Deadline Aware Packet Scheduling in Switches for Multimedia Streaming Applications

Praveen Bommannavar
Management Science and Engineering
Stanford University
Stanford, CA 94305 USA
bommanna@stanford.edu

John Apostolopoulos
Hewlett-Packard Labs
Palo Alto, CA 94304 USA
john_apostolopoulos@hp.com

Nicholas Bambos
Electrical Engineering and
Management Science and Engineering
Stanford University
Stanford, CA 94305 USA
bambos@stanford.edu

***Abstract*—We consider the problem of scheduling packets in an input queued switch with a focus on processing streamed multimedia data. In such applications, packets arrive with hard service deadlines; after the deadline for a packet has passed, it is no longer useful and does not get delivered - it is dropped. We seek policies to minimize the number of late packets, which are then dropped. The problem is formulated in a Dynamic Programming framework and shown to be intractable. The formulation is contrasted to the related crossbar switch scheduling problem, with an emphasis on the fact that we have a different objective function. A simplified probabilistic version of the streaming problem is used as motivation for a heuristic solution. Finally, we present results from a simulation in which a simple heuristic based on weighting queues according to the deadline of the leading packet consistently outperforms the well-known maximum weight matching (MWM) algorithm.**

***Index Terms*—scheduling; deadlines; crossbar switch; input-queued switch; maximum weight matching**

I. Introduction

In this paper we consider the problem of moving data packets from a set of input ports to a set of output ports with the special provision that packets carry a deadline for delivery. Such a constraint appears in applications where the data are processed in a real-time streaming fashion; instead of incurring additional delay beyond a specified deadline, it is preferable to drop the late packet and move on to the next one. This appears frequently in the context of Voice over IP (VoIP), Video over Broadband (VoBB), Video on Demand (VoD) and a host of other applications in which delay beyond a specified threshold is no longer acceptable.

Results for queuing problems involving packet deadlines appear frequently in the multimedia streaming and wireless communications literature. In [1], the problem of scheduling multiple queues in which streaming packets have deadlines over a wireless channel is considered, but there are not multiple output queues. The analysis in [2] examines policies that are simultaneously aware of channel strength, packet deadlines and distortion and [3] looks at scheduling packets at a base station to maximize quality of service to downlink users where deadlines play a role in the quality. Policies for multicast routing with similar loss criteria and packet constraints are studied in [4]. Packet deadlines as well as packet values are considered in [5] for a single queue, and bounds on the competitive ratio are derived.

The applicability of deadline aware scheduling algorithms reaches as far as grid computation, in which threads much be adaptively scheduled to optimize distributed computation [6].

This literature, while rich, often does not involve constraints entangling service combinations for the input queues. Entanglement typically appears in the form of a matching constraint on input and output ports, as in the crossbar switch, and is a natural limitation in practical scenarios. Indeed, input queued switches are fundamental to the design of Internet routers due to underlying constraints in the silicon based mechanism for which traffic streams can be simultaneously transmitted.

The importance of the input queued switch architecture has resulted in a rich literature on processing queuing networks. The main result of [7] shows that the intuitive and effective scheduling MWM policy has desirable stability properties. [8] gives general results for stability and maximal throughput for the processing of parallel queues using cone scheduling polices. Other works give algorithms that aim to minimize the average delay experienced by a packet in the system. More recently, results in [9] show that one can obtain strong bounds on queue lengths under a policy derived by emulating an insensitive bandwidth sharing network with a product-form stationary distribution.

In this paper we model the crossbar switch problem in the context of multimedia streaming, where packet deadlines are an important modeling constraint. This modification to the classical input queued switch, while subtle, changes the problem significantly. We motivate a heuristic scheduling policy, called deadline-aware maximum weight matching (MWM), which is shown in simulations to outperform the standard queue length MWM algorithm.

In Section II we present the deadline constrained crossbar switch model under consideration along with a dynamic programming formulation of the problem. Section III reviews the standard crossbar switch and the associated queue length MWM-α policy. We then continue in Section IV with a simplified probabilistic model whose solution we use to motivate a deadline aware heuristic, which is presented in Section V. Section VI presents simulation results which reveal that this deadline aware heuristic outperforms queue length MWM+ in our modified problem space. Finally, concluding remarks and future directions for work are offered in Section VII.

II. Problem Formulation

We give a mathematical model for the delay sensitive crossbar switch problem, along with a dynamic programming (DP) formulation for the solution, which quickly becomes intractable. Approximate descriptions for the state of the system are proposed.

A. Model

We consider the $n \times n$ input-queued crossbar switch, which is used to route traffic from sources to destinations in a setting where each of n sources are composed of packets which are headed for n different destinations. Each packet additionally has a timestamp indicating the number of time steps after which the packet expires and cost is accrued. Fig. 1 depicts a schematic diagram of a 3×3 switch. At each time step t an allocation decision must be made to connect input ports to output ports, subject to a matching condition: each (input or output) port can only be connected to at most one other port of a different type. That is, no input port can have two connections, nor can any output port. The decision, once made, transports a packet for each connected input-output pair (i, j) if a packet is waiting at input i for port j. We then continue to the next time step $t + 1$ and packets that had deadlines for t expire, accruing cost (performance loss).

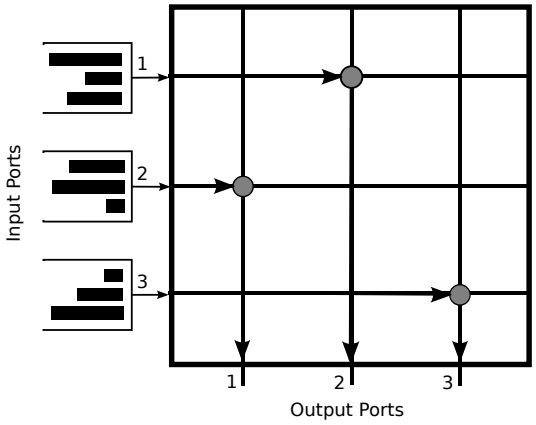

Fig. 1. A 3×3 crossbar switch. Each packet has an associated deadline.

The switch makes a matching decision at each discrete time step $t = 0, 1, \ldots$. We now introduce a bit of notation. Let $\mathcal{B}$ be the set of buffers:

$$\mathcal{B} = \{(i, j) | 1 \leq i, j \leq n\}$$

and let $u \in \{0, 1\}^{n^2}$ be a $0-1$ vector of allocation decisions. That is, $u_{ij} = 1$ if a connection is between input port i and output port j. As per our discussion above, we have the constraints

$$\sum_{j=1}^{n} u_{ij} \leq 1 \qquad 1 \leq i \leq n$$

$$\sum_{i=1}^{n} u_{ij} \leq 1 \qquad 1 \leq j \leq n$$

to enforce the matching condition. We denote the set of all allocations that satisfy these constraints by the set $\mathcal{U}$.

Packet arrivals are modeled as independent Bernoulli processes with the probability of arrival for each queue (i, j) equal to λ_{ij}. Service times are deterministic, taking place over a single time step, and the absolute deadline for a packet arriving at queue (i, j) at time t is given by $t+\delta_{ij}$ where δ_{ij} represents the amount of time an arriving packet must be processed in. That is, the allotted time for a packet to be processed depends deterministically on the input and output ports.

The state of the system can be summarized by a set V of arrays $\mathbf{v_{ij}}$ containing the sequence of deadlines for each queue (i, j). Next we let $q_{ij}(t) \in \mathbf{R}^{n^2}$ be the length of buffer (i, j) at time t after u_{ij} is applied and $d_{ij}(t) \in \mathbf{R}^{n^2}$ be the relative deadline of the leading packet of queue (i, j) at time t after the application of u_{ij}. That is, $d_{ij}(t)$ represents the number of time slots remaining for the leading packet to be processed in.

An admissible policy is a mapping $\pi : V \rightarrow u$ such that $u \in \mathcal{U}$. We aim to find an admissible policy π^* to minimize the expected number of dropped packets per time step:

$$J_\pi(V) = \limsup_{T \rightarrow \infty} \mathbf{E}_\pi \left[\frac{1}{T} \sum_{t=0}^{T-1} \sum_{i,j} \mathbf{1}_{d_{ij}(t)=0} \right]$$

where the expectation is under the policy π. Note that this is equivalent to minimizing the probability of a packet being dropped.

B. Dynamic Programming

Our problem formulation fits into a finite state dynamic programming framework. To see this, consider the optimal cost-to-go function $J_{\pi^*} = \lambda^* + h(V)^*$ which satisfies Bellman's equation [10]:

$$\lambda^* + h(V)^* = \sum_{ij} \mathbf{1}_{d_{ij}(t)=0} + \min_u \left[\sum_{V'} P_u(V \rightarrow V') h(V')^* \right] \tag{1}$$

where $P_u(V \rightarrow V')$ is the probability of transitioning from state V to V' given that allocation decision u is made. This probability takes into account the packets that are served due to allocation u as well as incoming packets, whose arrivals are governed by λ_{ij}.

We note that the set of all possible states $\mathcal{V}$ is finite. The reason for this is that any given queue receives at most one packet per time step, and packets expire after a fixed amount of time. Therefore, the largest that queue (i, j) can be is δ_{ij}. For any queue of length bounded by δ_{ij}, there is a fixed number of possible arrangements of deadlines comprising the queue $(2^{d_{ij}})$.

Despite this, however, the problem as stated becomes difficult for anything but the smallest instances. There are two culprits for the intractability of Bellman's equation:

1) The size of the state space, $|\mathcal{V}|$, grows exponentially with the number of queues and the lengths of the deadlines.

We have $|\mathcal{V}| = \prod_{i=1}^{n} \prod_{j=1}^{n} 2^{\delta_{ij}}$, since there could be at most one packet corresponding to each possible deadline in the queue. This results in an intractable number of cost-to-go parameters to store and a huge number of states to sum over in (1).

2) The number of possible actions at each time step, $|\mathcal{U}|$, also grows exponentially in n. A simple counting argument shows that $|\mathcal{U}| = n!$, which is an unwieldy number of actions to choose among in the minimization step of (1).

Approximations can be done to somewhat reduce the number of states:

1) Rather than enumerating states according to the deadline of each packet in each queue, we can look at the deadline of the leading packet, $d_{ij}(t)$ and the length of each queue $q_{ij}(t)$.
2) Use a surrogate value function which approximates the quantities h^* based on the reduced state description.

However, it is not immediately clear how to define the transitions between these reduced states, nor how to minimize over the $n!$ possible allocations at each time step.

III. Maximum Weight Matching

In light of the difficulties inherent in solving our problem exactly, we turn to heuristic solutions. In particular, we focus on a class of heuristic policies that has been widely studied in the literature for a variant of our problem, which is based on maximum weight matching [11]. These policies are meant for a formulation of our problem *without* packet deadlines and a different objective function. Consider minimizing the long term expected delay for packets to be served:

$$\limsup_{T\to\infty} \mathbf{E}_\pi \left[\frac{1}{T} \sum_{t=0}^{T-1} \sum_{i,j} q_{ij}(t) \right]$$

In this modified formulation, we can only consider instances in which

$$\sum_{j=1}^{n} \lambda_{ij} \leq 1 \qquad 1 \leq i \leq n \tag{2}$$

$$\sum_{i=1}^{n} \lambda_{ij} \leq 1 \qquad 1 \leq j \leq n \tag{3}$$

Without this provision, stability of the system is not possible, since any particular i or j violating this condition would not be able to serve packets fast enough since only one packet from an input queue and only one packet from an output queue can be processed at a time.

The proposed solution in this setting is to construct a weight function W which maps a queue's length to a real number. An allocation is chosen by solving the optimization problem

$$\max_{u\in\mathcal{U}} \left[\sum_{i,j} W(q_{ij}) u_{ij} \right] \tag{4}$$

at each time t and choosing the optimal allocation u. The structure of the constraints makes this optimization straightforward and efficient using well known algorithms for maximum weight matchings on bipartite graphs. Fig. 2 shows a representation of input and output ports as a bipartite graph in which a matching determines which queues (i, j) are processed. Even though there are $n!$ matchings to choose from, the optimal matching can be found in polynomial time ($\mathcal{O}(n^3)$).

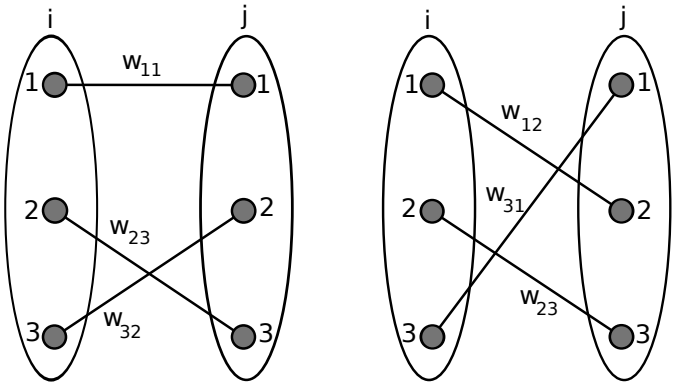

Fig. 2. Two of the six possible matchings on the bipartite graph. An algorithm based on augmenting paths yields a tractable method to find the matching of maximum weight.

The only remaining question is how to choose the weight function W. MWM-α is a class of policies that take a scalar $\alpha \geq 0$ as a parameter. The MWM-α policy uses weight function

$$W(q_{ij}) = q_{ij}^{\alpha} \mathbf{1}_{\{q_{ij}>0\}}$$

where $\mathbf{1}$ is the indicator function. Different values for α result in different heuristics. If $\alpha = 0$, the policy take a greedy approach to maximize the number of packets being served at each time step. Larger values of α result in a preference for queues with longer backlogs.

One of the main results in the literature is that for $\alpha \in (0, \infty)$, the MWM-α policy is *throughput optimal*, that is, for arrival rates satisfying (2) and (3), MWM-α is stable. Also, as $\alpha \to 0^+$, the average delay is reduced [12]. Although the policy with $\alpha = 0$ is unstable, since it does not specify how to break ties in the cases where multiple allocations are able to serve the same number of packets, we can consider a sequence of policies that approach $\alpha = 0$ from above [13]. This is called the MWM+ policy, which makes use of the fact that for α small,

$$W(q_{ij}) \approx (1 + \alpha \log(q_{ij})) \mathbf{1}_{\{q_{ij}>0\}}. \tag{5}$$

From this point, we shall refer to this choice as 'queue length MWM+'. Intuitively, this policy seeks to process the maximum number of packets possible, breaking ties in favor of longer queues. We interpret these as first order (packet processing) and second order (length of queue) effects. MWM+ was conjectured to be optimal in the heavy traffic regime, but this has since been shown not to be the case [14]. A variant of this

uses

$$W \approx (1 + \alpha \log(a_{ij}))\mathbf{1}_{\{q_{ij}>0\}}. \quad (6)$$

where a_{ij} is the time since the leading packet arrived in queue ij and will be referred to as 'oldest cell first MWM+'. It is worth noting that these policies do not take into account the arrival rates for queues.

IV. Geometric Deadlines

In order to further motivate a heuristic for our original problem *with* deadlines, we consider a modification of our problem statement. Rather than considering packets with hard deterministic deadlines, we 'soften' these deadlines by supposing that packets arrive with probabilistic clocks. Specifically, packets in each queue (i, j) expire after a duration of time dictated by a geometric random variable p_{ij} with support $\{1,2,\dots\}$. We further suppose that the clock on a packet doesn't start until it is at the head of its queue. Also, an empty queue in our setting has a clock with $p_{ij} = 0$.

This formulation attempts to mimic our original problem state in Section II by taking into account that packets expire in order of their arrival into the queue, and that different queues are characterized by different packet expiration times. In order to maintain stability of the system, we suppose that

$$\frac{1}{p_{ij}} > \lambda_{ij}$$

so that no one queue 'blows up'. We retain all other aspects of the problem detailed in Section II, such as admissible policies and cost structure. Fig. 3 gives a pictorial representation of the new formulation.

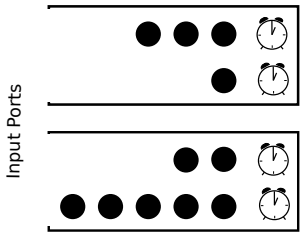

Fig. 3. Each packet has a clock which does not start until it reaches the front of its queue.

An optimal policy for this formulation is to greedily minimize the expected number of lost packets at each time step. This is equivalent to minimizing the long run packet loss, because packets are only lost when they are at the front of their respective queues. Therefore, we turn our attention to the problem of choosing the matching at each step which serves the packets mostly likely to be dropped in the next slot. We accomplish this with the same maximum weight matching algorithm as in the previous section.

The allocation policy chosen by maximizing (4) and using weight function

$$W(q_{ij}) = p_{ij}\mathbf{1}_{\{q_{ij}>0\}} \quad (7)$$

minimizes the expected number of packets that are lost by the next time step. The expected number of lost packets at each time step is

$$L = \sum_{ij}(1 - u_{ij})p_{ij}\mathbf{1}_{\{q_{ij}>0\}}.$$

V. Deadline Aware MWM

Let us now return to our original problem formulation and use insights from the scenarios in Section III and Section IV to construct a heuristic to minimize packet loss in the hard deadline case. Our approach will be to use the leading packets of queues for first order weighting effects and queue lengths as a secondary tie breaker, similar to MWM+ in Section III. Using the geometric packet drop probabilities for inspiration, we devise an appropriate weight function for our problem instance.

A. Weight Function

We propose a policy that maximizes (4) using weight function

$$W(q_{ij}, d_{ij}) = \left(\frac{1}{1 + d_{ij} - t} + \alpha \log(q_{ij})\right)\mathbf{1}_{\{q_{ij}>0\}} \quad (8)$$

and sufficiently small α. Note that this function preserves the form of MWM+ (5) with the modification motivated by optimal weights in (6). Indeed, the fraction $\frac{1}{1+d_{ij}(t)-t}$ is similar to $p_{ij}(t)$ since $1/p_{ij}(t)$ is the expected deadline of a packet in the formulation of the previous section. Henceforth, we shall refer to this as *deadline-aware* MWM+.

B. Intuition

Intuitively, all three weight functions (5), (6) and (7) attempt to 'align' an allocation to the stress placed on the input queues. Consider the backlogs of the queues in Fig. 4 in a scenario which aims to minimize expected delay (no deadlines). Without knowing the exact weight function values, one can see that the MWM algorithm tends towards choosing the matching which best relieves the stress to the overloaded buffers.

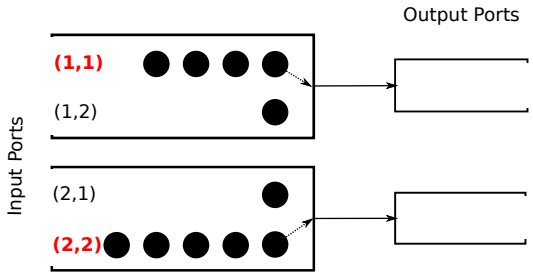

Fig. 4. Input queues (1,1) and (2,2) are the most stressed, so the matching allowing these queues to be processed is chosen.

This intuition is also present in our deadline-aware MWM+ heuristic, except that in our case, the system is 'stressed' when packets approach their deadlines.

VI. Simulation Results

We now evaluate the performance of using weight function (7) on small problems and compare the performance to the baseline of MWM+ (5). Note that as a first pass, it is reasonable to try to solve the minimum packet loss problem using a delay minimizing approach. The numerical experiments that follow, however, show that by modifying the weight function to better capture the packet loss minimizing criterion, we can obtain stronger performance.

We consider a 4×4 switch and plot the performance of the benchmark queue length MWM+ policy alongside that of deadline-aware MWM+ for a range of traffic conditions. First, we let the deadlines and arrival rates be symmetric across all input queues. Then we look at two cases in which some input queues have different arrival rates as well as deadlines than other queues.

In a forthcoming journal paper we give numerical results for larger switches and more deeply examine loss rates for each input-output pair.

A. 4×4 Switch with Symmetric Deadlines & Arrivals

In this experiment, a long run average (over 10^6 time steps) rate of packet loss was computed for the case where all 16 input queues have the same arrival rates and deadlines. The rate of packet loss was computed for low, medium and heavy traffic regimes.

Let the arrival rates be represented by $\Lambda = [\lambda_{ij}] \in \mathbf{R}^{n \times n}$ and the deadlines be represented by $D = [\delta_{ij}] \in {\mathbf{Z}_+}^{n \times n}$. These parameters were chosen for this simulation as

$$\Lambda = \frac{\lambda}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad D = \begin{pmatrix} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{pmatrix}$$

where the parameter λ is allowed to range from $0 \rightarrow 1$. This encompasses the range from no traffic ($\lambda = 0$) to heavy traffic ($\lambda = 1$). When $\lambda = 1$, each input and output port is at capacity for arrivals. That is, $\sum_{j=1}^{n} \lambda_{ij} = 1$ and $\sum_{i=1}^{n} \lambda_{ij} = 1$. Beyond this, more traffic is arriving to the system than can feasibly be processed. The resulting curve is given in Fig. 5.

B. 4×4 Switch (Asymmetric Deadlines & Arrivals: Case 1)

We now modify the parameters Λ and D to examine the performance of each algorithm when arrival rates are not uniform across input queues, and more frequent arrivals correspond to longer deadlines:

$$\Lambda = \lambda \begin{bmatrix} 0.5 & 0.25 & 0.15 & 0.1 \\ 0.25 & 0.5 & 0.1 & 0.15 \\ 0.15 & 0.1 & 0.5 & 0.25 \\ 0.1 & 0.15 & 0.25 & 0.5 \end{bmatrix} D = \begin{pmatrix} 8 & 6 & 4 & 2 \\ 6 & 8 & 2 & 4 \\ 4 & 2 & 8 & 6 \\ 2 & 4 & 6 & 8 \end{pmatrix}$$

Fig. 6 shows the curve corresponding to this modified scenario.

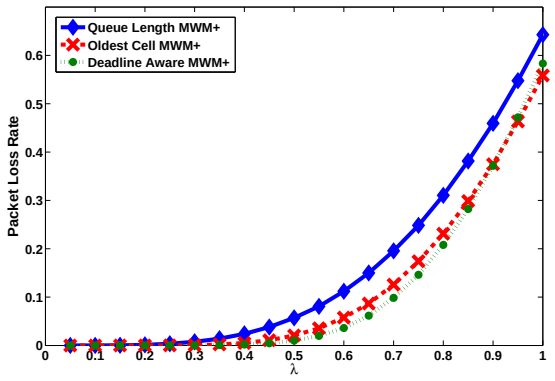

Fig. 5. Packet loss rate for 4×4 switch with symmetric arrival rates and deadlines.

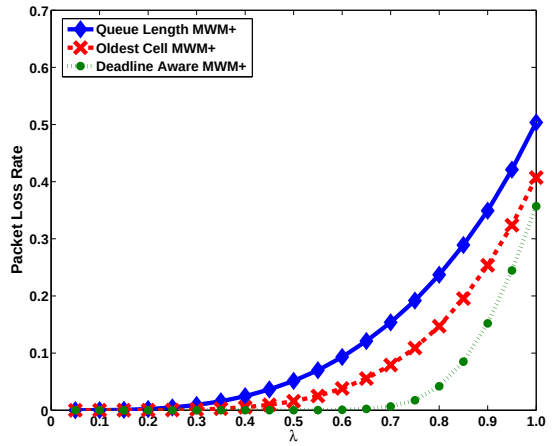

Fig. 6. Packet loss rate for 4×4 switch with greater arrival rates for longer deadlines.

C. 4×4 Switch (Asymmetric Deadlines & Arrivals Case 2)

Now let us look at a scenario with the reverse type of arrival-deadline skewness: greater arrival rates correspond to shorter deadlines. We simply use the same arrival rates as the previous case, but flip the assignment of deadlines.

$$\Lambda = \lambda \begin{bmatrix} 0.5 & 0.25 & 0.15 & 0.1 \\ 0.25 & 0.5 & 0.1 & 0.15 \\ 0.15 & 0.1 & 0.5 & 0.25 \\ 0.1 & 0.15 & 0.25 & 0.5 \end{bmatrix} D = \begin{pmatrix} 2 & 4 & 6 & 8 \\ 4 & 2 & 8 & 6 \\ 6 & 8 & 2 & 4 \\ 8 & 6 & 4 & 2 \end{pmatrix}$$

The plot for this scenario is given in Fig. 7.

D. Discussion

The first observation that is immediately seen from these plots is that deadline-aware MWM+ consistently outperforms the venerable queue length MWM+ algorithms. This is true at every range of values studied except large values of λ in the symmetric case, but this region of traffic is hardly practical since a disproportionate number of packets are lost no matter what policy is used.

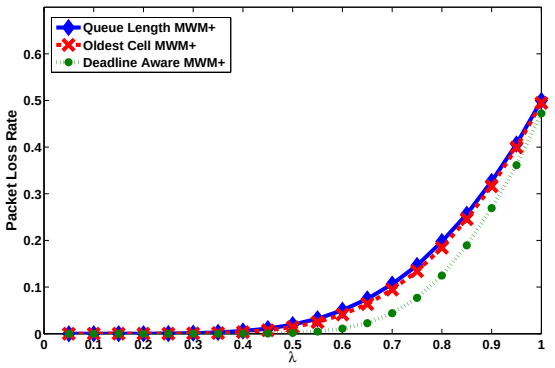

Fig. 7. Packet loss rate for 4×4 switch with greater arrival rates for shorter deadlines.

Indeed, although minimizing expected packet delay using queue length MWM+ or oldest cell first MWM+ achieves reasonable loss rate, the modifications we have made improve it significantly in the medium traffic region *in some cases by as much 300%*. In low traffic, there are so few arrivals that all three policies are able to process most packets. In high traffic, all policies struggle to keep up with the combined pressure of packet deadlines as well as rapid arrivals.

Comparing the two asymmetric arrivals and deadlines plots, it is also apparent that deadline-aware MWM+ makes the greatest gains over queue length MWM+ and oldest cell first MWM+ when longer deadlines have greater arrival rates. This is perhaps because weighting by queue length, in this case, loses many packets which are about to expire but are in shorter queues. On the other hand, when shorter deadlines have greater arrival rates, the performance gap is tighter. This is because we expect longer queues to have the shorter deadline in many cases, and so the policies somewhat coincide. There are still significant gains over both queue length MWM+ and oldest cell first MWM+, however.

Although the deadlines chosen in these examples are relatively small values, true implementations may have a long series of switches carrying traffic from many sources to destinations, and hence the delays could sum to result in significant quality loss.

Finally, we add that although it cannot be seen in the plots, greater values of λ cause all curves to tend towards the same (high) packet loss rates. We omit this part of the graph because it corresponds to values of λ which call for more arrivals than can be processed by *any* policy.

VII. Conclusion and Future Work

In this paper we have presented a variation of the input-queued crossbar switch in which arriving packets carry a processing deadline. We search for a policy that can minimize the long run average number of dropped packets per time step. This requirement that packets are processed before a certain time is highly relevant in real time streaming applications. The maximum weight matching (MWM) policies, while reasonable, are shown to be suboptimal. Taking the intuition behind a simpler scenario in which deadlines are probabilistic (geometrically distributed) and fusing it with the structure of MWM+, we construct a deadline based heuristic, which we term deadline-aware MWM+. Long run simulations show our heuristic drops fewer packets (on average) than queue length MWM+ and oldest cell first MWM+, and the difference is most pronounced in the medium traffic region.

Returning to our original motivation to study this problem, we note that there are a multitude of directions in which to advance this work. In the context of the deadline constrained crossbar switch, it is of interest to find theoretical guarantees of the performance of these heuristics. A bound on the deviation of a heuristic from the optimal performance is a particularly appealing result to pursue. Such bounds could be on the average packet drop rate, but other meaningful metrics could also be used, such as competitive ratio. Studying stochastic networks of these switches may also prove to be a fruitful research agenda.

References

[1] S. Shakkottai and R. Srikant, Scheduling real-time traffic with deadlines over a wireless channel, *Wireless Networks*, vol. 8, no. 1, pp. 13-26, Jan. 2002

[2] A. Dua, C. W. Chan, N. Bambos, J. Apostolopoulos, Channel, deadline, and distortion aware scheduling of video streams over wireless, *IEEE Trans. on Wireless Communications*, 2010.

[3] A. Dua and N. Bambos, Downlink wireless packet scheduling with deadlines, *IEEE Trans. Mobile Computing*, vol. 6, no. 12, pp. 1410-1425, December 2007.

[4] V. Raghunathan, V. Borkar, M. Cao, and P. R. Kumar, Index policies for real-time multicast scheduling for wireless broadcast systems, in *IEEE INFOCOM*, Phoenix, AZ, USA, pp. 1570-1578, 2008,.

[5] B. Hajek. On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time, *Conference on Information Sciences and Systems*, Johns Hopkins University, March 21-23, 2001, pp. 434-439.

[6] X. Zhao and N. Jamali, Supporting Deadline Constrained Distributed Computations on Grids, *IEEE Internation Conference on Grid Computing*, pp. 165-172, September 2011.

[7] L. Tassiulas and A. Ephremides, Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936-1948, 1992.

[8] K. Ross and N. Bambos, Projective Cone Scheduling (PCS) Algorithms for Packet Switches of Maximal Throughput, In *IEEE/ACM Transactions on Networking*. 17(3): 976-989, 2009.

[9] D. Shah, N. Walton, and Y. Zhong, Optimal queue-size scaling in switched networks, submitted, 2011.

[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientic, 1995.

[11] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. In *Proceedings of IEEE INFOCOM*, pp. 296-302, San Francisco, CA, March 1996

[12] I. Keslassy and N. McKeown. Analysis of scheduling algorithms that provide 100% throughput in input-queued switches. In *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October 2001

[13] I. Keslassy, R. Zhang-Shen, and N. McKeown. Maximum size matching is unstable for any packet switch. *IEEE Communications Letters*, 7(10):496-498, October 2003

[14] T. Ji, E. Athanasopoulou, and R. Srikant. Counter-examples to the optimality of mws-alpha policies for scheduling in generalized switches. preprint, 2009