March 2003

Optimum Scheduling and Memory Management in Input Queued Switches with Finite Buffer Space

Saswati Sarkar
University of Pennsylvania, swati@seas.upenn.edu

Optimum Scheduling and Memory Management in Input Queued Switches with Finite Buffer Space

Abstract
This paper addresses scheduling and memory management in input queued switches with finite input buffers, with the objective of minimizing packet loss. The framework and algorithms proposed here apply to buffer constrained wireless networks as well. The scheduling problem has been extensively addressed under the assumption of infinite input buffers. We study the finite buffer case here which arises in practice. The introduction of memory constraint significantly complicates the problem. The optimal strategies for infinite buffer case no longer apply and become strictly suboptimal in presence of memory limitations. We present closed form optimal strategies which minimize packet loss in 2 x 2 switches with equal arrival rates for all streams. We identify certain characteristics of the optimal strategy for arbitrary arrival rates, and use these properties to design a near optimal heuristic. We use the insight obtained from the investigation for 2 x 2 switches to propose a heuristic for N x N switches, arbitrary N and show numerically that this strategy performs close to optimal. The policies presented here reduce packet loss by about 25% as compared to the optimal strategy for the infinite buffer case.

Keywords
buffer storage, packet switching, queueing theory, scheduling, 2 x 2 switch, N x N switch, arbitrary packet, arrival rate, buffer constrained wireless network, closed form optimal strategy, equal packet arrival rate, finite input buffer space, input queued switch, memory management, optimum scheduling, packet loss minimization

Optimum Scheduling and Memory Management in Input Queued Switches with Finite Buffer Space

Saswati Sarkar

Abstract— **This paper addresses scheduling and memory management in input queued switches with finite input buffers, with the objective of minimizing packet loss. The framework and algorithms proposed here apply to buffer constrained wireless networks as well. The scheduling problem has been extensively addressed under the assumption of infinite input buffers. We study the finite buffer case here which arises in practice. The introduction of memory constraint significantly complicates the problem. The optimal strategies for infinite buffer case no longer apply and become strictly suboptimal in presence of memory limitations. We present closed form optimal strategies which minimize packet loss in 2×2 switches with equal arrival rates for all streams. We identify certain characteristics of the optimal strategy for arbitrary arrival rates, and use these properties to design a near optimal heuristic. We use the insight obtained from the investigation for 2×2 switches to propose a heuristic for $N \times N$ switches, arbitrary N and show numerically that this strategy performs close to optimal. The policies presented here reduce packet loss by about 25% as compared to the optimal strategy for the infinite buffer case.**

I. Introduction

We consider resource allocation in input queued switches with finite memory in the input adapters. The performance objective is to minimize packet loss due to memory overflow. To the best of our knowledge this problem has not been addressed in input queued switches with memory constraints, even though its counterpart with the infinite buffer assumption has been the subject of extensive research. The finite memory constraint introduces significant additional complications, and as we discuss later the optimum strategies are significantly different from the infinite buffer case. For example, if the buffers are assumed to be infinite then the challenge is to decide the packet scheduling so as to attain the desired performance objective. However, in the finite buffer case, additionally one needs to decide whether or not to accept an arriving packet and which packet to drop in case of a memory overflow. This decision has a significant impact on the packet loss performance. Besides the packet scheduling must adapt to the memory constraints.

We first briefly review the existing literature in optimum scheduling in input queued switches. Karol *et al* [6] showed that under FIFO scheduling the throughput is limited to 58.5% of the switching capacity on account of head of line blocking. Mckeown *et al.* [7] presented the maximum weighted matching based scheduling which attains the maximum possible throughput. We will refer to this policy as MM in the rest of the paper. Tassiulas *et al.* [15] presented a maximum difference in backlog based scheduling which attains the maximum possible throughput in any network of constrained queues. The constraints there apply to input queued switches as well. However, all these generic results apply to the case where the queues have infinite buffer space, and consequently they have not addressed memory management at all. Next, we discuss how the resource management problem differs in input queued switches when input buffers are assumed to have finite memory.

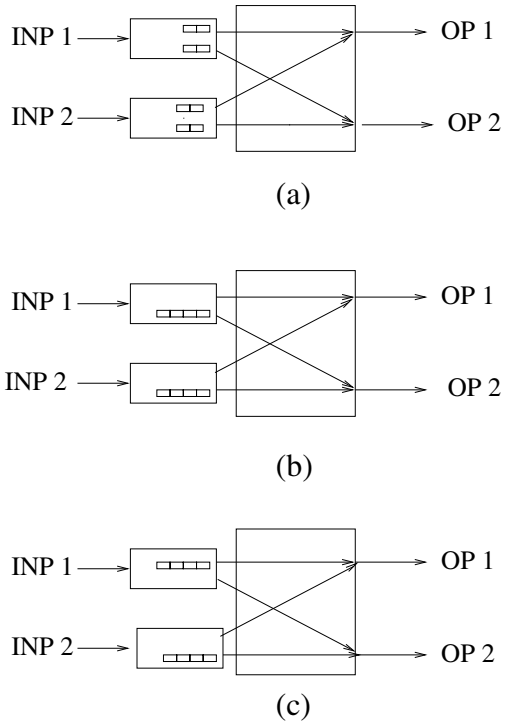

Fig. 1. The figure shows a 2×2 switch with three different buffer occupancies. The buffer occupancies are $\vec{x} = (2, 2, 2, 2)$ (Figure (a)), $\vec{x} = (0, 4, 0, 4)$ (Figure (b)) and $\vec{x} = (4, 0, 0, 4)$ (Figure (c)). Only one packet can be transferred to the output at a time in Figure (b), while two packets can be transferred simultaneously in the other two cases.

The resource management problem has two components when the input buffers have finite memory: (a)packet scheduling and (b)memory management. The first decides which packets can be scheduled without violating the switching constraints imposed by the input queued switch. Input queueing

S. Sarkar is with the Departments of Electrical and Systems Engineering and Computer and Information Sciences with the University of Pennsylvania. Her email is swati@ee.upenn.edu

The work of Saswati Sarkar was supported in part by NSF grant ANI01-06984.

introduces several scheduling inter-dependencies, and allows only certain scheduling patterns. The scheduling should be such that an input or output port can be used for the transfer of only one packet at a time. Different packets can be simultaneously transferred to the outputs as long as these do not share the same input or output port. Memory management determines the packet acceptance policy, namely which incoming packet will be accepted, and which packet should be dropped in case of memory overflow (the choice is between incoming packets and existing packets). These decisions will depend on buffer occupancy and possibly arrival and service statistics. More importantly, the packet scheduling and memory management decisions must be taken in conjunction, and will depend on each other. For example, only one packet can be transferred to the output side if all outstanding packets are waiting at the same input or are destined to the same output. However, many more packets can be transferred if the outstanding packets do not have common inputs and outputs. Refer to figure 1 for an illustration. Memory management can be utilized efficiently to ensure that fewer outstanding packets share input and output ports. We discuss this in details later. To the best of our knowledge, neither problem has been addressed in context of input queued switches with memory constraint. Interestingly, we observed that the MM scheduling strategy [7] which maximizes throughput in input queued switches with infinite buffers is strictly suboptimal in terms of throughput in presence of memory constraints.

Memory management has been addressed for other types of switch architectures, e.g., shared memory switches[1], [2], [5], [12]. However, scheduling is not an issue in these switches. This is because multiple packets can be simultaneously transferred to an output from different inputs. The outputs can simultaneously serve packets independent of each other. Memory is the only shared resource in these switches. Thus the problem of deciding a jointly optimal scheduling and memory management strategy does not arise, and the memory management strategies need not be designed to cater to scheduling dependencies in these switches.

Similar scheduling and memory management problems arise in optical switches as well[17], and these are seriously constrained in memory. Also, mathematically the scheduling and memory management problems in input queued switches are equivalent to those in finite memory nodes in wireless ad hoc networks. This observation has been made elsewhere as well [14], and scheduling policies which attain maximum throughput in wireless networks have been found to maximize the throughput in networks of input queued switches as well [7], [15]. The significance of this observation is that *mathematically the problem we address in this paper forms the core of several applications of practical utility, and the solution of this problem is expected to apply to all these problems.* We elaborate this aspect later (pp. 2).

The objective of this paper is to provide a theoretical basis for the problem of jointly optimum scheduling and memory management. The investigation yields several apparently counter-intuitive results as will be described later. First we mathematically model the problem and show its close connection with the bandwidth and memory management problems in wireless adhoc networks in section II. The optimum strategy can be obtained by computationally solving a markov decision process [13]. Markov decision process based computations are time and memory intensive, and the computations become intractable even for switches with moderate buffer sizes. However, using this framework we present a closed form computationally simple optimal scheduling and memory management strategy for input queued switches with 2 inputs and 2 outputs under the assumption of equal arrival rates for all input-output streams in section III-A. The optimal scheduling transfers as many packets as possible at all times. The optimal memory management uses a packet acceptance which balances the number of outstanding packets for the different outputs. This in turn allows the scheduling to transfer several packets to the output and reduces the buffer occupancy and thus the packet loss in the switch. We obtain several features of the optimal strategy in 2×2 switches for arbitrary arrival rates in section III-B, and using these properties we design a near optimal heuristic strategy (SOP) in this case. The numerical results will also demonstrate that the MM scheduling which is known to be optimal in absence of memory constraint [7] has 25% more packet loss than SOP in many cases. We present a heuristic strategy (BCT) for the general case of switches with N inputs and N outputs in section IV using the strategies for the 2×2 switches and certain properties derived from the general markov decision process framework. We present numerical performance evaluation to demonstrate that the packet loss experienced by BCT is close to the minimum possible value. In addition, BCT performs substantially better than MM, and in many cases reduces the packet loss by 40% or more as compared to MM. We discuss implementation challenges and future research directions in context of specific applications such as input queues switches, wireless networks and optical switches in section V. *We omit proofs here on account of space constraint. Refer to technical report [13] for details.*

II. Network Model

We consider an input queued switch with N inputs and N outputs (Figures 1 and 2). The size of the input buffers are $B_1, \ldots, B_N$. Incoming packets have predetermined outputs and the arrival processes are independent Poisson. The arrival rate of packets at input i for output j is λ_{ij}, $i = 1, \ldots, N$, $j = 1, \ldots, N$. Packets are stored in the respective input buffers until they are transferred to the intended outputs. Queue length of packets waiting at input i for output j at time t is $x_{ij}(t)$. Clearly $\sum_{j=1}^{N} x_{ij}(t) \leq B_i$. Packets have exponential duration with the service rate for the packets at the ith input and jth output is μ_{ij}. For simplicity we will assume equal buffer sizes and equal service rates, i.e., $B_i = B$ for all i and $\mu_{ij} = \mu$ for all i, j. We will also assume that all packets which start service at the same time end service at the same time as well. These assumptions simplify the mathematical framework and

the analysis substantially, without altering the nature of the results. The assumptions can be justified from the observations that switch performance is a function of the arrival rates, service rates and the buffer sizes, and unequal buffer sizes and service rates have the same effect on the performance as unequal arrival rates. We will study the effect of unequal arrival rates. In fact, the performance depends on the ratio between the arrival and service rates rather than their absolute values. We relax many of these assumptions in simulation, e.g., we consider the effect of unequal packet sizes and heavy tailed arrival processes on the design (Figure 10).

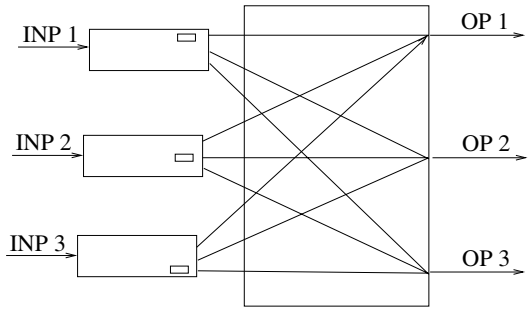

Fig. 2. The figure shows a switch with 3 inputs and 3 outputs. Here, each input has only one packet waiting for transfer to the output, $x_{11} = x_{22} = x_{33} = 1, x_{ij} = 0$, if $i \neq j$. The input-output pairs $1-1, 2-2, 3-3$ constitute a matching. Similarly, $1-2, 2-1, 3-3$ is another example matching. Either matching can be scheduled, but the first transfers 3 packets while the second transfers only one packet as the $1-2$ and $2-1$ queues are empty ($x_{12} = x_{21} = 0$).

An input-output pair can be involved in transfer of only one packet at a time, i.e., while input i is transferring a packet to output j, input i can not transfer any other packet and output j can not receive any other packet. The scheduling problem is to decide which packets are transferred from the input to the output at any time t. Clearly the transfers must constitute a matching* at any time t. There can be several possible matchings and the scheduling problem at any time is to choose the appropriate matching. Refer to Figure 2 for examples of valid scheduling. Every time a packet transfer is completed from input i to output j the queue length x_{ij} decreases by one.

The memory management problem is to decide how the input buffers should be shared by packets intended for different outputs. When a new packet arrives at an input, one of the following actions can be taken: (a)the packet is accepted (b)the packet is rejected and (c) the packet is accepted while some other packet waiting at the input is dropped. The last action is commonly referred to as "push-out" (following the terminology of [1], [2], [12] in a different context). If a packet arrives when the input queue is full, then one of the last two actions must be taken. The memory management problem is to choose the appropriate course of action when a new packet arrives. Every time a new packet is accepted in input i for output j, the queue length x_{ij} increases by 1. If a packet waiting at input i intended for output j is dropped, x_{ij} decreases by one.

The objective is to choose the scheduling and the memory management scheme so as to minimize the average packet loss. A packet loss happens every time an incoming packet is rejected or an existing packet is dropped. Throughput of the switch is the average number of packets transferred from the inputs to the outputs. Throughput maximization and loss minimization are equivalent objectives, as the sum of throughput and loss rates is equal to the sum of the arrival rates. Thus a strategy which minimizes loss maximizes throughput and vice versa.

Now we argue how the finite memory assumption complicates the problem. Firstly, the issue of memory management arises only because of the finite buffer assumption. Packet loss never happens and thus packets can always be accepted for infinite buffers. However, in presence of finite buffers, which is the case in practice, packet loss happens whenever an incoming packet finds the buffer full, and thus packet acceptance and scheduling decisions need to be sophisticated so as to minimize this loss. Secondly, a scheduling which maximizes the system throughput under the infinite buffer assumption will not maximize the throughput (and hence minimize the loss) in the finite buffer case in general. This can be explained as follows. A policy is said to maximize the system throughput under the infinite buffer case if it "stabilizes" the system under maximum possible load, i.e., the policy must "stabilize" the system for all arrival rates $(\lambda_{11}, \ldots \lambda_{NN})$ such that$\sum_{i=1}^{N} \lambda_{ij} < \mu$ and $\sum_{j=1}^{N} \lambda_{ij} < \mu$ $\forall\ i, j$ [7], [14]. A system is said to be stable as long as the queues become empty ($x_{ij}(t) = 0 \forall i, j$) infinitely often. Note that for stability it is not important *how soon* the queues empty, but it is sufficient that the system reaches the all zero state infinitely often. However, in the finite buffer case, the average time taken to empty the queues affect the performance of the system. Consider a simple example to illustrate this fact. Consider a switch with 2 inputs and 2 outputs ($N = 2$), $\lambda_{11} = \lambda_{22} = 0.1$, $\lambda_{12} = \lambda_{21} = 0$ $\mu = 0.4\ \forall i, j$. Consider two different policies. The first is work conserving and serves the packets in FIFO order at the output terminals. The second serves the packets in FIFO order but takes a vacation after serving every packet. The duration of the vacation is an exponential random variable with average 2.5 unit. Both of these policies stabilize† the system and attain the same throughput under infinite buffer assumption (system throughput is 0.2 for both policies). However, the average duration of the busy period for the second is at least twice that for the first, and the packet loss is higher in the second for any finite buffer length. The bottomline is that the infinite buffer assumption offers the system significant latitude, while in presence of memory constraints the scheduling strategies need

*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.

†The $1-1$ and $2-2$ streams are served independent of each other in both cases. The service rates are greater than the arrival rates in both cases (the service rate is $1/(2.5 * 2)$ in the second case), and thus the system is stable in both cases.

to use resource more carefully. Thus not all policies which optimize the performance under infinite buffer assumption will do so in the finite buffer case. In fact, as discussed before, we will demonstrate that the MM scheduling which maximizes system throughput under infinite buffer assumption [7] does not maximize throughput in the finite buffer case.

Finally, we point out the similarities between resource allocations in input queued switches and wireless adhoc networks. Incidentally, future adhoc networks are expected to consist of small, light weight terminals like PDAs, palmtops or laptops which must perform the functions of end nodes as well as intermediate routers. Thus these nodes will be constrained in both memory and bandwidth, and hence efficient usage of both resources via scheduling and memory management is necessary for meeting the performance objectives. Consider an adhoc network with nodes $1, \ldots, N$. Node i has buffer B_i. Every node has a locally unique frequency [‡], but only one radio unit. A node can be involved in at most one transmission at a time in the role of either a transmitter or a receiver on account of the single radio constraint. Several transmissions can proceed simultaneously as long as every node is involved in at most one transmission. At any time the successful transmissions must constitute a matching, and the scheduling problem is to decide the appropriate matching like in input queued switches[§]. The memory management problem is to decide whether to accept an incoming packet at a node and also whether an existing packet should be pushed out by a new packet. This is the same as the memory management problem in input queued switches. We conclude that the scheduling and the memory management problems are similar in both cases. This further motivates the investigation of the scheduling and the memory management problem in either case, as the solution for one may be used in the other case as well.

The optimal scheduling and memory management policy is the one which minimizes the average packet loss. We present a generic markov decision process(MDP) [10], [11] based technique for computing the optimal loss rates and the optimal scheduling and memory management decisions in technical report [13]. However, the computations become intensive with increase in B or N. This happens because the computation needs several iterations and each iteration has a complexity $\Omega\left(B^{2N^2}\right)$.[¶] For example, for a 2×2 switch with $B = 50$ each iteration involves computations with 10^6 variables. The bottomline is that the generic computation technique does not scale with the buffer size. However, we will use the general results obtained from this MDP framework to compute closed form optimal strategies for the simple case of 2×2 switches where all arrival streams have equal arrival rates ($\lambda_{ij} = \lambda$ for all i, j). Subsequently we will use the insight obtained from the optimal strategy for this specific case and certain properties deduced from the MDP framework to design near optimal heuristics for the more general cases of 2×2 switches with arbitrary arrival rates and $N \times N$ switches with $N > 2$.

[‡]Several current day networks have locally unique frequencies, e.g., Bluetooth networks.

[§]In wireless adhoc network a node may only be able to transmit packets to a subset of other nodes as the rest may be out of its transmission range. Similarly, in input queued switches an input node may not transmit packets to all output nodes, e.g., λ_{ij} may equal zero for certain pairs i, j.

[¶]A computation is said to have $\Omega(n)$ complexity for input size n if it requires at least cn steps for some constant c and all large n.

III. Optimal Strategy for 2×2 switches

In this section we will consider the scheduling and memory management strategies for input queued switches with 2 inputs and 2 outputs ($N = 2$). We will first present the intuition behind the optimal resource allocation problem in this case. Subsequently we will present the closed form optimal strategy for symmetric traffic in subsection III-A. Extension to the case of asymmetric traffic is considered in subsection III-B. There we obtain certain key properties of the optimal strategy for the asymmetric traffic case, and then using these properties we design a heuristic. In subsection III-C we will show that this heuristic attains near optimal packet loss using numerical computations.

We first describe the optimal resource allocation problem in this case. Refer to Figure 1 for illustration. There are four different queues of packets, x_{11}, x_{12}, x_{21} and x_{22}. The queue x_{ij} stores packets at input i waiting for output j. The first two queues share the memory in input buffer 1 and the other two share the memory in input buffer 2. Arrival process is poisson for each of these queues with rates $\lambda_{11}, \lambda_{12}, \lambda_{21}$ and λ_{22} respectively and the service process is exponential at rate μ. Each buffer can store a maximum of B packets.

The pairs $[x_{11}, x_{22}]$ and $[x_{12}, x_{21}]$ can transmit packets simultaneously as the queues in these pairs do not share an input or output port. The switch can also schedule a single queue at a time. No other combination is allowed. Packet loss can be minimized if the buffer occupancy is reduced. Intuitively this happens if a pair of queues is scheduled together rather than singletons because the former removes two packets from the buffer while the latter removes only one packet from the buffer. However, if the buffer occupancy is such that one queue is empty in each pair then the switch will be forced to schedule singletons. For example, if $x_{11} = x_{21} = 0$ then the switch can schedule either x_{12} or x_{22} (Figure 1(b)). For further illustration consider the buffer occupancies in Figures 1(a) ($\vec{x} = (2, 2, 2, 2)$) and 1(b)($\vec{x} = (0, 4, 0, 4)$). Observe that both configurations have the same queue lengths at the two inputs (4 packets) and also the total number of packets waiting for transmission is the same for both (8 packets). The switch can transmit only one packet in Figure 1(b) (from x_{12} or x_{22}) as all packets are intended for the same output (output 2). However, the switch can transmit two packets in Figure 1(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 packets can be transmitted and hence packet 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 1(b) ($\vec{x} = (0, 4, 0, 4)$) and 1(c)($\vec{x} =$

$(4, 0, 0, 4)$). Comparing the configurations in the two cases, only two queues have packets in both cases and the queue lengths at the two inputs (4 packets) and the total number of packets waiting for transmission (8 packets) are the same in both. However, the switch can transfer only one packet in Figure 1(b) as discussed before, whereas two packets can be transferred in Figure 1(c) (from queues x_{11} and x_{22}) since there are packets waiting at different inputs intended for different outputs. The configuration in Figure 1(c) allows a transfer of a larger number of packets because it 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 1(a) and 1(c) and 4 for Figure 1(b). In general, a larger number of packets can be transferred if these differences are smaller.

Appropriate push-out policy can be used to reduce these differences. When a new packet arrives and the input buffer is full, then the packet can be rejected or accepted by pushing out an existing packet. The appropriate decision can be taken with a view to reducing this difference. Assume that $B = 4$ in Figure 1. Let a packet arrive at input 1 for output 1. In Figure 1(b) the differences between the queue lengths of the queues in the pairs remain $4, 4$ if the new packet is rejected, while the differences become $3, 3$ if the new packet is accepted while pushing out an existing packet waiting at input 1 for output 2 (in the x_{12} queue). After this replacement, both queues x_{11} and x_{22} can be scheduled transferring two packets simultaneously. In Figure 1(a) the new packet should be rejected as its acceptance using push-out will adversely affect the balance and increase the differences (packet rejection leaves the differences at $0, 0$ and push out increases the differences to $1, 1$). Summarizing, *a pair of queues should be scheduled whenever possible, and packet acceptance/rejection/push-out should be used judiciously to balance the load across the appropriate queues which facilitates a simultaneous transfer of two packets whenever possible.* These observations are the key behind designing optimal strategies for symmetric traffic and near-optimal heuristics for asymmetric traffic.

A. Optimal Strategy for Symmetric Traffic

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 which minimize the average packet loss in this case. We will refer to the optimal policy as "SOP" (**symmetric optimal policy**).

SOP Scheduling: The scheduling strategy needs to decide which queues to serve whenever the current packets finish transmission. The optimal scheduling is to serve the queues of either pair 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 can be arbitrary and does not affect packet loss(proved in [13]). 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. If all the queues are empty ($\vec{x} = \vec{0}$), no queue is scheduled for service.

SOP Memory Management: The memory management strategy decides whether to accept an incoming packet, and if the decision is to accept the packet then whether to push out an existing packet. The optimal decision is to accept an incoming packet without any push out as long as the input buffer has space. If the input buffer is full when a new packet arrives, then the packet is either rejected or accepted while dropping an existing packet from a different queue in the same input buffer. 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: $\text{diff}_1(\vec{x}) = x_{11} - x_{22}$, $\text{diff}_2(\vec{x}) = x_{12} - x_{21}$. Let a packet arrive at input buffer 1 for output 1 (in queue $1-1$) and let input buffer 1 be full ($x_{11} + x_{12} = B$). Then the new packet is rejected if $\text{diff}_1(\vec{x}) \geq \text{diff}_2(\vec{x}) - 1$, accepted and a packet of queue $1-2$ is dropped otherwise. Let $x_{11} + x_{12} = B$, and a new packet arrives at queue $1-2$. Then the new packet is rejected if $\text{diff}_2(\vec{x}) \geq \text{diff}_1(\vec{x}) - 1$, accepted and a packet of queue $1-1$ is dropped otherwise. Let $x_{21} + x_{22} = B$, and a new packet arrives at queue $2-1$. Then the new packet is rejected if $\text{diff}_2(\vec{x}) \leq \text{diff}_1(\vec{x}) + 1$, accepted and a packet of queue $2-2$ is dropped otherwise. Let $x_{21} + x_{22} = B$, and a new packet arrives at queue $2-2$. Then the new packet is rejected if $\text{diff}_1(\vec{x}) \leq \text{diff}_2(\vec{x}) + 1$, accepted and a packet of queue $2-1$ is dropped otherwise. An illustrative example follows.

Example III.1: We illustrate SOP using the configurations in Figure 1. Let $B = 4$. In Figure 1(a) ($\vec{x} = (2, 2, 2, 2)$), SOP schedules either the pair $[1-1, 2-2]$ or the pair $[1-2, 2-1]$ choosing between them arbitrarily. Any incoming packet is rejected. In Figure 1(b) ($\vec{x} = (0, 4, 0, 4)$), only one queue can be scheduled. SOP schedules either $1-2$ or $2-2$ choosing between them arbitrarily since both have the same length (4 packets). At either input, an incoming packet for output 1 is accepted by pushing out a packet waiting for output 2 at the same input, and an incoming packet for output 2 is rejected. In Figure 1(c) ($\vec{x} = (4, 0, 0, 4)$), SOP schedules the pair $1-1, 2-2$. Any incoming packet 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 packet in all three cases.

The following theorem outlines the optimality of SOP.

Theorem 1: SOP minimizes the average packet loss in 2×2 switches with equal arrival rates for all streams ($\lambda_{ij} = \lambda$ for all i, j).

B. A near optimal heuristic for asymmetric traffic

In this section we will consider the general case with unequal arrival rates for different arrival streams, i.e., we

no longer assume that $\lambda_{ij} = \lambda$. First, we identify certain properties of the optimal scheduling and memory management strategy, and subsequently present 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 packet without any push out as long as the input buffer has space.

These properties specify the strategy in certain cases. For instance, in Figure 1(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 packets for all three configurations in Figure 1. 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 1(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 1(b)). The packet acceptance/rejection/push-out decision when the input buffer is full is also not known in the asymmetric case (e.g., all three cases in Figure 1 with $B = 4$).

We propose SOP (Section III-A) as a heuristic here. We justify the choice as follows. SOP satisfies the properties obtained for the optimal strategy in this general case. Besides, the scheduling and the memory management decisions of SOP strive to balance the traffic across different queues which allow the scheduling to transfer larger number of packets and thus reduce the buffer occupancy and the packet loss. Numerical computations presented in the next subsection will demonstrate that the heuristic attains near minimum packet loss.

We have obtained some other properties of the optimal strategy under some additional assumptions. Let $\lambda_{i1} = \lambda_{i2}$ $i = 1, 2$ (we still allow $\lambda_{1j} \neq \lambda_{2k}$ for any j, k i.e., unequal arrival rates at different inputs). Under this additional assumption, (a)when all queues have packets, then either pair can be scheduled and the optimal scheduling does not distinguish between pairs, (b) if an incoming packet finds the input buffer full, then the optimal packet acceptance/rejection/push-out decision is the same as that for SOP (Section III-A) [13]. Under this additional assumption, the optimal strategy is known for all cases in Example III-A.1, except the scheduling in Figure 1(b). More generally, this additional assumption fully specifies the optimal memory management scheme. The scheduling policy is also specified except that it is not known how to choose between queues when only one queue can be scheduled(e.g., Figure 1(b)).

C. Performance Evaluation using Numeric Computation

First we evaluate the performance of SOP for unequal arrival rates, using the packet loss experienced by the optimal strategy as a benchmark. We compute the average packet losses of SOP and the optimal strategy using MDP based techniques [13]. We have observed that SOP attains near minimum average packet loss. We present numerical results for two different traffic patterns. The computation technique uses the MDP formulation, and has been described in technical report [13].

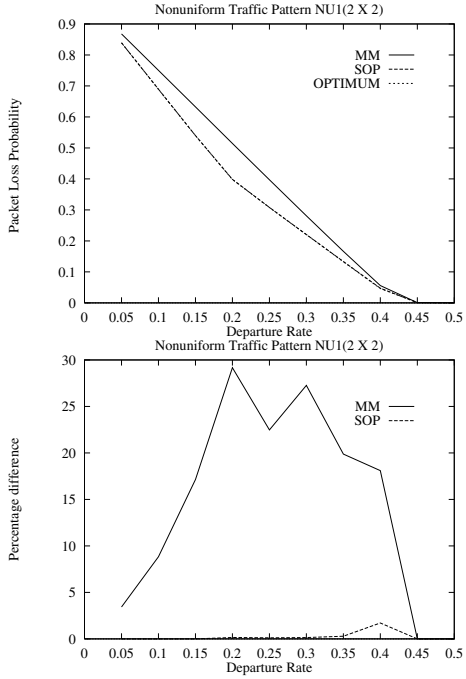

Fig. 3. The figures compare the performances of SOP and MM with the optimal strategy for different departure rates (μ) in a 2×2 switch with $B = 50$ and the following arrival rate patterns: $\lambda_{11} = 3.5\lambda_{12} = 3.5\lambda_{21} = 7\lambda_{22}, \lambda_{22} = \frac{1-\mu}{15}$. Figure (a) plots the absolute values of the packet loss probabilities ($l_{\text{MM}}, l_{\text{SOP}}, l_{\text{OPTIMUM}}$). The SOP and the OPTIMUM curves can not be distinguished, while the MM curve is above both. Figure (b) plots the percentage relative difference between the loss probabilities for the SOP and optimal (curve SOP) $(100 * (l_{\text{SOP}}/l_{\text{OPT}} - 1))$, and that between MM and SOP (curve MM) $(100 * (l_{\text{MM}}/l_{\text{SOP}} - 1))$. The SOP curve is close to zero, and hence hardly visible.

When all arrival rates are equal, there is no difference between the arrival rates of queues in the pairs which can be scheduled together, and as discussed these differences are important metrics. Thus in nonuniform traffic model, we study the cases when the differences between arrival rates of queues in these pairs are nonzero. First, we consider 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." We compare the performances of SOP with the optimal for different values of the departure rates (μ) (Figure 3), and for each μ we choose the arrival rates such that the above ratios are satisfied and $\mu + \sum_{i=1}^{2}\sum_{j=1}^{2} \lambda_{ij} = 1$ ($\lambda_{22} = \frac{1-\mu}{15}$ ensure this). Figure 3 show that the performances of the two are almost identical in most cases with the difference between the loss probabilities less than 2% for all μ (the curves can not be distinguished). Next, we consider the case where the arrival rates are unequal for both pairs, while the total arrival rates are equal for both inputs, i.e, $\lambda_{11} + \lambda_{12} = \lambda_{21} + \lambda_{22}$ (this did not hold in the previous case). Here, $\lambda_{11} = 4\lambda_{22}$, $\lambda_{21} = 4\lambda_{12}$

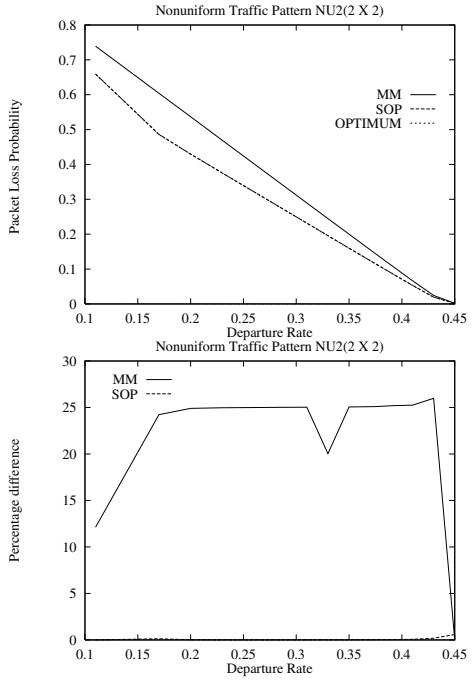

Fig. 4. The figures compare the performances of SOP and MM with the optimal strategy for different departure rates (μ) in a 2×2 switch with $B = 50$ and the following arrival rate patterns: $\lambda_{11} = \lambda_{21} = 4\lambda_{12} = 4\lambda_{22}, \lambda_{22} = \frac{1-\mu}{10}$. Figure (a) plots the absolute values of the packet loss probabilities($l_{\text{MM}}, l_{\text{SOP}}, l_{\text{OPTIMUM}}$). The SOP and the OPTIMUM curves are very close to each other and can not be distinguished, while the MM curve is above both. Figure (b) plots the percentage relative difference between the loss probabilities for the SOP and the optimal (curve SOP) $(100 * (l_{\text{SOP}}/l_{\text{OPT}} - 1))$, and that between MM and SOP (curve MM) $(100 * (l_{\text{MM}}/l_{\text{SOP}} - 1))$. The SOP curve is close to zero, and hence hardly visible.

and $\lambda_{11} = \lambda_{21}$. We denote this pattern as "NU2." Again, we vary μ and for each μ we choose the arrival rates such that the above ratios are satisfied and $\mu + \sum_{i=1}^{2}\sum_{j=1}^{2}\lambda_{ij} = 1$ ($\lambda_{22} = \frac{1-\mu}{10}$ ensure this). Figure 4 show that the performance of SOP is near optimal all through, again the curves can not be distinguished. The trends remain similar for other patterns of arrival rates which we do not exhibit on account of space constraints.

We believe that the minor difference in the packet losses of SOP and the optimal strategy is principally due to the sub-optimality of the packet acceptance/rejection/push-out decision of SOP in the general case. Consider traffic pattern NU1 as an example. Let $\vec{x}$ be the current state. Since the $1-1$ stream has significantly higher arrival rate as compared to other streams, the optimal strategy should push out the packets of the $1-2$ stream less frequently than that of the $1-1$ stream when input buffer 1 is full, i.e., $1-2$ packets should be pushed out only if $\text{diff}_1(\vec{x})$ is much smaller than $\text{diff}_2(\vec{x})$. This is because the $1-1$ stream will receive more packets than the $1-2$ stream in future. However, SOP pushes out the $1-2$ packets whenever a $1-1$ packet finds input buffer 1 full and $\text{diff}_2(\vec{x}) > \text{diff}_1(\vec{x}) + 1$, even if the difference between the two sides is small. But again, a state $\vec{x}$ where $\text{diff}_2(\vec{x}) > \text{diff}_1(\vec{x}) + 1$ is rarely reached since the $1-1$ stream has significantly high arrival rate as compared to the other streams. Thus there is only slight sub-optimality. Also, depending on the arrival rates, unlike SOP, the optimal strategy may not schedule the longest queue whenever only a single queue can be scheduled. Once again, the sub-optimality is minimal.

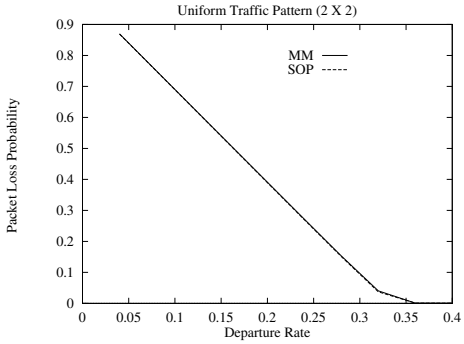

Fig. 5. The figure plots the packet loss probabilities of MM and SOP for different departure rates (μ), and equal arrival rates $\lambda_{ij} = \frac{1-\mu}{4}, \forall i, j$ in a 2×2 switch with $B = 50$. Note that SOP is optimal in this case (Theorem 1).

Now we consider the MM policy which schedules packets as per a maximum weighted matching. This policy has been proposed for the infinite buffer case and attains the maximum possible throughput there [7]. Memory management is not an issue under infinite buffer assumption, and as such MM does not address it. We consider the obvious extension, whereby MM always accepts a packet whenever there is space in the buffer and always rejects a packet if the input buffer is full. Push out is not used. Average packet loss experienced by this policy can be computed using MDP based techniques [13]. Numerical computation shows that MM has considerably higher packet loss than SOP for unequal arrival rates. For traffic patterns NU1 and NU2 (Figures 3 and 4 respectively), SOP decreases the packet loss by more than 25% in certain cases. Note that the difference brought about by intelligent resource allocation is pronounced for medium departure rates. For large departure rates, all policies attain low packet loss and all policies suffer from heavy packet loss in the other extreme. Thus MM and SOP perform similarly in these two extremes, while SOP substantially outperforms MM in the middle. We observed similar trends for other unequal arrival rate patterns as well. When arrival rates are equal, MM is consistently inferior to SOP for all departure rates(Figure 5). However, the difference in performance is small in this case, and thus the curves look identical in the figure.

The relative performance of MM w.r.t. SOP is much worse for nonuniform traffic (Figures 3 and 4) than for uniform traffic (Figure 5). 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. This load mismatch worsens the performance for MM. Thus MM experiences much heavier packet loss for non-uniform traffic than for uniform traffic. The memory management in SOP restores a balance in the queue lengths by replacing packets 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 for the uniform traffic case. As an example, at service rate 0.2, sum of the arrival rates ($\sum_{i=1}^{2}\sum_{j=1}^{2}\lambda_{ij}$) 0.8, MM has 39.1% and 51.5% packet loss for equal arrival rates and arrival pattern NU1 respectively, while the numbers are 38.95% and 39.87% for SOP. This shows that SOP is robust to different traffic patterns (note that neither SOP nor MM uses statistics information in the decision process).

Finally we discuss the implications of the sub-optimality of MM. The main reason for sub-optimality of MM is that it does not use any intelligent memory management scheme which is important for performance optimization when buffers are finite. The scheduling used by MM differs from the optimal scheduling as well. Note that the optimal scheduling schedules a pair whenever possible both for symmetric and asymmetric traffic, but this is not the case with MM. Thus the optimal scheduling is to schedule a matching of maximum size or a "maximum matching," which is different from the maximum weighted matching MM uses. We explain MM scheduling to illustrate this. The matchings for a 2×2 switch are the single queues $[1-1], [1-2], [2-1], [2-2]$ and the pairs $[1-1, 2-2]$, $[1-2, 2-1]$. Weight of a matching is the sum of the queue lengths of the queues in the matching. MM schedules the queues which constitute the matching with the maximum weight. Let the current state be $\vec{x}$. Suppose $x_{22} = 0$, $x_{11} > x_{12} + x_{21}$ and $x_{12} > 0, x_{21} > 0$ (e.g., $\vec{x} = (5, 2, 2, 0)$). MM schedules only the queue $1-1$, while the optimum scheduling schedules the pair $1-2, 2-1$. Thus the optimum scheduling transfers a larger number of packets than MM. This reduces the buffer occupancy and hence the packet loss for the optimal strategy. MM attains optimum throughput in the infinite buffer case in spite of transferring fewer number of packets because the infinite buffer assumption offers more latitude than the finite buffer case as discussed before. Interestingly, an example provided by Mckeown *et al.* shows that for infinite buffers maximum matching scheduling is strictly suboptimal in terms of throughput [7], whereas maximum weighted matching attains optimum throughput. Apparently, it is somewhat counterintuitive that the result will be opposite in the finite buffer case, particularly since infinite buffer assumption is the limiting case of finite buffers. This apparent contradiction can be explained by two observations. (a)The example was provided for 3×3 switches while the optimal scheduling properties presented so far are for 2×2 switches. (b)The example showed that the choice of a *specific* maximum matching is suboptimal for unequal arrival rates and does not show that every maximum matching based scheduling is suboptimal. There can be several maximum matchings. The analytical results obtained here show that the optimal scheduling is to choose "some" maximum matching or rather to choose a pair of queues whenever possible. The choice between the pairs can be arbitrary for symmetric traffic, but the choice may matter for asymmetric traffic.

We compared the performance of SOP with some other scheduling strategies, e.g., choosing the maximum weighted matching of maximum size etc. The performance differences are similar to that with MM (technical report [13]), which indicates that the performance advantage of SOP is primarily due to the load balancing attained by the memory management. Refer to [13] for performance comparisons for heavy tailed and bursty arrival processes.

IV. Resource Allocation for $N \times N$ switches for arbitrary N

We consider the scheduling and memory management problems in switches with N inputs and N outputs for arbitrary N. The objective is to minimize the packet loss. The optimal strategy can be computationally obtained using MDP [13]. However, no closed form solution is known for this case. Since computations using MDP are complex, we propose a computationally simple heuristic in this case and show using numerical computation that this heuristic attains low packet loss.

The objective of the scheduling policy is to transfer as many packets as possible so as to minimize the packet loss, and memory management decides which packets to accept, reject and push-out so as to allow the transfer of several packets in every scheduling. Consider the example shown in Figure 2. All three packets 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 2×2 switches we need to balance the load across appropriate queues. The SOP memory management attains this in 2×2 switches by accepting/rejecting/pushing out packets 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, e.g., $1-1, 2-2, 3-3$ can be scheduled together in the 3×3 switch showed in Figure 2. 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 packets 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 2, $\mathrm{cong}(\vec{x}) = 3$ if all three packets were waiting at the same input, or waiting for the same output in Figure 2. Three packets can be transferred to the outputs simultaneously in the first case, while only one packet can be transferred in the latter cases. Thus intuitively larger number of packets can be transferred when $\mathrm{cong}(\vec{x})$ is smaller. We propose a strategy which minimizes this congestion measure. First we justify why $\mathrm{cong}(\vec{x})$ is a measure of goodness for a state $\vec{x}$.

1) If the current state is $\vec{x}$, every packet has duration T units and there is no future arrival, then the minimum time taken to transfer all waiting packets is $T\mathrm{cong}(\vec{x})$ [4], [16]. This indicates that higher the value of $\mathrm{cong}(\vec{x})$, more congested is the switch in some sense.

2) We show in technical report [13] that the optimal policy minimizes a parametrized function $J_{\beta,\text{OPT}}(\vec{x})$ (parameter β) referred to as the "discounted loss rate function" for each state $\vec{x}$ and all values of a certain parameter $\beta < 1$. We show here that under heavy traffic this function is lower bounded by $\text{cong}(\vec{x})$ for each state $\vec{x}$ and each $\beta < 1$.
Lemma 1: For all $\beta < 1$ and all states $\vec{x}$ $J_{\beta,\text{OPT}}(\vec{x}) \geq \text{cong}(\vec{x})$ if the total arrival rate at(intended for) each input(output) is greater than the service rate, i.e., $\min(\min_i \sum_{j=1}^{N} \lambda_{ij}, \min_j \sum_{i=1}^{N} \lambda_{ij}) > \mu$.
This indicates that low packet loss can be obtained by choosing the next state so as to minimize this measure of congestion.

We present the heuristic now. The scheduling is to schedule the matching of largest size which minimizes $\text{cong}(\vec{y})$ 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 packets complete transmission. Fast algorithms can be found for computing such matchings using the theory of edge coloring of bipartite graphs [3]. The memory management policy accepts packets without pushing out any existing packet as long as the input buffer has space (in fact this is the optimal decision [13]). We consider the acceptance decision when an incoming packet finds the input buffer full. Let the current state be $\vec{x}$. Suppose the packet is for input i and output j. We consider the total number of packets waiting for each output across all inputs. Let this number be maximum for output k among those outputs for which at least one packet wait at input i, i.e., $k = \arg\max_{m:x_{im}>0} \sum_{l=1}^{N} x_{lm}$. The incoming packet is rejected if $\sum_{l=1}^{N} x_{lj} \geq \sum_{l=1}^{N} x_{lk} - 1$, and accepted while pushing out a packet waiting for output k at input i otherwise. In other words the acceptance policy reduces the load for a more heavily loaded output at the expense of that for a lightly loaded output. Overall, the policy minimizes the congestion at both inputs and outputs, and thus we denote it as a policy which balances congestion at terminals (BCT). We present an example to illustrate this policy.

Example IV.1: Consider a 3×3 switch. Let the current state be $\vec{x}$ with $x_{11} = 2, x_{12} = x_{21} = x_{32} = 1$. All other queues are empty. Note that at most 2 packets can be transferred, and the matchings which transfer two packets are $[1-1, 3-2]$, $[1-2, 2-1]$, $[2-1, 3-2]$. The next states have congestion measures $2, 2, 3$ respectively for these choices. Thus either the first or the second matching can be selected. Suppose $B = 3$. Any new arrival at inputs 2 and 3 are accepted without any push outs. Consider a new arrival at input 1 which is full. If the packet is for output 1 or 2, it is rejected. However, a new arrival for output 3 is accepted while pushing out a packet for output 1, since $\sum_{m=1}^{3} x_{m1} > \sum_{m=1}^{3} x_{m3} + 1$.

BCT applies for 2×2 switches as well, and takes the same decisions as SOP there [13]. Weller *et al.* [16] proposed a similar scheduling strategy, and analyzed its throughput in presence of infinite buffers and pseudo-deterministic traffic arrival (a "$\alpha - S$" traffic model) [16]. However, memory constraints were not considered there, and as such the memory management component has not been proposed.

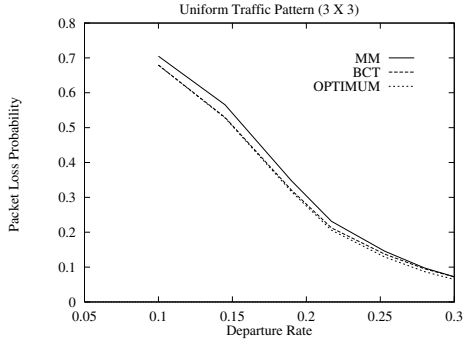

Fig. 6. The figure plots the packet loss probabilities of the optimum strategy, BCT and MM for different departure rates (μ), and equal arrival rates $\lambda_{ij} = \frac{1-\mu}{9}, \forall i, j$ in a 3×3 switch with $B = 5$. The packet losses are almost identical for BCT and the optimum, while MM has a slightly higher packet loss than BCT.

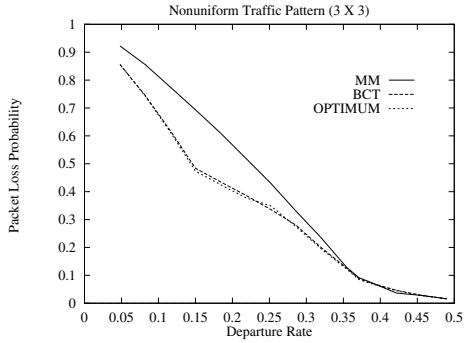

Fig. 7. The figure compares the performances of the optimum strategy, BCT and MM for different departure rates (μ) in a 3×3 switch with $B = 5$ and the following arrival rate patterns: $\lambda_{11} = 9\lambda, \lambda_{ij} = \lambda, (i,j) \neq (1,1)$, $\lambda = \frac{1-\mu}{17}$. The packet losses are almost identical for BCT and the optimum, while MM has substantially higher packet loss than BCT.

Average packet loss experienced by this policy can be computed using MDP based techniques [13]. We present the numerical performance evaluations for a 3×3 switch. We first compare the performance of BCT with the optimal. However, computation of the packet loss experienced by the optimal using MDP involves several iterations, each involving B^{18} operations for buffer size B(refer to the concluding discussions of Section II). We could thus compare the performances for small values of B ($B = 5$) only. First consider the case of uniform traffic $\lambda_{ij} = \lambda, \forall i, j, \lambda = \frac{1-\mu}{9}$ for scaling purposes. Figure 6 shows the performance curves. Now consider a nonuniform traffic pattern where the arrival rate in one queue is significantly higher than others. Specifically, $\lambda_{11} = 9\lambda$, $\lambda_{ij} = \lambda$ for $(i,j) \neq (1,1)$ and $\lambda = (1-\mu)/17$ (the last relation scales the sum of the arrival and departure rates to 1). Figures 6 and 7 show that in both cases the packet loss of BCT is close to that of the optimum for all departure rates.

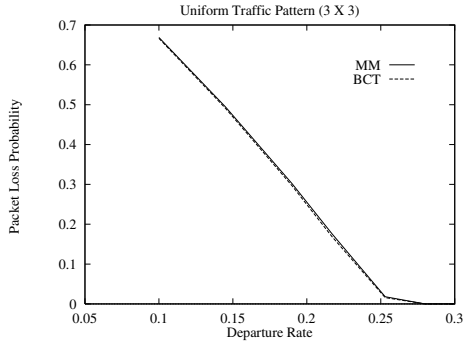

Fig. 8. The figure plots the packet loss probabilities of BCT and MM for different departure rates (μ), and equal arrival rates $\lambda_{ij} = \frac{1-\mu}{9}, \forall i,j$ in a 3×3 switch with $B = 50$. MM has a slightly higher packet loss than BCT.

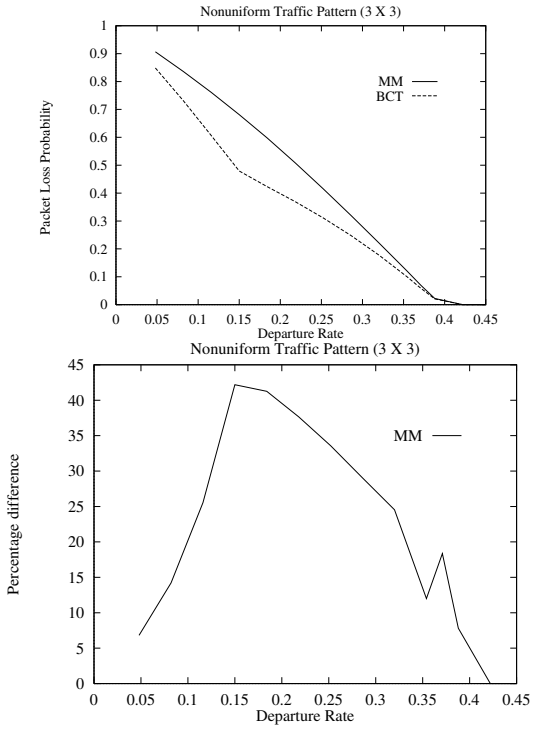

Fig. 9. The figures compare the performances of BCT and MM for different departure rates (μ) in a 3×3 switch with $B = 50$ and the following arrival rate patterns: $\lambda_{11} = 9\lambda, \lambda_{ij} = \lambda, (i,j) \neq (1,1), \lambda = \frac{1-\mu}{17}$. Figure (a) plots the absolute values of the packet loss probabilities (l_{MM}, l_{BCT}). Figure (b) plots the relative increase of the packet loss probability of MM w.r.t. BCT ($100 * l_{\text{MM}}/l_{\text{BCT}} - 1$).

MM performs inferior to BCT in both cases (the discrepancy is substantially higher for nonuniform traffic case). Investigations for other nonuniform traffic patterns indicate the same trends.

Buffer size of only 5 units may be small for drawing reliable conclusions. Thus we compare the performance of both BCT and MM via simulation for larger B, $B = 50$. Note that MDP based computations need to consider 50^{18} operations in each iteration, and are thus computationally infeasible. Thus we do not know the optimal packet loss rates. Nevertheless, simulation results demonstrate the improvement

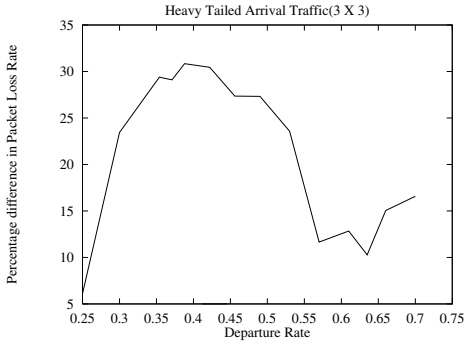

Fig. 10. The figure compares the performances of BCT and MM for different departure rates (μ) in a 3×3 switch for heavy tailed arrival distribution with $B = 500$. Each arrival brings a Pareto distributed burst of packets, mean burst size 3, $\alpha = 1.24$. Arrival epochs are poisson with rates: $\lambda_{11} = 9\lambda, \lambda_{ij} = \lambda, (i,j) \neq (1,1), \lambda = \frac{1-\mu}{17}$. The figure plots the relative increase of the packet loss probability of MM w.r.t. BCT ($100 * l_{\text{MM}}/l_{\text{BCT}} - 1$). Refer to the technical report for absolute values of the packet losses.

obtained by BCT over MM. In the uniform traffic case, MM is slightly inferior to BCT (Figure 8), while in the nonuniform traffic case the difference is substantial (Figure 9). BCT decreases the packet loss rate by more than 40% over MM (Figure 9(b)) in certain cases! Once again, we observe similar differences in performances between BCT and MM for other nonuniform traffic patterns we considered. For instance, Figure 10 shows the performance difference for bursty Poisson arrivals, with the bursts having a Pareto distribution ($F(x) \leq 1 - (b/x)^{\alpha}, b = 0.6, \alpha = 1.24$). The absolute values of the packet losses can be found in [13]. Like in the 2×2 case, the load balancing brought about by the memory management of BCT safeguards against the detrimental effect of the difference in arrival rates, while MM has no such protection, and this explains the difference in performance for nonuniform traffic patterns. We conclude that BCT is more robust than MM.

V. Discussion and Future Research Issues

We mention potential approaches for addressing several implementation related challenges of the proposed algorithms, in context of the switching and wireless applications. We hope that this paper will entice further research required to "nail down these issues."

Real time traffic demands delay guarantees, and thus loss minimization may not be enough. Packet deadline satisfaction can be facilitated in the proposed design if the oldest packet is discarded (or replaced by a new packet) from a stream during pushout (note that the push out strategy decides the queue for push out and any packet can be discarded from the chosen queue). Another approach is to deviate from the loss optimal scheduling and serve a packet if it is close to deadline expiry.

Computing the desired matching for every scheduling decision may become computationally intensive particularly for high speed operations. Further research is required for designing a computationally simple scheduling which attains low packet loss. A possible direction is to proceed along the results obtained by Mekkittikul et al. [8] and Tassiulas [14] in

simplifying the throughput optimal scheduling strategies for switches with infinite buffers.

The proposed memory management uses packet push-out which may become difficult from an implementation point of view, particularly for optical switches. An interesting topic for future research is to design a memory management strategy which optimizes packet loss under the constraint that existing packets can not be discarded. In line with the results obtained for similar objective in shared memory switches [5], the optimal strategy is likely to accept packets selectively even if the input buffer has space. We believe that a threshold based packet acceptance policy will be required.

Our objective has been to minimize packet loss. We do not address fairness issues in this paper. For example, in Figure III the proposed strategy may starve the queue x_{12} if the queue x_{21} has no packet, and queues x_{11}, x_{22} are heavily loaded. The fairness properties can be improved by forcing the scheduler to serve a queue if it has not received service for certain time, even at the cost of optimality. The time interval can be tuned to attain the desired tradeoff between fairness and optimality. Another possibility is to proceed in the direction of [9].

The scheduling strategy needs shared scheduling states in an input queued switch, and similarly a centralized coordination for application in wireless networks. Further research is needed to develop distributed scheduling strategies, and we believe that developing a centralized policy is the first step in that direction.

REFERENCES

[1] M. Arpaci and J. Copeland. Buffer-management of shared memory atm switches. *IEEE Communications Surveys*, 2000.

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

[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] 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.

[6] M. Karol, M. Hluchyj, and S. Morgan. Input versus output queueing on a space-division switch. *IEEE Transactions on Communications*, 35, December 1987.

[7] N. Mckeown, A. Mekkittikul, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input queued switch. *IEEE Transactions on Communications*, 47(8), 1999.

[8] A. Mekkittikul and N. Mckeown. A practical scheduling algorithm to achieve 100% throughput in input-queued switches. *Proceedings of INFOCOM'1998*, April 1998.

[9] N. Ni and L. Bhuyan. Fair scheduling and buffer management in internet routers. *Proceedings of INFOCOM, NY*, June 2002.

[10] M. Putterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1998.

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

[12] R. Roy and S. Panwar. Optimal space priority policies for shared memory atm systems. *Proceedings of Allerton Conference on Communication, Control, and Computing*, October 1997.

[13] S. Sarkar. A scheme for jointly optimum scheduling and memory management with application to switches and wireless networks. *Technical Report: Available at http://www.seas.upenn.edu/∼swati/publication.htm*, 2002.

[14] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switchs. *Proceedings of INFOCOM'1998*, April 1998.

[15] 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.

[16] T. Weller and B. Hajek. Scheduling nonuniform traffic in a packet switching system with small propagation delay. *Proceedings of INFOCOM'1994*, 1994.

[17] M. Yoo, C. Qiao, and S. Dixit. Qos performance of optical burst switching in ip-over-wdm networks. *IEEE Journal on Selected Areas in Communications*, 18(10), October 2000.