Proceedings of the 41st IEEE
Conference on Decision and Control
Las Vegas, Nevada USA, December 2002

A Jointly Optimum Scheduling and Memory Management for Matching Based Service

Saswati Sarkar*

Abstract

We consider resource allocation in a system which must process and subsequently deliver jobs from N input terminals to N output terminals as per matching constraints. Jobs are stored in a common memory before processing and transfer. The resource allocation objective is to minimize the job loss due to memory overflow. We present optimal scheduling and memory management policies for attaining this objectives for $N = 2$ and symmetric traffic. We identify certain characteristics of the optimal strategy for $N = 2$ and asymmetric traffic, and present a near optimal heuristic for this case. We obtain a heavy traffic lower bound for the optimal discounted cost function for the general case of arbitrary N and arbitrary traffic. We use this lower bound to propose a heuristic strategy whose performance is close to the optimal strategy in this case. The policies proposed in this paper substantially outperform the existing strategy for the system, which was designed and proved to be optimal under the assumption of infinite storage.

1 Introduction

We consider a jointly optimum scheduling and memory management problem for delivery of jobs from N input terminals to N output terminals. The service constraint is that the scheduled jobs must constitute a matching*. The resource management objective is to design the scheduling and the memory management so as to minimize job loss due to memory overflow. This resource allocation problem arises in several communication and computer systems. An example application is a shared memory processor architecture with finite storage. The architecture consists of N processors which need to process and subsequently transfer jobs to N output terminals. Each processor can perform only one type of task. Thus each job must be served by a pre-assigned processor depending on its nature and subsequently transferred to a pre-assigned output terminal. All jobs are stored in a common memory. The transfer of jobs from input to output must follow matching constraints, i.e., one input can transfer only one job at a time, and one output can receive only one job at time. Multiple jobs can be transferred as long as they do not share a common input or output. This transfer constraint arises when the input and output terminals are connected by a crossbar type switching fabric. Refer to Figure 1 for an illustration.

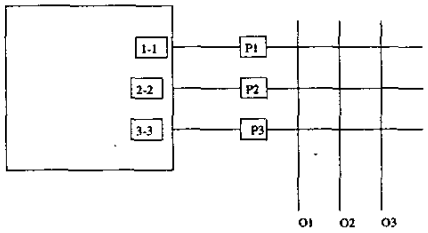

Figure 1: The figure shows a system with 3 input processor (P1, P2, P3) and 3 outputs (O1, O2, O3) connected by a crossbar. A packet $i-j$ is waiting for processing at input i and subsequent transfer to input j. A few example matchings are $[1-1, 2-2, 3-3]$ and $[1-2, 2-1, 3-3]$. Either matching can be scheduled, but the first completes 3 jobs while the second completes only one job as the $1-2$ and $2-1$ queues are empty.

The resource management problem has two components here: (a)job scheduling and (b)memory management. The first decides which jobs can be transferred to the output without violating the matching constraints. The challenge is to efficiently choose among the available matchings (Figure 1). Memory management determines the appropriate course of action when a new job arrives. When a new job arrives at an input, one of the following actions can be taken: (a) the job is accepted (b) the job is rejected and (c) the job is accepted while some other job waiting at the input is dropped. The last action is commonly referred to as "push-out." If a job arrives when the buffer is full, then one of the last two actions must be taken. These decisions will depend on buffer occupancy and possibly arrival and service statistics. More importantly, the job scheduling and memory management decisions must be taken in conjunction, and will depend on each other. For exam-

*S. Sarkar is with the Electrical and Systems Engineering department in the University of Pennsylvania. Email: swati@ee.upenn.edu. Her work was supported in part by NSF grant ANI01-06984.

*In graph-theoretic notions, a matching is a collection of edges which do not share a node. In the switching context a matching is a collection of input output pairs i, j which do not have a common input or output.

0-7803-7516-5/02/$17.00 ©2002 IEEE

ple, only one job can be transferred to the output side if all outstanding tasks are waiting at the same input or are destined to the same output. However, many more jobs can be transferred if the outstanding tasks do not have common inputs and outputs. Refer to figure 2 for an illustration. Memory management can be utilized efficiently to ensure that fewer outstanding jobs share input and output terminals. We discuss this in details later.

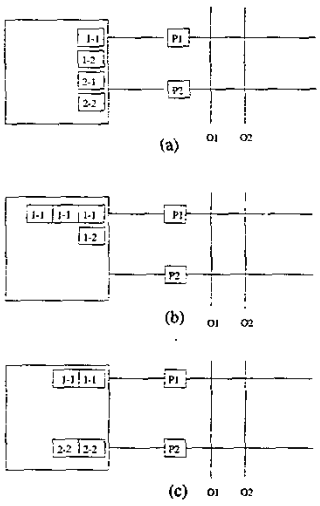

Figure 2: The figure shows a system with $N = 2$ with three different buffer occupancies. The buffer occupancies are $\vec{x} = (1, 1, 1, 1)$ (Figure (a)), $\vec{x} = (3, 1, 0, 0)$ (Figure (b)) and $\vec{x} = (2, 0, 0, 2)$ (Figure (c)).

We first briefly review the existing literature in optimum scheduling and memory management. Tassiulas *et al.*[7] showed that scheduling jobs as per a maximum weighted attains the maximum possible throughput if the storage is infinite. We refer to this scheduling as MM scheduling in the rest of this paper. Memory management problem does not arise under this assumption. Interestingly, the optimal scheduling in presence of finite buffers does not use a maximum weighted matching. Thus the memory constraints introduce the memory management problem and also necessitate a change in the scheduling policy. Towsley *et al.*[8] have considered several routing and scheduling problems in context of N queues sharing a single server. These queues have separate storage. So memory is not a shared resource. The scheduling problem is a N to 1 scheduling while we need a N to N scheduling. Memory management has been addressed for other types of architectures where memory is the only shared resource, and hence these do not consider any scheduling, e.g., [1, 2].

Now we outline the contribution of this paper. *The optimal scheduling and memory management policy is the one which minimizes the average job loss* (note that loss minimization is equivalent to throughput maximization). This can be formulated as a markov decision process(MDP)[5]. However, the generic computation techniques like value iteration and policy iteration do not scale with the buffer size (B) and the number of input and output terminals on account of state space explosion ($O(B^{N^2})$ states). We present a closed form computationally simple optimal scheduling and memory management strategy for the special case of $N = 2$ under the assumption of equal arrival rates for all input-output streams in section 3.1. The optimal scheduling transfers as many jobs as possible at all times. The optimal memory management uses an acceptance policy which balances the number of outstanding jobs of different input-output streams. This in turn allows the scheduling to transfer several jobs to the output and reduces the buffer occupancy and the memory overflow. We obtain several features of the optimal strategy for $N = 2$ and arbitrary arrival rates in section 3.2, and using these properties we design a near optimal heuristic strategy, "pair scheduling minimum difference" (PSMD). The numerical performance evaluation will demonstrate the strict suboptimality of MM. In fact, PSMD reduces the job loss by 30% as compared to MM in many cases. We present a heavy traffic lower bound for the optimal discounted cost function for the general case of architectures with N input processors and N output terminals in section 4. This lower bound and certain properties of the optimal strategy derived from the general markov decision process framework help us design a heuristic, "minimum congestion policy" (MC). We present numerical performance evaluation to demonstrate that MC performs substantially better than MM, and in many cases reduces the job loss by 30% or more. We defer the proofs to technical report [6].

2 System Assumptions

There are N input terminals and N output terminals. Jobs are stored in a common memory of size B until they are processed at the desired input and thereafter transferred to the intended outputs. Arrival processes are independent Poisson. The arrival rate of jobs for input i and output j is λ_{ij}, $i = 1, \ldots, N$, $j = 1, \ldots, N$. Queue length of jobs waiting for input i and output j at time t is $x_{ij}(t)$. Clearly $\sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij}(t) \leq B$. Services have exponential duration with μ_{ij} being the service rate for the jobs for the ith input and jth output. For simplicity we will assume equal service rates, $\mu_{ij} = \mu$ for all i, j. In general, the performance depends on the ratio between the arrival and service rates rather than their absolute values, and as such without loss of generality we assume that $\mu + \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{ij} = 1$, and $\mu > 0$. All jobs which start service at the same time end

service at the same time. Packet arrivals preempt the service (a preemptive service model has been assumed elsewhere as well [8]).

3 A loss optimal scheduling and memory management for $N = 2$

We first describe the optimal resource allocation problem for $N = 2$. Refer to Figure 2 for illustration. There are four different queues of jobs, x_{11}, x_{12}, x_{21} and x_{22} sharing the common memory of size B. The queue x_{ij} stores jobs which must be served by input i and thereafter transferred to output j. The jobs in the pairs x_{11}, x_{22} and x_{12}, x_{21} can be served simultaneously as the queues in these pairs do not share an input or output terminal. The other possibility is to schedule singletons. Job loss can be minimized if the buffer occupancy is reduced. Intuitively this happens if a pair of queues is served together rather than singletons because the former removes two jobs from the buffer while the latter removes only one job from the buffer. However, if the buffer occupancy is such that one queue is empty in each pair then only singletons can be scheduled. For example, if $x_{11} = x_{21} = 0$ then either x_{22} or x_{12} can be served (Figure 2(b)). For further illustration consider the buffer occupancies in Figures 2(a) ($\vec{x} = (1, 1, 1, 1)$) and 2(b)($\vec{x} = (3, 1, 0, 0)$). Observe that the total number of jobs waiting for service is the same for both configurations(4 jobs). Only one job can be served in Figure 2(b) (from x_{11} or x_{12}) as all jobs must be served by the same input (input 1). However, two jobs can be served in Figure 2(a) (can schedule either the pair x_{11}, x_{22} or the pair x_{12}, x_{21} as all the queues are nonempty).

Thus a larger number of jobs can be served and hence job loss can be reduced when the load is balanced across the queues. However, load should be balanced across "appropriate" queues. For example, consider the buffer occupancies in Figures 2(b) ($\vec{x} = (3, 1, 0, 0)$) and 2(c)($\vec{x} = (2, 0, 0, 2)$). Only two queues have jobs in both cases and the total number of jobs waiting for transmission (4 jobs) are the same in both. However, only one job can be served in Figure 2(b) as discussed before, whereas two jobs can be served in Figure 2(c) (from queues x_{11} and x_{22}). Figure 2(c) balances the load between appropriate queues. This can be quantified by considering the difference between the queue lengths of the queues which can be scheduled together, i.e., $|x_{11} - x_{22}|$ and $|x_{12} - x_{21}|$. Note that these differences are $0, 0$ for Figures 2(a) and 2(c) and $3, 1$ for Figure 2(b). In general, a larger number of jobs can be served if these differences are smaller.

Appropriate push-out policy can be used to reduce these differences. Assume that $B = 4$ in Figure 2. Let a job arrive for input 2, for output 1. In Figure 2(b) the differences between the queue lengths of the queues in the pairs remain $3, 1$ if the new job is rejected, while the differences become $2, 0$ if the new job is accepted while pushing out an existing job waiting for input 1, output 1 (in the x_{11} queue). After this replacement, both queues x_{12} and x_{21} can be scheduled serving two jobs simultaneously. In Figure 2(a) the new job should be rejected as its acceptance using push-out will adversely affect the balance and increase the differences from $(0, 0)$ to $(1, 1)$ Summarizing, *a pair of queues should be scheduled whenever possible, and job acceptance/rejection/push-out should be used judiciously to balance the load across the appropriate queues which facilitates a simultaneous service of two jobs whenever possible.* These observations are the key behind designing optimal strategies for symmetric traffic and near-optimal heuristics for asymmetric traffic.

3.1 Equal arrival rates

In this subsection we will consider the symmetric traffic case. We assume that all arrival rates are equal, i.e., $\lambda_{ij} = \lambda$, for all i, j. We present the scheduling and memory management strategies, "PSMD" (**pair scheduling minimum difference acceptance**) which minimize the average job loss in this case.

PSMD Scheduling (pair scheduling): PSMD schedules a pair of queues whenever possible. Let the current state be $\vec{x}$ (let $\vec{x} \neq \vec{0}$). If all the queues are nonempty ($x_{ij} > 0$ for all i, j), schedule either the $1-1, 2-2$ pair (x_{11}, x_{22}) or the $1-2, 2-1$ pair (x_{12}, x_{21}). The choice between the pairs do not affect performance. Now consider the case when some queues are empty. If $\min(x_{11}, x_{22}) > \min(x_{12}, x_{21}) = 0$ schedule the $1-1, 2-2$ pair. If $\min(x_{12}, x_{21}) > \min(x_{11}, x_{22}) = 0$ schedule the $1-2, 2-1$ pair. If $\min(x_{12}, x_{21}) = \min(x_{11}, x_{22}) = 0$, only one queue can be scheduled, and the longest queue is selected.

PSMD Memory Management (minimum difference memory management): PSMD accepts an incoming job without any push out as long as there is available storage. If the buffer is full when a new job arrives, then the job is either rejected or accepted while dropping an existing job from a different queue. The choice between the two is made with the objective of reducing the difference between the queue lengths of the pairs. We introduce some notations for describing this part of the memory management more formally. A "co-operating set" is a pair of queues which can be served together. Co-operating set 1 consists of the pair $1-1$ and $2-2$ and co-operating set 2 consists of the pair $1-2$ and $2-1$. We quantify the differences between the queues in the same co-operating set: $\text{diff}_1(\vec{x}) = |x_{11} - x_{22}|$, $\text{diff}_2(\vec{x}) = |x_{12} - x_{21}|$. Let $\vec{e}_{ij}$ be a vector with a 1 in the $i-j$th position and a 0 elsewhere. Let a job arrive for input i, output j (in queue

$i-j$) and let the storage be full ($\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij} = B$). Then the system considers the state $\vec{y} = \vec{x} + \vec{e}_{ij}$. If $\text{diff}_1(\vec{y}) = \text{diff}_2(\vec{y}) = 0$, the new arrival is rejected. Otherwise, a job is dropped so as to reduce the larger of the two differences under buffer occupancy $\vec{y}$. If the two differences are equal, a job is dropped so as to reduce any one of the differences. If queue $i-j$ is selected for dropping a job, then the new arrival is rejected. The process is as though the new job is accepted and then a job is dropped so as to reduce the larger "diff" function in the resulting state. An illustrative example follows.

Example 3.1: Let $B = 4$. In Figure 2(a) ($\vec{x} = (1,1,1,1)$), PSMD schedules either the pair $1-1, 2-2$ or the pair $1-2, 2-1$ choosing between them arbitrarily. Any incoming job is rejected. In Figure 2(b) ($\vec{x} = (3,1,0,0)$), only one queue can be scheduled. PSMD schedules the longer queue, queue $1-1$. An incoming job for input 1, output 1 is rejected. Any other incoming job is accepted while dropping a job from the $1-1$ queue. In Figure 2(c) ($\vec{x} = (2,0,0,2)$), PSMD schedules the pair $1-1, 2-2$. Any incoming job is rejected. Now let $B = 5$. The scheduling remains the same as before in each case. The memory management decision is to accept any incoming job in all three cases.

Theorem 1 *PSMD minimizes the average job loss for systems with 2 inputs and 2 output terminals ($N = 2$) and equal arrival rates ($\lambda_{ij} = \lambda$ for all i, j).*

3.2 Unequal arrival rates

In this section we no longer assume that $\lambda_{ij} = \lambda$. First, we identify certain properties of the optimal scheduling and memory management strategy, and subsequently design a heuristic using these properties.

Optimal Scheduling: The optimal scheduling schedules a pair of queues whenever possible.
Optimal Memory Management: The optimal decision is to accept an incoming job without any push out as long as the input buffer has space.

These properties specify the strategy in certain cases. For instance, in Figure 2(c) the optimal scheduling is to serve the pair $1-1, 2-2$. Also, if $B = 5$ then the optimal memory management strategy accepts all incoming jobs for all three configurations in Figure 2. However, these properties do not completely specify the optimal strategy. For example, it is not known how to choose between pairs of queues when either pair can be scheduled (e.g., Figure 2(a)), or how to choose a queue when the system state is such that only one queue can be scheduled, i.e., when $\min(x_{12}, x_{21}) = \min(x_{11}, x_{22}) = 0$ (e.g., Figure 2(b)). The optimal job acceptance/rejection/push-out decision is also not known in the case when the input buffer is full (e.g., all three cases in Figure 2 with $B = 4$).

We propose PSMD (Section 3.1) as a heuristic here. We justify the choice as follows. PSMD satisfies the properties obtained for the optimal strategy in this general case. Besides, the scheduling and the memory management decisions of PSMD strive to balance the traffic across different queues which allows the service of larger number of jobs and thus reduce the buffer occupancy and the job loss. Numerical computations will demonstrate that PSMD attains near minimum job loss in this case.

3.3 Performance Evaluation using Numerical Computation

We will present results for the case where the arrival rates are unequal for one pair, and equal for the queues in the other pair. Specifically, $\lambda_{11} = 7\lambda_{22}$, $\lambda_{12} = \lambda_{21}$ and $\lambda_{11} = 3.5\lambda_{12}$. We denote this pattern as "NU1." Figure 3 shows that the performances of PSMD and the optimal are almost identical in most cases with the difference between the loss probabilities less than 1% (3%) in most (all) cases. The maximum difference is 2.8% and this happens for a high value of service rate when the blocking probabilities are less than 3% for both PSMD and the optimal. PSMD decreases the job loss by more than 30% as compared to MM. The trends remain similar for other patterns of arrival rates which we do not exhibit on account of space constraints [6].

The relative performance of MM w.r.t. PSMD is much worse for nonuniform traffic than for uniform traffic (Figure 3). This happens as the queues tend to be heavily dis-balanced when arrival rates are unequal, while the discrepancy is less for equal arrival rates. MM experiences heavier job loss for non-uniform traffic than for uniform traffic on account of the load mismatch in the former case. The memory management in PSMD restores a balance in the queue lengths by replacing jobs of more heavily loaded queues with those of the lightly loaded queues whenever possible, and this retains the performance for nonuniform traffic at the same level as the uniform traffic case. As an example, at service rate 0.2, sum of the arrival rates 0.8, MM has 40.19% and 52.84% job loss for equal arrival rates and arrival pattern NU1 respectively, while the numbers are 39.44% and 40.52% for PSMD. This shows that PSMD is robust to different traffic patterns even without using statistics information in the decision process.

4 Resource allocation for arbitrary N

We consider the scheduling and memory management problems for arbitrary N. The objective of the scheduling policy is to transfer as many jobs as possible so as to minimize the job loss, and memory management decides which jobs to accept, reject and push-out so as to allow the transfer of several jobs in every schedul-

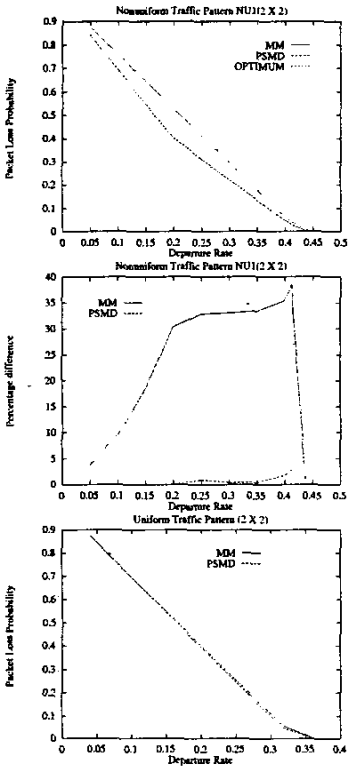

Figure 3: The first two figures compare the performances of PSMD and MM with the optimal strategy for different departure rates (μ) for $N = 2, B = 50$ and the following arrival rate patterns: $\lambda_{11} = 3.5\lambda_{12} = 3.5\lambda_{21} = 7\lambda_{22}, \lambda_{22} = \frac{1-\mu}{15}$. The first figure plots the absolute values of the job loss probabilities ($l_{\mathrm{MM}}, l_{\mathrm{PSMD}}, l_{\mathrm{OPT}}$). The PSMD and the OPTIMAL curves can not be distinguished. The second figure plots the percentage relative difference between the loss probabilities for the optimal and PSMD (curve PSMD) $(100 * (l_{\mathrm{PSMD}}/l_{\mathrm{OPT}} - 1))$, and that between PSMD and MM (curve MM) $(100 * (l_{\mathrm{MM}}/l_{\mathrm{PSMD}} - 1))$. The last figure considers equal arrival rates $\lambda_{ij} = \frac{1-\mu}{4}, \forall i, j$. Note that PSMD is optimal in this case (Theorem 1). The packet loss attained by MM is slightly higher than PSMD.

ing. Consider the example shown in Figure 1. All three jobs can be transferred at the same time since they are waiting at different inputs and intended for different outputs. However, if all three were waiting at the same input, only one can be transferred. Thus like in the special case for $N = 2$ we need to balance the load across appropriate queues. The memory management attains this in PSMD by accepting/rejecting/pushing out jobs so as to reduce the difference between queue lengths of the queues which can be scheduled together($[1-1, 2-2]$ and $[1-2, 2-1]$). For $N > 2$ more than two queues can be scheduled together. For example, $1-1, 2-2, 3-3$ can be scheduled together for $N = 3$ (Figure 1). Thus the notion of reducing these differences can not be extended directly for $N > 2$.

We consider a new measure of congestion, the maximum of the total number of jobs waiting at an input and waiting for an output. We denote this measure as $\mathrm{cong}(\vec{x})$, where $\mathrm{cong}(\vec{x}) = \max(\max_i \sum_{j=1}^{N} x_{ij}, \max_j \sum_{i=1}^{N} x_{ij})$, and $\vec{x}$ is the current state. Note that $\mathrm{cong}(\vec{x}) = 1$ in Figure 1, $\mathrm{cong}(\vec{x}) = 3$ if all three jobs were waiting at the same input, or waiting for the same output in Figure 1. Three jobs can be transferred to the outputs simultaneously in the first case, while only one job can be transferred in the latter cases. Thus intuitively larger number of jobs can be transferred when $\mathrm{cong}(\vec{x})$ is smaller. We propose a strategy which minimizes this congestion measure. First we justify why this is a good measure to minimize. If the current state is $\vec{x}$, every job has duration T units and there is no future arrival, then the minimum time taken to transfer all waiting jobs is $T\mathrm{cong}(\vec{x})$[4]. This indicates that higher the value of $\mathrm{cong}(\vec{x})$, more congested is the switch in some sense. Next, we present lemma 1 which shows that the optimal discounted cost function $J_\beta(\vec{x})$ is lower bounded by $\mathrm{cong}(\vec{x})$ under heavy traffic.

Lemma 1 *Let the total arrival rate at(intended for) each input(output) is greater than the service rate, i.e.,* $\min(\min_i \sum_{j=1}^{N} \lambda_{ij}, \min_j \sum_{i=1}^{N} \lambda_{ij}) > \mu$. *Then for all* $\beta \in [\beta_0 1)$, $J_{\beta,OPT}(\vec{x}) \geq \mathrm{cong}(\vec{x})$ *for all states* $\vec{x}$, *where* $\beta_0 = \frac{B}{B+\min(\min_i \sum_{j=1}^{N} \lambda_{ij}, \min_j \sum_{i=1}^{N} \lambda_{ij})-\mu}$.

Lemma 1 motivates the design of a heuristic approach which minimizes this congestion metric in every step. The scheduling is to serve the matching of largest size which minimizes $\mathrm{cong}(\vec{y})$ of the next state $\vec{y}$. Note that there can be several maximum matchings or matchings of the largest size. We choose a specific maximum matching which minimizes the congestion measure of the next state. Under such a matching, the congestion measure reduces by one whenever the currently scheduled jobs complete transmission. Fast algorithms can be found for computing such matchings using the the-

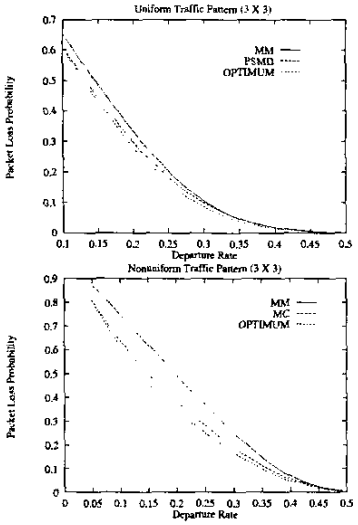

Figure 4: The first figure plots the loss probabilities of MC, MM and the optimal strategy for different departure rates (μ), and equal arrival rates $\lambda_{ij} = \frac{1-\mu}{9}, \forall i,j$ for $N = 3, B = 10$. The second figure considers the following arrival rate patterns: $\lambda_{11} = 9\lambda, \lambda_{ij} = \lambda, (i,j) \neq (1,1)$, $\lambda = \frac{1-\mu}{17}$.

ory of edge coloring of bipartite graphs [3]. The memory management policy accepts jobs without pushing out any existing job as long as the input buffer has space We consider the acceptance decision when an incoming job finds the input buffer full. Let the current state be $\vec{x}$. Suppose the job is for input i and output j. The policy considers the state $\vec{y} = \vec{x} + \vec{e}_{ij}$. Note that this would have been the next state had the new job been accepted. A queue is selected for job drop so as to reduce $\text{cong}(\vec{y})$ by one, i.e., a job is dropped from the most congested terminal for state $\vec{y}$. Overall, the policy minimizes the congestion at both inputs and outputs, and thus we denote it as the "minimum congestion" policy (MC). PSMD is a special case of this policy. We present an example to illustrate MC.

Example 4.1: Let $N = 3$. Let the current state be $\vec{x}$ with $x_{11} = 2, x_{21} = x_{12} = x_{23} = 1$. All other queues are empty. Note that at most 2 jobs can be transferred, and the matchings which transfer two jobs are $1-1, 2-3$ or $1-2, 2-1$ or $1-2, 2-3$. The next states have congestion measures $2, 2, 3$ respectively for these choices. Thus either the first or the second matching can be selected. If $B > 5$ then any new arrival is accepted without any push out. Suppose $B = 5$. A new arrival for input 1 or output 1 is rejected. However, a new arrival for input 3, output 3 is accepted while pushing out a job from queue $1-1$.

We present the numerical performance evaluations for $N = 3$. Figure 4 shows that the job loss of MC is close to that of the optimal for both uniform and nonuniform traffic. MC performs better than the existing heuristic MM in both cases. The performance difference is higher for the nonuniform traffic case. MC decreases the job loss rate by more than 30% over MM in the nonuniform traffic case. The explanation for the difference in performance for nonuniform traffic patterns is similar to that in the $N = 2$ caser. The load balancing brought about by the memory management of MC safeguards against the detrimental effect of the difference in arrival rates, while MM has no such protection. We conclude that MC is more robust than MM.

5 Future Research

Future research will be directed towards designing the optimal strategy for a multi-stage network. Investigation for non-preemptive service models forms an interesting topic of future research.

References

[1] I. Cidon, L. Georgiadis, R. Guerin, and A. Khamisy. Optimal buffer sharing. *IEEE Journal on Selected Areas in Communications*, 13(7), 1995.

[2] F.Kamoun and L.Kleinrock. Analysis of shared storage in a computer network node environment under general traffic conditions. *IEEE Transactions on Communications*, 28(7), July 1980.

[3] H. Gabow and O. Kariv. Algorithms for edge coloring bipartite graphs and multigraphs. *SIAM Journal of Computing*, 11(1), February 1992.

[4] M. Hall Jr. *Combinatorial Theory.* John Wiley and Sons, 1998.

[5] S. Ross. *Introduction to Stochastic Dynamic Programming.* Academic Press, 1998.

[6] S. Sarkar. A scheme for jointly optimum scheduling and memory management under matching constraints. *Technical Report: Available at http://www.seas.upenn.edu/~swati/publication.htm*, 2002.

[7] L. Tassiulas and A. Ephremidis. Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12), 1992.

[8] D. Towsley, P. Sparaggis, and C. Cassandras. Optimal routing and buffer allocation for a class of finite capacity queueing systems. *IEEE Trans. on Automatic Control pp. 1446-1451*, 37(9), 1992.