$. This recursive approach allows us to compute the coefficients $\alpha_{i,j}$ in terms of previously computed ones. For that purpose, we identify several *regions* corresponding to different forms of the balance equations, that reflect the influence of various boundary conditions.

As shown in figure 5, there are six different cases that need to be considered as their balance equations have different expressions.

1. State $(0, B)$. This is the only state on row 0, and it corresponds to the case where the system is full with class 2 packets.
2. States $(0, B-i)$, $1 \leq i \leq B-1$. These states are the first ones on each row and correspond to cases with no class 1 packets in the system.
3. States $(i, B-i)$, $1 \leq i < K^*$. These states correspond to a full system where the number of class 1 packets is below the threshold.
4. States $(K^*, B-K^*)$. This state corresponds to a full system where the number of class 1 packets is at the threshold.
5. States $(i, B-i)$, $K^* < i \leq B-1$. These states correspond to a full system where the number of class 1 packets is above the threshold.
6. States $(j, B-i)$, $2 \leq i \leq B-1$, and $1 \leq j \leq i-1$. These are essentially the states in the *interior* of the state diagram.

**Case 1:**

The balance equation for case 1 gives us the following relation:

$$\begin{aligned} C_0(\lambda_1+\mu_2) &= \lambda_2 P_{0,B-1} \\ \Rightarrow P_{0,B-1} &= \frac{\lambda_1+\mu_2}{\lambda_2} C_0 \\ \Rightarrow \alpha_{0,B-1}(0) &= \frac{\lambda_1+\mu_2}{\lambda_2} \quad \text{and} \quad \alpha_{0,B-1}(k)=0, \quad 1 \leq k \leq B \end{aligned} \tag{34}$$

This means that we have obtained $P_{0,B-1}$ in terms of the desired unknowns.

**Case 2:**

Balance equations for case 2 states yield the following expression for $1 \leq i \leq B-1$:

$$\begin{aligned} (\lambda_1+\lambda_2+\mu_2)P_{0,B-i} &= \mu_2 P_{0,B-i+1} + \mu_1 P_{1,B-i} + \lambda_2 P_{0,B-i-1} \\ \Rightarrow P_{0,B-i-1} &= \left(\frac{\lambda_1+\lambda_2+\mu_2}{\lambda_2}\right) P_{0,B-i} - \frac{\mu_2}{\lambda_2} P_{0,B-i+1} - \frac{\mu_1}{\lambda_2} P_{1,B-i} \end{aligned}$$

which gives

$$P_{0,B-i-1} = \sum_{k=0}^{B} \left[ \left( \frac{\lambda_1 + \lambda_2 + \mu_2}{\lambda_2} \right) \alpha_{0,B-i}(k) - \frac{\mu_2}{\lambda_2} \alpha_{0,B-i+1}(k) - \frac{\mu_1}{\lambda_2} \alpha_{1,B-i}(k) \right] C_k$$

This then allows us to express $\alpha_{0,B-i-1}(k)$ in terms of coefficients from states in previous rows. Hence, providing a recursive computational procedure.

$$\alpha_{0,B-i-1}(k) = \left( \frac{\lambda_1 + \lambda_2 + \mu_2}{\lambda_2} \right) \alpha_{0,B-i}(k) - \frac{\mu_2}{\lambda_2} \alpha_{0,B-i+1}(k) - \frac{\mu_1}{\lambda_2} \alpha_{1,B-i}(k) \tag{35}$$

**Case 3:**

The Balance equation for case 3 have the following form for $1 \leq i < K^*$:

$$\begin{aligned} (\lambda_1 + \mu_1 + \mu_2) P_{i,B-i} &= \lambda_1 P_{i-1,B-i} + \lambda_1 P_{i-1,B-i+1} + \lambda_2 P_{i,B-i-1} \\ \Rightarrow P_{i,B-i-1} &= \left( \frac{\lambda_1 + \mu_1 + \mu_2}{\lambda_2} \right) P_{i,B-i} - \frac{\lambda_1}{\lambda_2} P_{i-1,B-i} - \frac{\lambda_1}{\lambda_2} P_{i-1,B-i+1} \end{aligned}$$

This again gives the coefficients $\alpha_{i,B-i-1}(k)$ in terms of coefficients from states in previous rows.

$$\alpha_{i,B-i-1} = \left( \frac{\lambda_1 + \mu_1 + \mu_2}{\lambda_2} \right) \alpha_{i,B-i} - \frac{\lambda_1}{\lambda_2} \alpha_{i-1,B-i} - \frac{\lambda_1}{\lambda_2} \alpha_{i-1,B-i+1} \tag{36}$$

**Case 4:**

The balance equation for the state corresponding to a full system with a number of packets of each class exactly equal to the threshold value gives the following relation:

$$\begin{aligned} (\mu_1 + \mu_2) P_{K^*,B-K^*} &= \lambda_1 P_{K^*-1,B-K^*} + \lambda_1 P_{K^*-1,B-K^*+1} \\ &\quad + \lambda_2 P_{K^*,B-K^*-1} + \lambda_2 P_{K^*+1,B-K^*-1} \\ \Rightarrow P_{K^*,B-K^*-1} &= \frac{\mu_1 + \mu_2}{\lambda_2} P_{K^*,B-K^*} - \frac{\lambda_1}{\lambda_2} P_{K^*-1,B-K^*} \\ &\quad - \frac{\lambda_1}{\lambda_2} P_{K^*-1,B-K^*+1} - P_{K^*+1,B-K^*-1}, \end{aligned}$$

where we have assumed $2 \leq K^* \leq B-2$. Some minor modifications are needed if $K^* < 2$ or $K^* > B-2$, but we shall not detail them here.

Based on the above relation, we again obtain $\alpha_{K^*,B-K^*-1}(k)$ in terms of coefficients from states in previous rows.

$$\alpha_{K^*,B-K^*-1} = \frac{\mu_1+\mu_2}{\lambda_2}\,\alpha_{K^*,B-K^*} - \frac{\lambda_1}{\lambda_2}\,\alpha_{K^*-1,B-K^*} - \frac{\lambda_1}{\lambda_2}\,\alpha_{K^*-1,B-K^*+1} - \alpha_{K^*+1,B-K^*-1}, \tag{37}$$

**Case 5:**

The Balance equations for case 5 have the following form for $K^* < i \leq B-1$:

$$\begin{aligned} (\lambda_2+\mu_1+\mu_2)P_{i,B-i} &= \lambda_1 P_{i-1,B-i} + \lambda_2 P_{i+1,B-i-1} + \lambda_2 P_{i,B-i-1} \\ \Rightarrow P_{i,B-i-1} &= \left(\frac{\lambda_2+\mu_1+\mu_2}{\lambda_2}\right) P_{i,B-i} - \frac{\lambda_1}{\lambda_2} P_{i-1,B-i} - P_{i+1,B-i-1} \end{aligned}$$

The coefficients $\alpha_{i,B-i-1}(k)$ are as for case 3 obtained in terms of coefficients from states in previous rows.

$$\alpha_{i,B-i-1} = \left(\frac{\lambda_2+\mu_1+\mu_2}{\lambda_2}\right)\alpha_{i,B-i} - \frac{\lambda_1}{\lambda_2}\,\alpha_{i-1,B-i} - \alpha_{i+1,B-i-1} \tag{38}$$

**Case 6:**

As mentioned earlier, case 6 corresponds to the balance equations for "typical" states, i.e., states which do not involve any boundary condition. For these states, the balance equation for state $(j, B-i)$ with $2 \leq i \leq B-1$ and $1 \leq j \leq i-1$, is as follows:

$$\begin{aligned} (\lambda_1+\lambda_2+\mu_1+\mu_2)P_{j,B-i} &= \lambda_1 P_{j-1,B-i} + \mu_2 P_{j,B-i+1} + \mu_1 P_{j+1,B-i} + \lambda_2 P_{j,B-i-1} \\ \Rightarrow P_{j,B-i-1} &= \left(\frac{\lambda_1+\lambda_2+\mu_1+\mu_2}{\lambda_2}\right) P_{j,B-i} - \frac{\lambda_1}{\lambda_2} P_{j-1,B-i} \\ &\quad - \frac{\mu_2}{\lambda_2} P_{0,B-i+1} - \frac{\mu_1}{\lambda_2} P_{1,B-i} \end{aligned}$$

This gives the following recursion for the coefficients $\alpha_{j,B-i-1}(k)$:

$$\alpha_{j,B-i-1} = \left(\frac{\lambda_1+\lambda_2+\mu_1+\mu_2}{\lambda_2}\right)\alpha_{j,B-i} - \frac{\lambda_1}{\lambda_2}\,\alpha_{j-1,B-i} - \frac{\mu_2}{\lambda_2}\,\alpha_{0,B-i+1} - \frac{\mu_1}{\lambda_2}\,\alpha_{1,B-i} \tag{39}$$

Using equations (34-39) and proceeding in a row-by-row fashion, all the coefficients $\alpha_{i,j}(k)$ can be computed so that based on equation (33) the $P_{i,j}$'s are expressed in terms of the $B+1$ unknowns $C_k$, $0 \leq k \leq B$. It now remains to obtain $B+1$ additional and independent equations to determine these unknowns.

The first such equation is simply the normalization equation $\sum_{ij} P_{i,j} = 1$, which gives:

$$\sum_{k=0}^{B} \left( \sum_{i,j} \alpha_{i,j}(k) \right) C_k = 1 \tag{40}$$

The remaining $B$ equations are obtained next from balance equations for states in the last row, $(B-i, 0)$, $0 \leq i \leq B-1$, which have not been used before. Note that state $(0,0)$ is not used as it would not contribute an independent equation. As before, we need to distinguish between typical and boundary cases and we start with the latter, i.e., $i = 0$.

The balance equation for state $(B, 0)$ gives:

$$\begin{aligned}
C_B(\lambda_2 + \mu_1) &= \lambda_1 P_{B-1,0} \\
C_B(\lambda_2 + \mu_1) &= \sum_{k=0}^{B} \lambda_1 \alpha_{B-1,0}(k) C_k \\
\Rightarrow 0 &= \sum_{k=0}^{B-1} \lambda_1 \alpha_{B-1,0}(k) C_k + \left[\lambda_1 \alpha_{B-1,0}(B) - (\lambda_2 + \mu_1)\right] C_B
\end{aligned} \tag{41}$$

Similarly, the balance equations for states $(B-i, 0)$, $1 \leq i \leq B-1$ give:

$$(\lambda_1 + \lambda_2 + \mu_1) P_{B-i,0} = \mu_1 P_{B-i+1,0} + \mu_2 P_{B-i,1} + \lambda_1 P_{B-i-1,0}$$

$$\begin{aligned}
0 = \sum_{k=0}^{B} &\left[(\lambda_1 + \lambda_2 + \mu_1)\alpha_{B-j,0}(k) - \mu_1 \alpha_{B-j+1,0}(k)\right. \\
&\left. - \mu_2 \alpha_{B-j,1}(k) - \lambda_1 \alpha_{B-j-1,0}(k)\right] C_k
\end{aligned} \tag{42}$$

Using these $B+1$ equations, the unknowns $C_k$, $0 \leq k \leq B$ can be determined, which allows us to then obtain all the state probabilities.

## References


[1] D. M. Bertsekas, Dynamic Programming, Deterministic and Stochastic Models, Prentice-Hall, 1987.

[2] M. Devault, J. Cochennec and M. Servel, "The Prelude ATD Experiment: Assessments and Future Prospects," *IEEE J. Selected Areas in Communications,* Vol. 6, No. 9, pp. 1528–1537, December 1988.

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

[4] 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.

[5] H. Kroner, G. Hebuterne, P. Boyer and A. Gravey, "Priority Management in ATM Switching Nodes," *IEEE Trans. Commun.,* COM-9(3):418–427, April 1991.

[6] H. Kuwahara, N. Endo, M. Ogino and T. Kozaki, "Shared Buffer Memory Switch for an ATM exchange," *in Proc. Int. Conf. on Communications,* Boston, MA, pp. 4.4.1–4.4.5, June 1989.

[7] S. Lippman, "Applying a New Device in the Optimization of Exponential Queueing Systems," *Operations Reaserch,* Vol. 23, pp. 687–710, 1975.

[8] S. Ross, *Introduction to Stochastic Dynamic Programming,* Academic Press.

[9] R. F. Serfozo, "An Equivalence Between Continous and Discrete Time Markov Decision Processes," *Operations Research*, vol 27, pp 616-620, 1979.

[10] H. Suzuki, *et al.,* "Output-Buffer Switch Architecture for Asynchronous Transfer Mode," *in Proc. Int. Conf. on Communications,* Boston, MA, pp. 4.1.1–4.1.5, June 1989.

[11] F. A. Tobagi, "Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks," *Proceedings of the IEEE,* Vol. 78, No. 1, pp. 133–167, January 1990.

[12] 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.

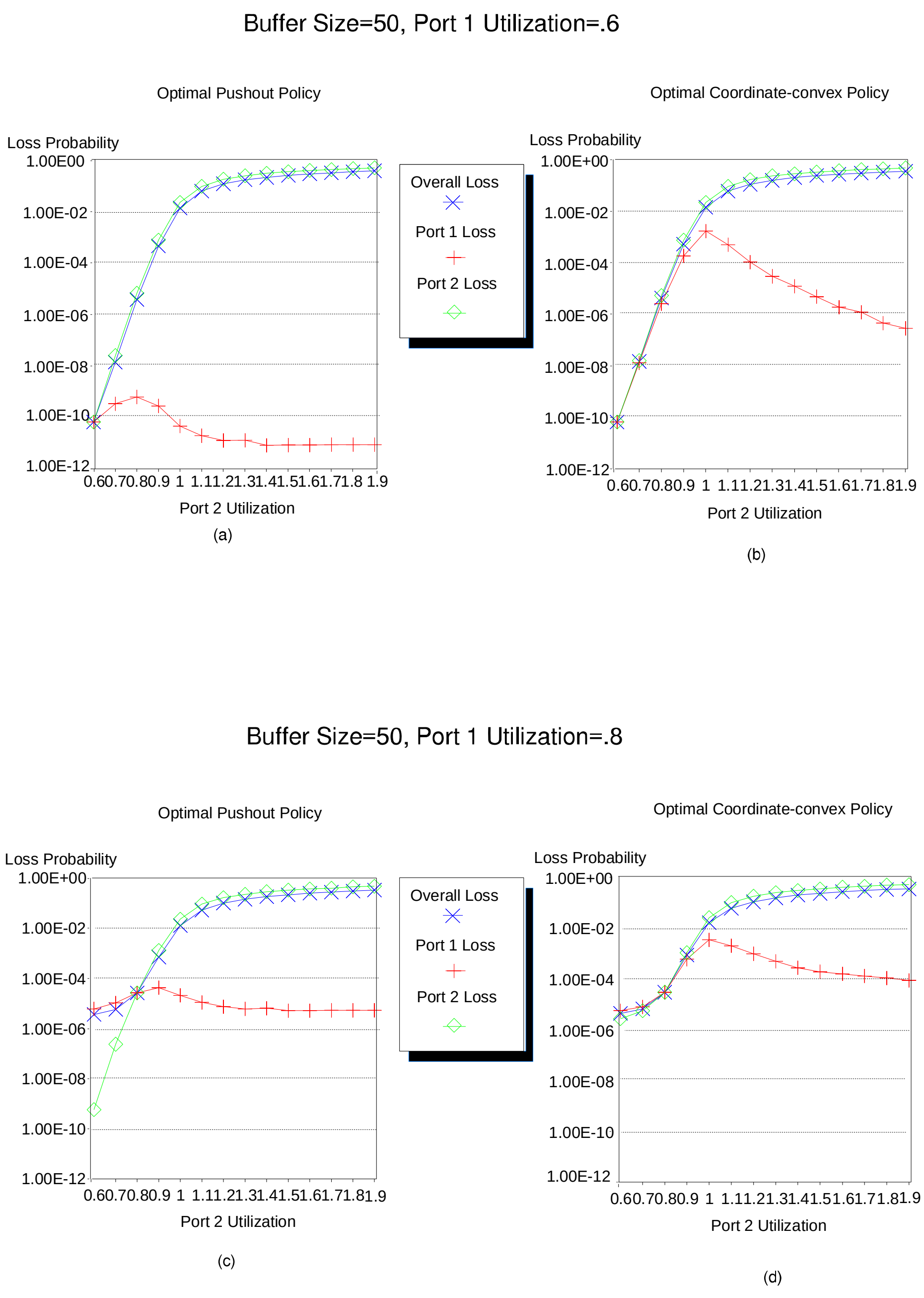



Figure 1: Comparison of Pushout and Coordinate-convex Policies

Buffer Size = 30

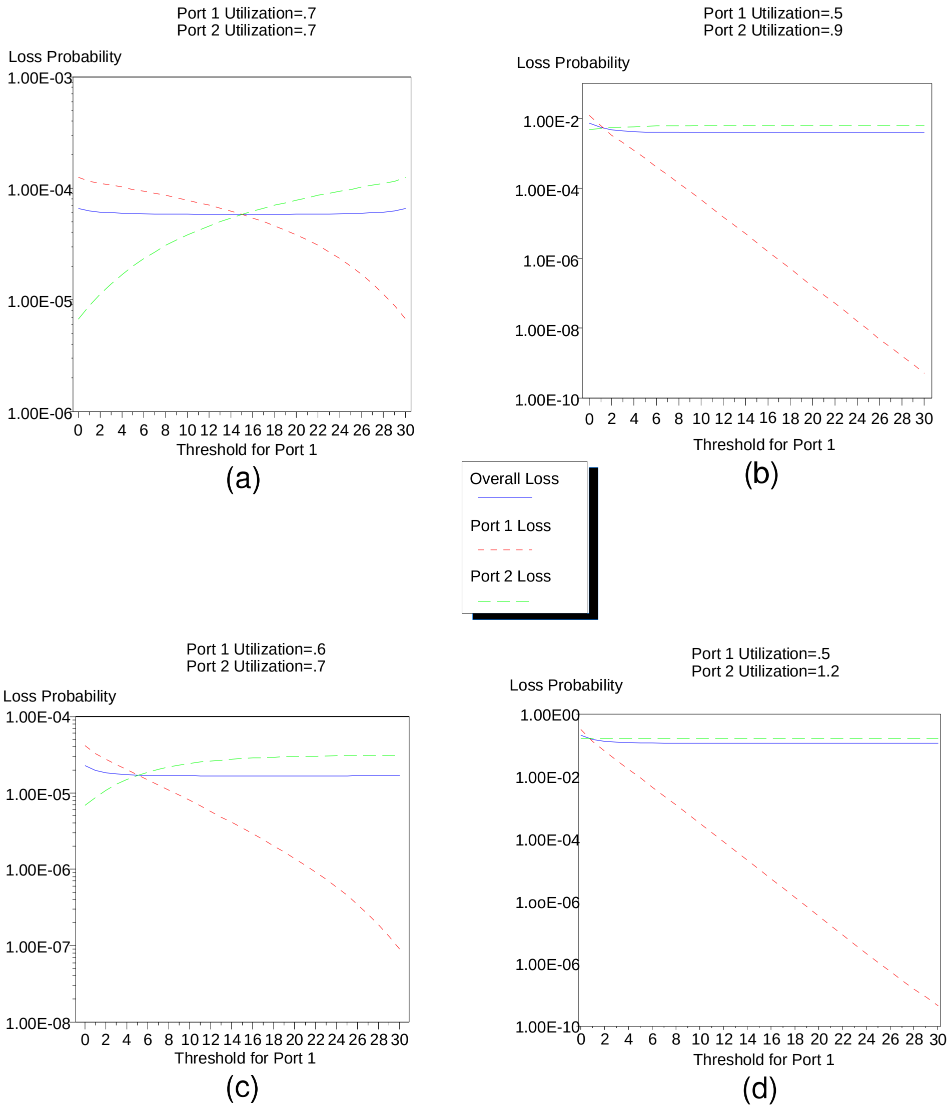
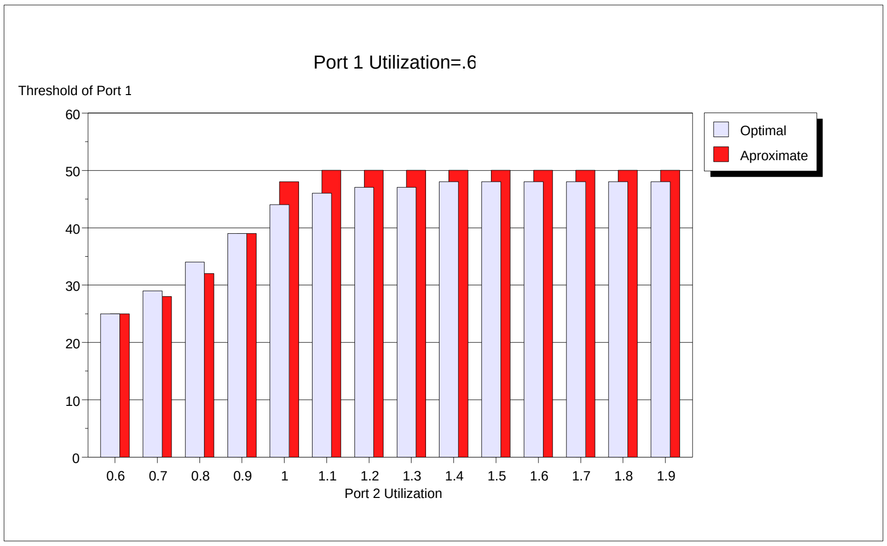
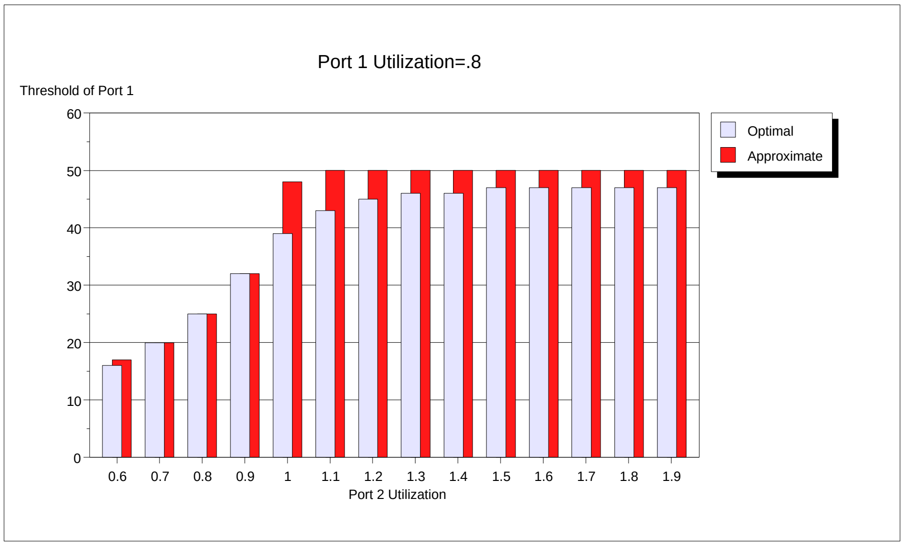



Figure 2: Performance of a POT Policy as the Threshold Varies

Buffer Size = 50




(a)




(b)

Figure 3: Comparison of Approximate and Optimal Threshold

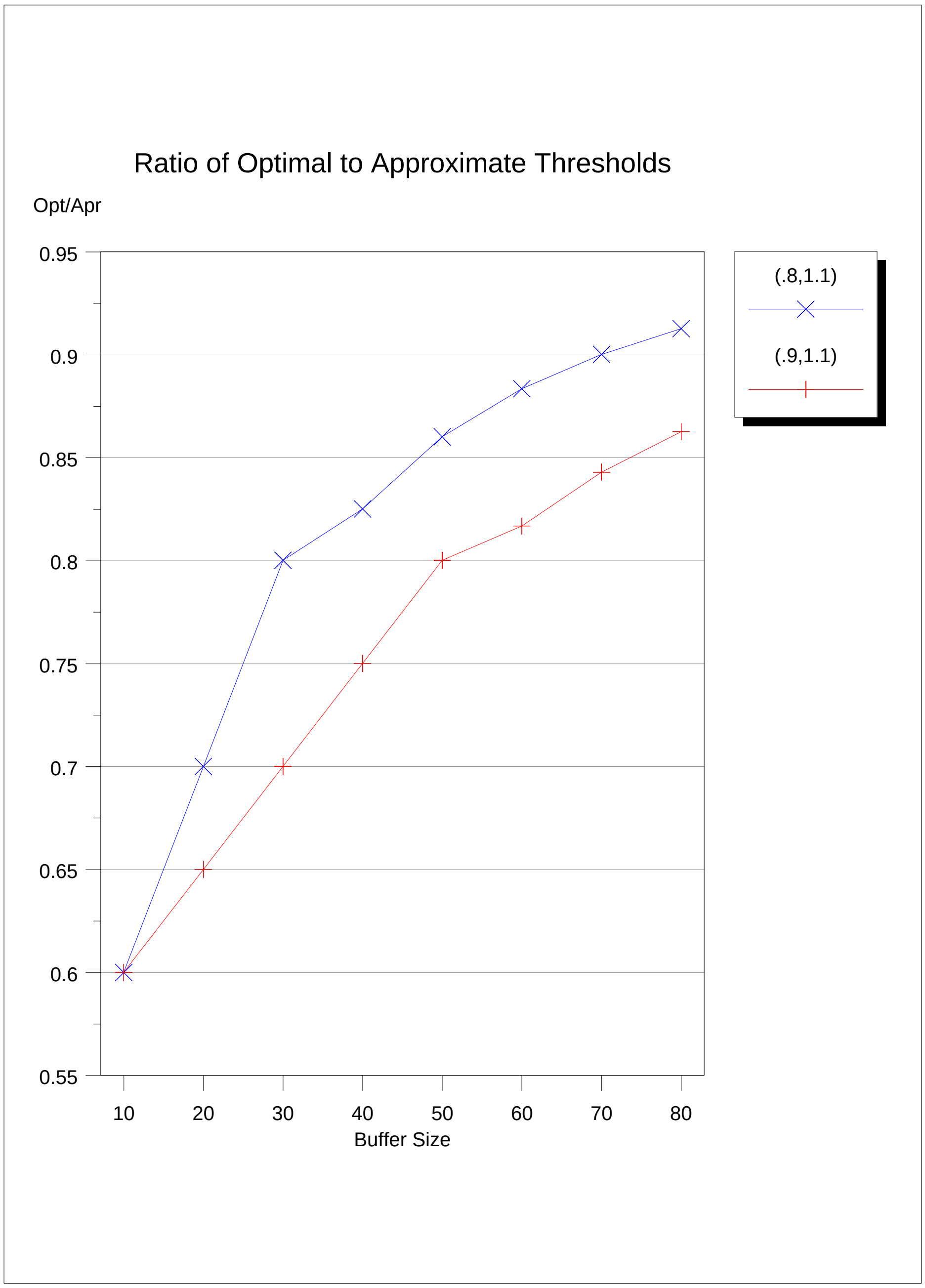



Figure 4: Accuracy of the Approximation as the Buffer Size Increases

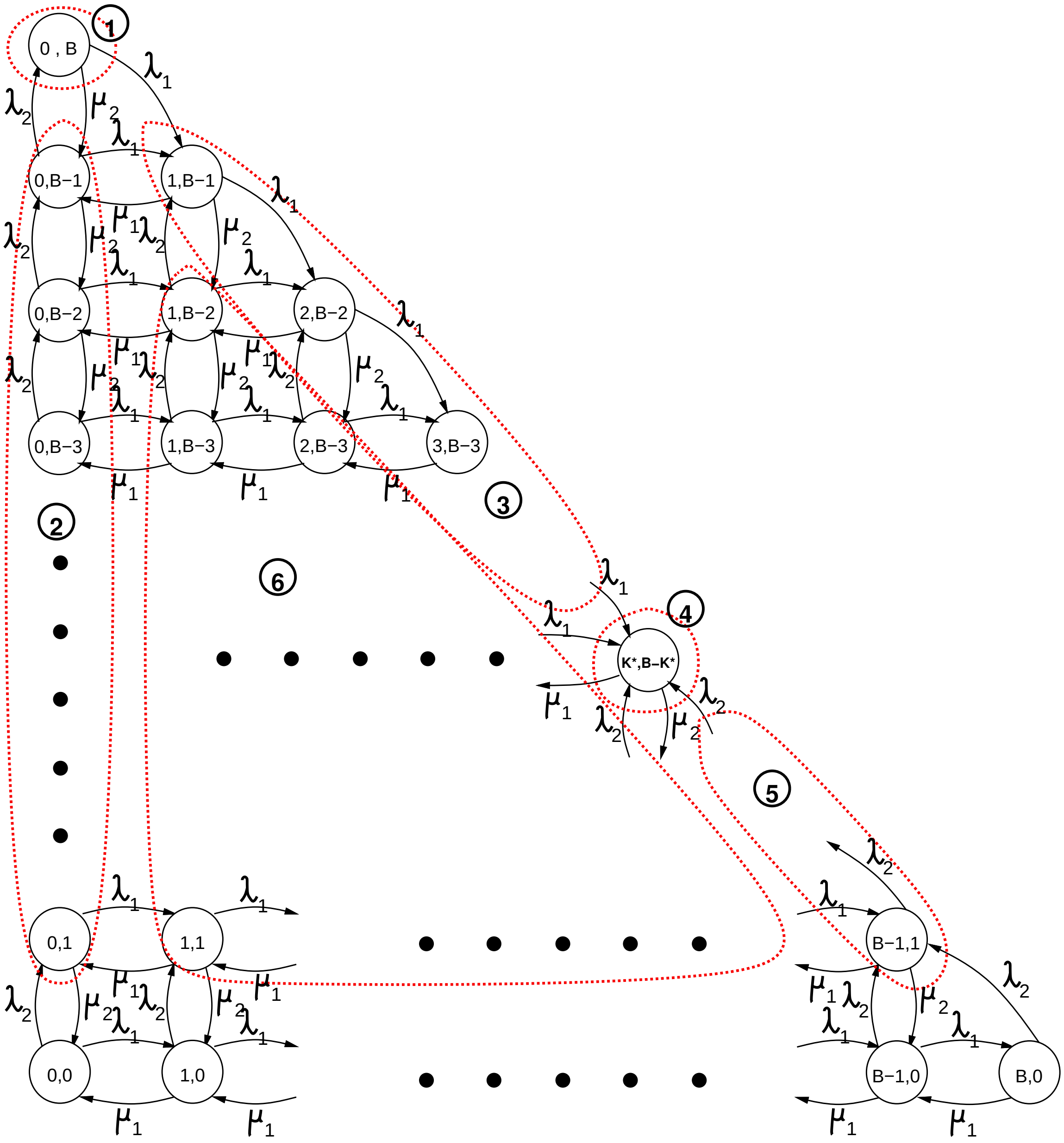



Figure 5: State diagram for $2 \times 2$ system with $B$ buffers and threshold $K^*$