PRIORITIZED ARBITRATION FOR INPUT-QUEUED SWITCHES WITH 100% THROUGHPUT

Rainer Schoenen and Guido Post and Gerald Sander
Institute for Integrated Signal Processing Systems, Aachen University of Technology
Templergraben 55, 52056-Aachen, Germany
schoenen@ert.rwth-aachen.de, http://www.iss.rwth-aachen.de/

Abstract

Input buffered switches are known to suffer from head-of-line (HOL) blocking that limits the throughput to approximately 58.6%. It has been shown that 100% throughput can be achieved with virtual output queueing (VOQ) and an arbitration algorithm that controls the access to the switch fabric in each time slot. No weight-dependent algorithm was known that supports priorities, which are necessary for the isolation of connections with different QoS requirements in ATM or IPv6 switches. In this paper we show how a weighted matching can be extended to support priorities in order to separate CBR, VBR and ABR traffic. The number of bits per priority is exposed as the main parameter. Based on simulation results the priority performance is evaluated for the ideal algorithm and an approximation SIMP which offers a delay performance close to the ideal maximum weight matching.

1 Introduction

Broadband switches for ATM and IPv6 are needed to build the future backbone networks. Among the switching architectures input queued switches are the most powerful because the access rate of crossbar and buffer memory is not higher than the line rate of the connected link. For the classical FIFO queue organization it is known that due to head-of-line blocking the maximum throughput is approximately 58.6% [1]. A different organization can avoid blocking by bypassing cells destined for free output ports. Thus Virtual Output Queueing (VOQ) [2] is mandatory where each input manages a separate queue for each output (fig. 1). It has been shown that a throughput of 100% can then be achieved [3, 4, 5].

Arbitration algorithms are used to control the access of queues to the switch fabric by resolving the contention for the same output ports in each time slot. The achievable throughput and delay performance depends on the arbitration algorithm. Weighted algorithms [4] offer a much better delay performance than unweighted or single-iteration distributed algorithms [6]. The recent literature offers some attractive methods, but the problem of priorities has not been treated sufficiently. In fact either weighted single-priority [2] or unweighted prioritized [7] algorithms exist.

For providing Quality-of-Service, as required for ATM or proposed for IPv6, it is unavoidable to enforce a priority of real-time over best effort connections [8]. A static priority scheme must be established, where lower priority (data) traffic has no noticable influence on a higher priority (real-time) traffic.

In this paper we extend the concept of weighted matching to support priorities without increasing the complexity of an arbitration algorithm. The idea is a weighted summation of partial weights that are held per priority. The partial weights are mapped by several functions which we discuss and compare by simulation. The number of bits used for the partial weights is exposed as the main parameter.

In addition we offer a new weighted arbitration algorithm that supports the priority concept and asymptotically achieves 100% throughput. Results are shown for uniform and non-uniform workload and bursty traffic. We show that the delay performance is comparable to the best known alternative algorithms.

The paper is organized as follows. Section 2 discusses current knowledge. Priorities are introduced in section 3. The SIMP arbitration algorithm is explained in section 4. Section 5 contains numerical results.

2 Background and Related Work

While a FIFO queue organization is commonly assumed for input-queued switches, we use the VOQ configuration [2] shown in fig. 1 that consists of M ports for input and output, a nonblocking switch fabric and an arbitration unit. Arriving

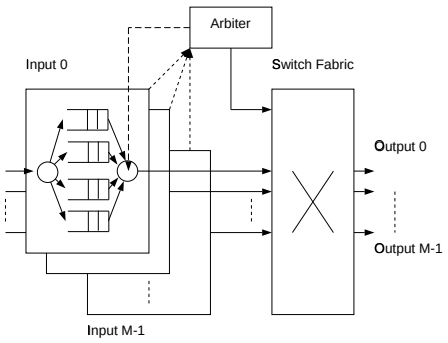

Figure 1: virtual output queueing

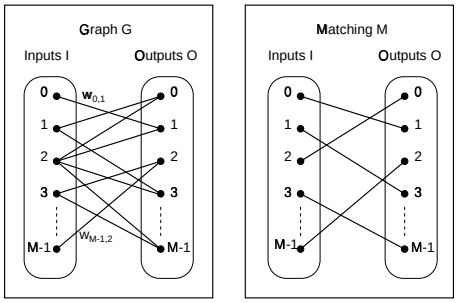

Figure 2: Bipartite graph matching

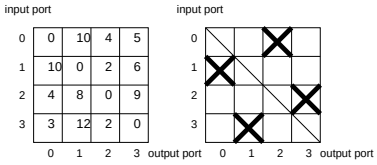

Figure 3: typical match

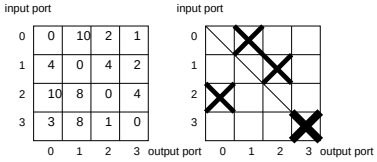

Figure 4: bad match

cells on input port i are placed into the corresponding queue for their destination port o. Its current queue size is $q_{i,o}$. The process of arrivals to this queue is characterized by a mean rate $\lambda_{i,o}$ (normalized to the line rate). The load is admissible [2] if $\forall o : \sum_{i=0}^{M-1} \lambda_{i,o} < 1$ and $\forall i : \sum_{o=0}^{M-1} \lambda_{i,o} < 1$ In each time slot the arbiter selects unique pairs of input and output ports (a "match" (i,o)) based on information sent to it from the input ports.

This task can be viewed as a bipartite graph matching problem [2] as shown in fig 2. A number of algorithms exist to solve it optimally, either as a maximum size matching (MSM) or a maximum weight matching (MWM) [9]. MSM finds a match with the maximum number of input-output connection crosspoints. MWM finds a match where the sum of the weights of the crosspoints $w_{i,o}$ are maximum. For the weight chosen as $w_{i,o} = q_{i,o}$ the algorithm is called "longest queue first" (LQF) in [4], and it has been shown that 100% throughput can be achieved for all admissible i.i.d. arrivals [4] with MWM. Without weight information, as with MSM, instability and unfairness are likely, because queue backlogs can accumulate and cell delays become very large for bursty or asymmetric load.

Algorithms for solving the MWM problem optimally are computationally complex. They take $O(M^3 logM)$ and are much too complex for a hardware implementation [5]. A number of alternatives have been proposed, such as MCFF and iMCFF [5], iLQF [3] or the analog MUCS [10]. Even the unweighted MSM problem is rated $O(M^{2.5})$. MSM approximations exist with PIM [6], iSLIP [3]. A number of implementations of unweighted matching exist, e.g. [11] or [7, 12] based on iSLIP. Complexity statements must be interpreted with care. The best iterative unweighted algorithm takes $O(M \log M)$ serial or $O(\log M)$ parallel time steps, while the hardware area takes $O(M^4)$. Priorities have been used with iSLIP and ESLIP [7, 12], but as with every unweighted algorithm the delay becomes large for bursty or asymmetric load.

Obviously an internal speedup can make the algorithms easier as shown in [13, 14, 15, 16], but this reduces the available time for switching, memory access and the arbitration algorithm. Multicast is not treated here, because it can be handled in a separate step and by using fanout splitting [7].

So far there has been no explicit treatment of priority classes with input-queued switches and weighted matching, where the goal is the independence of the performance of one traffic class from those of lower priorities [17] as well as a small cell delay.

Because of the complexity of the M^2-dimensional state space an analytical treatment is not in reach. In the literature only unweighted algorithms are treated [3, 18], or the assumptions simplify it to an OQ model [19].

3 Introducing Priorities for Weighted Arbitration Algorithms

Weighted matching algorithms can achieve full throughput under admissible load [4]. However, with different priority classes a stable operation cannot be guaranteed. In fact an admissible load of higher priority cell streams can nevertheless lead to unbounded delay if lower priority (best effort) connections experience congestion (high queue weight $w_{i,o}$). When a smaller weight on another port represents real-time cells only, they have to suffer.

The goal is to have no influence of lower priority queue weights over higher priority weights. A three-dimensional extension of the known weighted matching algorithms is discarded because of the complexity. We aim for a method operating in two dimensions only.

Consider a queue organization as shown to fig. 1 and assume the queues are additionally separated per priority (with P priority levels, 1=highest). Let the queue states on input i, output o and priority level p be $q_{i,o,p}$. The arrivals to these queues are characterized by a mean rate $\lambda_{i,o,p}$. All weighted matching algorithms operate on a two-dimensional weight

$w_{i,o}$. Therefore we map $[q_{i,o,p}]$ to $[w_{i,o}]$ in a special way. The simplest case is $w_{i,o} = \sum_{p=1}^{P} q_{i,o,p}$ (no priority support).

The key idea of the simplification proposed in this paper is to use the following mapping function (modified weighted summation) to reduce q_{iop} to a two-dimensional format that can be handled by any weight dependent arbitration algorithm.

$$w_{io} = \sum_{p=1}^{P} c_p \cdot f(q_{iop}) \tag{1}$$

$$c_p = \prod_{q=p+1}^{P-1} k_q = \prod_{q=p+1}^{P-1} 2^{b_q} \tag{2}$$

The coefficients c_p are computed in eq. 2 such that the weight of a queue with priority p is k_{p+1} times higher (b_{p+1} bits) than that of the next lower priority $p+1$. For the function $f(q_{iop})$ the alternatives are

$$f_1(q_{iop}) = q_{iop} \tag{3}$$

$$f_2(q_{iop}) = sat(q_{iop}, k_p) \tag{4}$$

$$f_3(q_{iop}) = compand(q_{iop}) \tag{5}$$

$$sat(x,s) = \{ \begin{array}{ccc} x & if & x < s \\ s-1 & if & x \geq s \end{array} \} \tag{6}$$

For example, $P = 3; b_p = 3 \Rightarrow k_p = 8; q_1 = 2 = 010_b, q_2 = 5 = 101_b, q_3 = 15 \Rightarrow sat(q_3, 8) = 111_b \Rightarrow w = 010101111_b$.

With this mechanism priorities are supported because queued cells of a higher priority are given a higher weight $w_{i,o}$. We now have the choice of a number of parameters. The number of bits per priority b_p represents the dynamic range within the class and allows scaling from MSM ($b_p = 1$) to MWM ($b_p \to \infty$). The second choice is the monotonous nonlinear scaling function f.

A sufficient priority separation for $f = f_1$ is achieved if the probability of a lower priority queue being filled with more than k_q cells is low, i.e. $q_{i,o,p}$ should not exceed $k_q \cdot q_{io(p-1)}$ at any time[1]. A complete separation can be forced if saturation is used for $f = f_2$ (eq. 4). If a wide dynamic range is necessary (e.g. for bursty traffic) but bits for the wordlength must be saved, a companding characteristic[2] (eq. 5) can additionally be used. Therefore the integer numbers $0..2^{b_v} - 1$ are mapped to $0..2^{b_p} - 1$ by a monotonic nonlinear function with a decreasing slope and saturation beyond the interval $[0..2^{b_v} - 1]$. The following companding function is considered most suitable:

$$f_{compand}(w_{iop}) = \left\{ \begin{array}{rl} 0 & : \; w_{iop} = 0 \\ \lfloor \log_2(w_{iop}) + 1 \rfloor & : \; 0 < w_{iop} < 2^{b_v} \\ 2^{b_p} - 1 & : \; w_{iop} \geq 2^{b_v} \end{array} \right. \tag{7}$$

which requires $2^{b_p} - 1 = \lfloor log_2(2^{b_v} - 1) + 1 \rfloor$ or $2^{b_p} - 1 = b_v$. The most effective value is $b_p = 3$ bit offering a dynamic range of $b_v = 7$ bit (fig. 6), which has shown to be sufficient for good results. The operation can very well be implemented in hardware, since only the position of the highest 1 bit must be determined for the log_2 operation.

In the port a scheduler (static priority, HOL) must react properly to the instructions of the arbiter, i.e. it must send a cell with the highest priority available in the case that cells of more than one priority are available for the specified output port, but only a globally prioritizing arbitration can achieve the correct separation.

4 An Approximation for Weighted Arbitration

As one low complexity method to achieve weighted matching by approximating MWM, the *SIMP* (successive incremental matching over multiple ports) algorithm is explained in this section.

We observe that SIMP approximates MWM by successively choosing the edge with the highest weight for each output node in order. This resolves the output contention problem for each port in a cyclic manner. It cannot generally find the global maximum MWM would find. Visually we can imagine that SIMP only considers one triangular half of the weight matrix, instead of the whole as with MWM. Due to the cyclic shift of this viewport the number of weights considered for a specific output port changes cyclically from 1 to M. As we see in section 5 the approximation yields reasonable results.

[1] Fig. 5 shows an example where $Pr(q_{iop} > 2^8) < 5 \cdot 10^{-5}$

[2] similar to the *µlaw* used for speech coding

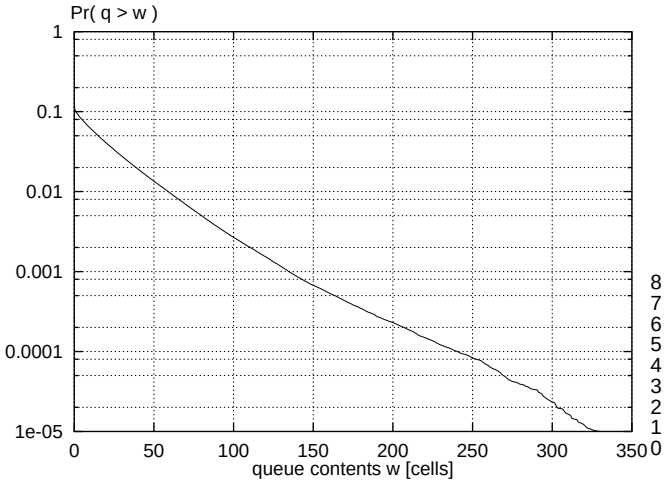

Figure 5: CDF for $w_{i,o,2}$, $b_1 = 8$, $\rho = 0.7, s_{p1} = 0.2$, bursty

Algorithm „SIMP"

1	let I be an ordered list of all input ports and O the list of all output ports
2	let $I' \leftarrow I$ and $O' \leftarrow O$
3	choose the first output port o_c out of the ordered list O'
4	choose the input port i_c to match as one with $w_{i_c} = \max_{i \in I'}(w'_i)$, resolve ambiguities (same weights) in round-robin fashion
5	if $w_{i_c} > 0$ match i_c with o_c and let $I' \leftarrow I' \setminus i_c$
6	reduce the match space by letting $O' \leftarrow O' \setminus o_c$
7	repeat steps 3-6 until $O = \emptyset$ (M repititions)
8	shift the list O cyclically before the next slot to achieve round-robin fairness between outputs
9	start the next time slot at step 2

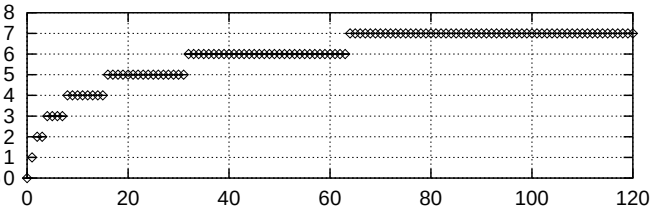

Figure 6: proposed companding function $f_{compand}$ ($b_v = 7$)

This algorithm loops over M output ports, needs $(M-1)..0$ comparisons per output port, so the overall complexity is $M \cdot (M-1)/2)$ sequential operations ($O(M^2)$). In hardware we need one M-port-comparator of complexity $\log_2 M$ for M iterations, i.e. $O(M \log_2 M)$ parallel operations. The hardware complexity is M^2 due to the memory needed. For comparison, iLQF [3] has $2M$ times more comparators in parallel, thus reducing the calculation time. But the decision dependencies are broken and a considerable number of iterations is needed to achieve 100% throughput.

Typically the situation looks like fig. 3 left in the ports. The numbers on the chessboard represent the weight[3] w_{io}. Using the algorithm above (starting with $o = 0$) we successively match the port combinations $1-0$, $3-1$, $0-2$ and $2-3$ as shown in fig. 3 right. The diagonal elements have content zero, because cells are not routed in and out through the same port. Consider fig. 4 where after $M-1$ correct matches a match in the last column would only be possible on an empty diagonal position. This can occur here, but this is also the match found by MWM!

5 Performance Results

In this section we compare the performance results for reference algorithms output queueing, MWM, iSLIP[4], PIM, iMCFF with SIMP using simulations for an 16x16 switch with OPNET [20].

With Bernoulli traffic and a symmetrically chosen destination[5], each input port carries a total offered load of ρ, i.e. $\lambda_{i,o} = \rho/(M-1)$.

Bursty traffic is important for many reasons. According to [21] we use a packet train model producing a burst of $B(t)$ cells at full rate followed by empty slots such that the mean rate $\lambda_{i,o} = \rho/(M-1)$ is the desired fraction of the input rate. B is exponentially distributed here; the *mean burst length* $\overline{B}$ is 32. This corresponds to the mean length of an Ethernet PDU segmented into ATM cells.

In fig. 7 the results for PIM and iSLIP are as in [21]. The best case is output queueing which we want to approximate with VOQ. The ideal arbitration for VOQ is MWM which most closely approximates output queueing. As we see, iMCFF performs worse than SIMP. The performance of SIMP is in between iMCFF and MWM. Simulations results for a *stable_marriage* algorithm come close to iMCFF and are omitted here.

In fig. 8 we see the delay performance for the bursty scenario. The most important characteristics are: (i) the absolute delay is two decades higher than for the previous scenario due to the burst scale queueing effect and the short-term asymmetries. (ii) PIM and iSLIP perform similar with more than 300 cell transmission times above $\rho = 0.6$, i.e. noticably worse than the other algorithms. (iii) As before, iMCFF, SIMP, MWM and output queueing are quite close to each other with improved performance (less delay) in ascending order.

We observe that the algorithms that decide based on weights (iMCFF, SIMP, MWM) perform very well for typical traffic.

When using the first priority mechanism f_1 the main parameter for achieving priority separation is the priority weight factor $k_q = 2^{b_q}$ (b_q shift bits). The other two degrees of freedom in the choice of traffic parameters manifest in $\rho_{total} =$

[3] typical heavy load situation with all queues filled significantly

[4] here only 1 iteration shown

[5] among the 15 ports with a different port number

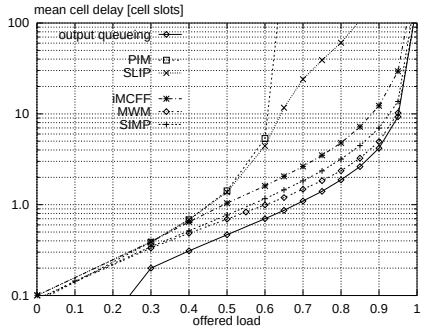

Figure 7: Std. scenario

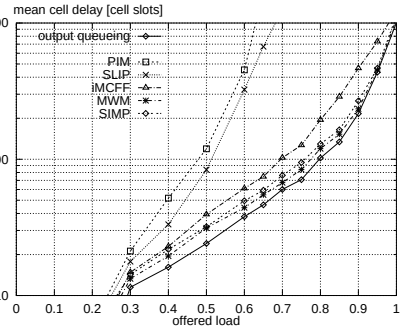

Figure 8: Burst scenario

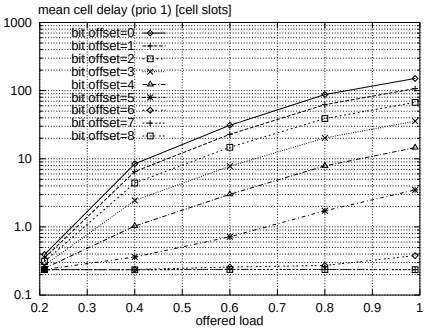

Figure 9: priority 1 (MWM, $\rho_1 = 0.2$)

$\rho_1 + \rho_2$ and s_{p1} (share of first priority cells) with $\rho_1 = s_{p1} \cdot \rho_{total}$ and $\rho_2 = (1 - s_{p1}) \cdot \rho_{total}$. The behavior for two priority classes[6] can be seen in fig. 9, where $\rho_1 = 0.2$ is held constant and ρ_2 is varied as bursty traffic. We see that there is almost no influence of the lower priority load on the higher priority performance when seven or more bits are used. As the CDF in fig. 5 shows, the probability of a weight overflow is low then.

To analyse quantitatively how many bits b_p are needed we study the mean delay for both priority groups at a fixed load $\rho = 0.7$ and a fixed ratio $s_{p1} = 0.2$ as a function of different b_p. The fixed values used have revealed to be the most expressive.

In fig. 10 and 11 we see that for Poisson traffic the result with about $b_p = 2$ bits is quite close to $b_p \to \infty$. For bursty traffic fig. 12 and 13 shows that about $b_p = 6$ bits are needed to fully separate the priorities. We can verify in fig. 10 and 12 and that with enough bits $\bar{d}_{prio1} = 0.2 slots$ at load $\rho_1 = 0.2$. This is exactly the delay for MWM with $\rho = 0.2$ in fig. 7. Thus we have an exact priority separation. The higher priority traffic performance does only depend on its total load. The values for SIMP are shown in fig. 13.

The rules for the necessary b_p depend on the traffic in priority p (fig. 5). This is not desirable, so we apply the deterministic separation by saturation (eq.4), where no overlapping is possible, i.e. $w_{io,p+1}$ cannot become higher than 2^{b_p}. We lose some of the weight dynamics in this priority because the values possible are in the interval $[0, 2^{b_p} - 1]$. Thus with b_p we can also adjust the performance within a priority between that of MWM ($b_p \to \infty$) and MSM ($b_p = 1$). Note that this has an influence also on the sensibility to asymmetry.

For the bursty scenario we observe in fig. 14 that now the separation is much improved for 1..5 bits. There is still some influence typical for MWM[7]. We can even choose one bit per priority and achieve a considerable separation. This, however, ignores all weights within a priority level and yields a bad performance under asymmetric workload as MSM does. In fig. 15 we finally observe the very best separation. Here three bits are sufficient for a complete separation, whereas the dynamics is as good as pure MWM.

6 Conclusion

In this paper we show how a weighted arbitration algorithm can be used to support priorities and offer performance results for two priority classes using MWM or our SIMP algorithm. The discussion of the number of bits needed to represent a weight is an important step towards implementation. As a result, using three bits per priority and the proposed companding function is sufficient for a complete priority separation.

References

[1] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications*, pp. 1347–1356, Dec. 1987.

[2] A. Mekkittikul and N. McKeown, "A Starvation-free Algorithm For Achieving 100% Throughput in an Input-Queued Switch," in *Proc. of the IEEE International Conference on Communication Networks*, 1996.

[3] N. McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, UC Berkeley, 1995.

[6] here the higher priority traffic is Bernoulli, the lower is bursty

[7] The maximum matching is the one that maximizes the sum of the matched weights. This sum might be still higher for some lower priority matches (e.g. $7+7$) than for the alternative match (e.g. $8+0$)

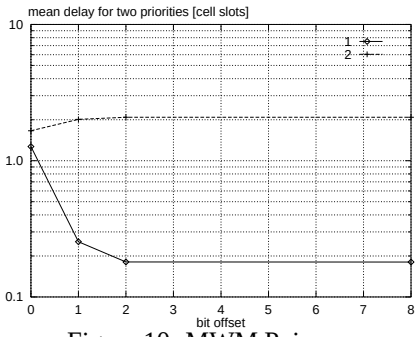

Figure 10: MWM Poisson

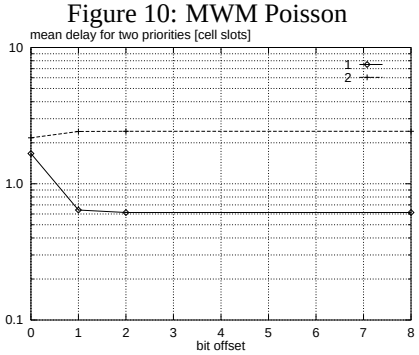

Figure 11: SIMP Poisson

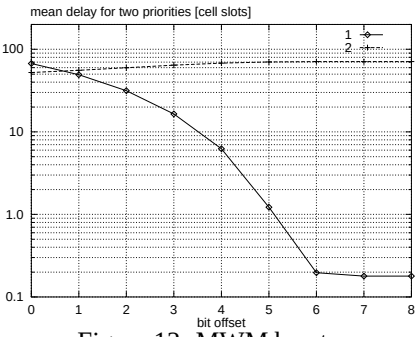

Figure 12: MWM bursts

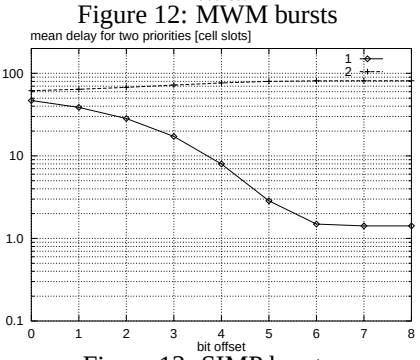

Figure 13: SIMP bursts

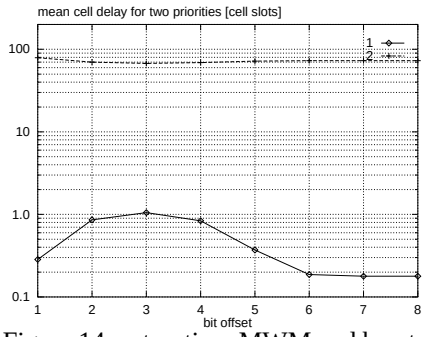

Figure 14: saturation, MWM and bursts

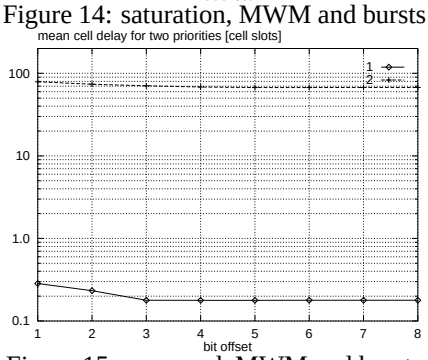

Figure 15: compand, MWM and bursts

[4] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Transactions on Communications,* 1998.

[5] A. Mekkittikul and N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," *Proceedings of the IEEE INFOCOM*, 1998.

[6] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High Speed Switch Scheduling for Local Area Networks," *ACM Transactions on Computer Systems*, vol. ?, Nov 1993.

[7] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, "The Tiny Tera: A Packet Switch Core," *IEEE Micro*, p. 26, Jan 1997.

[8] K. Rothermel, "Priority Mechanisms in ATM Networks," in *Proceedings of the IEEE GLOBECOM*, 1990.

[9] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization.* Prentice-Hall, Inc., 1982.

[10] H. Duan, J. Lockwood, S. Kang, and J. Will, "A High-Performance OC-12/OC-48 Queue Design Prototype for Input-buffered ATM Switches," *Proceedings of the IEEE INFOCOM*, 1997.

[11] Y. Tamir and H.-C. Chi, "Symmetric Cross Bar Arbiters for VLSI Communication Switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 13–27, 1993.

[12] P. Gupta and N. McKeown, "Design and Implementation of a Fast Crossbar Scheduler," in *Hot Interconnects,* 1998.

[13] M. Hluchyj and M. Karol, "Queueing in Space-Division Packet Switching," *Proceedings of the IEEE INFOCOM*, vol. ?, p. 334, Mar. 1988.

[14] C.-Y. Chang, A. Paulraj, and T. Kailath, "A Broadband Packet Switch Architecture with Input and Output Queueing," *IEEE*, pp. 448–452, 1994.

[15] M. Lee and S.-Q. Li, "Performance of a Nonblocking Space-Division Packet Switch in a Time Variant Nonuniform Traffic Environment," *IEEE Transactions on Communications,* vol. 39, p. 1515, Oct. 1991.

[16] N. McKeown, B. Prabhakar, and M. Zhu, "Matching Output Queueing with Combined Input and Output Queueing," in *35.th Annual Conf. on Communications, Control and Computing, Monticello, Illinois*, Oct 1997.

[17] H. Takagi, *Queueing Analysis - Vacation and Priority Systems*, vol. 1. ISBN 0-444-88910-8: North-Holland, 1991. RWTH: Bf9356-1.

[18] G. Nong, K. Muppala, and M. Hamdi, "Analysis of Non-blocking ATM Switches with Multiple Input Queues," in *Proceedings of the IEEE GLOBECOM*, 1997.

[19] H. Kim, C. Oh, and K. Kim, "A High-Speed ATM Switch Architecture Using Random Access Input Buffers and Multi-Cell-Time Arbitration," in *Proceedings of the IEEE GLOBECOM*, 1997.

[20] MIL 3, Inc., 3400 International Drive, NW Washington, DC 20008, USA, *OPNET Release 4.0*, 1998. http://www.mil3.com.

[21] N. McKeown and T. Anderson, "A Quantitative Comparison of Scheduling Algorithms for Input-Queued Switches," *published?*, 1997. http://http.cs.berkeley.edu/%7Etea/atm.html.