Weighted Arbitration Algorithms with Priorities for Input-Queued Switches with 100% Throughput

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

Abstract

Input buffered switches have the strong advantage of efficient crossbar usage. Virtual Output Queueing (VOQ) has to be established to circumvent the head-of-line (HOL) blocking which limits the throughput to 58.6%. Arbitration algorithms control the access to the switch fabric in each time slot. Weighted algorithms achieve 100% throughput with lowest delays under all admissible traffic even under highly asymmetric load. In this paper we compare existing algorithms and propose a new algorithm for weighted matching with a delay performance close to the ideal maximum weight matching. As an important step towards implementation we emphasize the finite wordlength required for weights. This can be exploited to support priorities which is required for ATM or IPv6 switches.

1 Introduction

Very high speed switches are needed for future ATM and IPv6 networks. Among the switching architectures input queued switches belong to the fastest 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]. Virtual Output Queueing (VOQ) [2] can totally eliminate this problem by each input having a separate queue for each output (fig. 1). This potentially allows a throughput of 100% [3, 4, 5].

Arbitration algorithms resolve the contention for the same output ports in each time slot. The arbitration algorithm mainly determines the achievable throughput and delay performance. Weighted algorithms [4] not only offer a much better delay performance than unweighted or single-iteration distributed algorithms [6], they are required to operate well in other than symmetric load configurations. There are very few algorithms for efficient weighted arbitration. Priorities are rarely treated so far, although priorities are most important to support more than best-effort traffic only.

In this paper we offer a new efficient weighted arbitration algorithm that achieves 100% throughput in load conditions where most other algorithms fail. 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. With very limited weight wordlengths it still operates well. Priority support is a straightforward extension of this concept. The complexity of the arbitration algorithm is only determined by the total number of bits. The performance impact of having a nonzero latency (to allow pipelining) is also shown.

The paper is organized as follows. Section 2 provides the necessary background. The new arbitration algorithm is presented in section 3. Finite wordlength and priorities are introduced in section 4. Section 5 shows performance results.

2 Background and Related Work

The VOQ configuration [2] shown in fig. 1 consists of M ports for input and output, a nonblocking switch fabric and an arbitration unit. Arriving cells on input port i are placed into the corresponding queue for their destination port o. Its current queue content 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 is equivalent to 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 iMCFF [5] and iLQF [3]. Even the unweighted MSM problem is rated $O(M^{2.5})$. MSM approximations exist with PIM [6], iSLIP [3, 10, 7] or WFA [8]. Priorities have been used with iSLIP and ESLIP [10, 7], but as with every unweighted algorithm the delay becomes large for bursty or asymmetric load.

In table 1 the most important algorithms are listed. The best iterative unweighted algorithm takes $O(M \log M)$ serial or $O(\log M)$ parallel time steps, while the hardware area takes $O(M^4)$. The weighted algorithm iLQF [3] has the lowest complexity in time, but it needs $2M$ times more hardware than our

[1] not for all admissible traffic loads (e.g. symmetric only)
[2] priorities possible with our method without complexity increase
[3] area complexity of the arbiter(s)
[4] area complexity of the state memory

Algorithm	ρ_{max} [%]	stable	fair	delay	weights	prio	serial	parallel	gate count	ref.
IQ/FCFS	59	-	-	–	-	-	$O(1)$	$O(1)$	$O(M)$	[1]
IQ/random	63	-	-	–	-	-	$O(1)$	$O(1)$	$O(M)$	[1]
PIM	63	-	-	–	-	-	$O(M \log M)$	$O(M \log M)$	$O(M^2)$	[6]
RRM	63	-	+	–	-	-	$O(M \log M)$	$O(M \log M)$	$O(M^2)$	[3]
SLIP	$(100)^1$	-	+	-	-	-	$O(M \log M)$	$O(M \log M)$	$O(M^4)$	[3]
iSLIP	$(100)^1$	-	+	-	-	-	$O(i \cdot M \log M)$	$O(i \cdot M \log M)$	$O(M^4)$	[3]
ESLIP	$(100)^1$	-	+	-	-	+	$O(i \cdot M \log M)$	$O(i \cdot M \log M)$	$O(M^4)$	[7]
PSLIP	$(100)^1$	-	+	-	-	+	$O(i \cdot M \log M)$	$O(i \cdot M \log M)$	$O(M^4)$	[3]
WFA	$(100)^1$	-	+	-	-	-	$O(M^2)$	$O(M)$	$O(M^2)$	[8]
MSM	$(100)^1$	-	-	-	-	-	$O(M^{2.5})$	$O(M^{2.5})$	-	[4]
MWM	100	+	-	+	+	$(*)^2$	$O(M^3 \log M)$	$O(M^3 \log M)$	-	[4]
SM-GSA	100	+	-	+	+	-	$O(M^2)$	$O(M^2)$	-	[3]
MCFF	100	+	-	+	+	-	$O(b \cdot M^3)$	$O(b \cdot M^3)$	-	[5]
iMCFF	100	+	-	+	+	-	$O(b \cdot M^2)$	$O(b \cdot M^2)$	N/A	[5]
iLQF(i=1)	63	-	-	–	+	$(*)^2$	$O(i \cdot b \cdot M \log M)$	$O(i \cdot b \cdot \log M)$	$O(b \cdot M \log M)^3 + O(b \cdot M^2)^4$	[3]
SIMP	100	+	+	+	+	$(*)^2$	$O(b \cdot M^2)$	$O(b \cdot M \log M)$	$O(b \cdot M)^3 + O(b \cdot M^2)^4$	here

Table 1: Arbitration algorithms (+ means good performance / feature available)

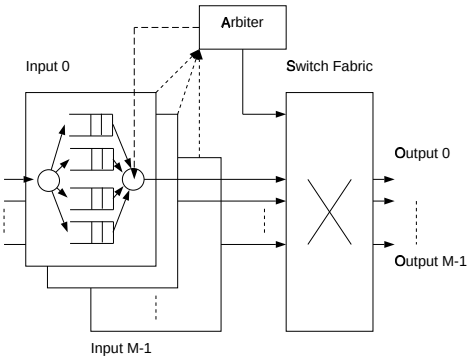

Figure 1: virtual output queueing

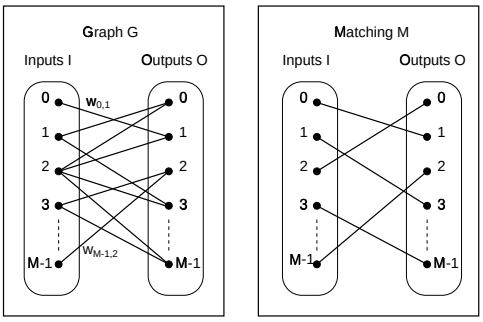

Figure 2: Bipartite graph matching

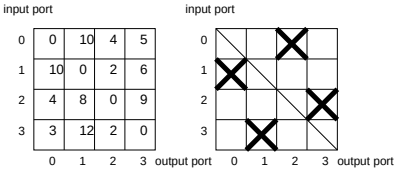

Figure 3: typical match

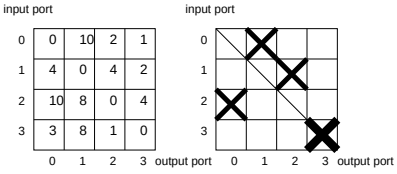

Figure 4: bad match

proposed algorithm and needs at least four iterations to achieve 100% throughput with comparable performance.

An internal speedup can simplify the algorithms [11, 12, 13, 14], 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 [10].

Our recent work [15] provides a concept to include priorities with input-queued switches and one-step weighted matching, where both independence of better traffic classes [16] and a small cell delay can be achieved. The limited wordlength effects are discussed here.

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, 17], or the assumptions simplify it to an OQ model [18].

3 A New Weighted Arbitration

For weighted matching MWM offers the best performance at the highest complexity. A practical algorithm to approximate MWM is *SIMP*[5] which simplifies the maximization of the total weight sum to the maximization of partial weights. This algorithm approximates MWM by sequentializing the search for maximum arc weights. Starting with any output port (which is rotated cyclically for the next time slot) the maximum of one weight column is taken (in hardware this requires $log_2 M$ comparator stages). In each following step the maximum weight of the remaining rows ($(M-1)..0$ comparisons) is chosen. After M steps $M \log_2 M$ parallel ($M \cdot (M-1)/2$ sequential) operations have been performed.

By this method only one triangular half of the weight matrix is considered in this time slot ($(M^2 \pm M)/2$ entries), not all M^2 entries as with MWM. As we see in section 5 the approximation yields reasonable results.

The hardware iteration period can be significantly improved by pipelining and parallelizing some comparator steps. Then the increased relative latency (in cell slot times) makes the algorithm work on slightly outdated weight information. In section 5 the performance impact is shown. The hardware complexity is $O(M^2)$ due to the memory needed.

Another MWM approximation, iLQF [3] reduces the calculation time with $2M$ times more comparators in parallel operating independently for every output. But the decision dependencies are broken, so that HOL blocking leads to 65% max. throughput. Many sequential iterations are needed to achieve 100% throughput.

In fig. 3 two state matrix situations are shown[6]. Using the algorithm above (starting with $o = 0$) we successively match the port combinations $1-0, 3-1, 0-2$ and $2-3$. 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 may occur, but is also the match found by MWM!

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'
4a	choose the input port i_c to match as one with $w_{i_c} = \max_{i \in I'}(w'_i)$,
4b	resolve ambiguities (same weights) in round-robin manner
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	rotate the list O cyclically before the next slot to achieve round-robin fairness between outputs
9	start the next time slot at step 2

[5] successive incremental matching over multiple ports

[6] The numbers on the chessboard represent the weights w_{io}; typical heavy load situation with all queues filled significantly

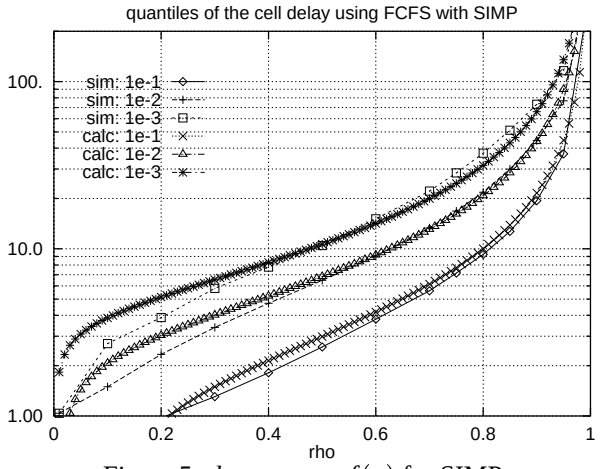

Figure 5: $d_{quantile(x)} = f(\rho)$ for SIMP

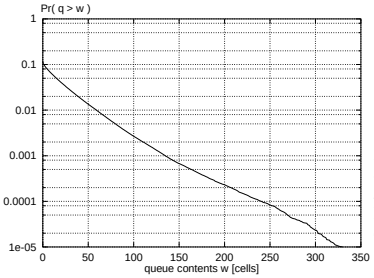

Figure 6: CDF for $w_{i,o,2}$, $b_1 = 8$, $\rho = 0.7, s_{p1} = 0.2$, bursty

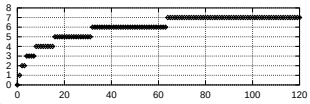

Figure 7: proposed companding function $f_{compand}$ with $b_v = 7$

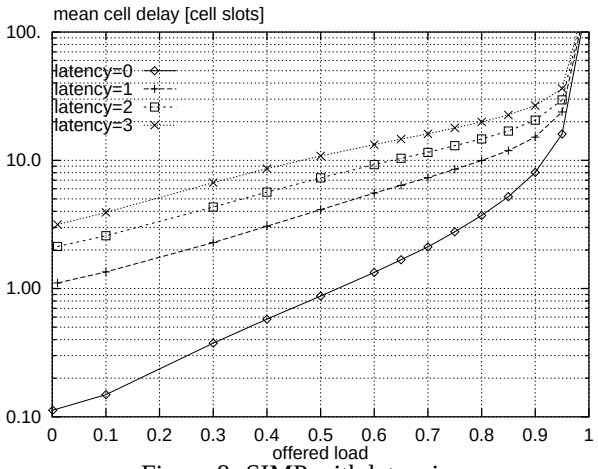

Figure 8: SIMP with latencies

4 Finite Weight Wordlengths

Hardware for SIMP or any other weighted arbitration algorithm cannot be implemented with an infinite range for weights. Any practical realization will limit the number of bits used for the calculations. As we show now, it is not necessary to support a huge range of weights for each possible queue occupancy (which can be several MB for best-effort traffic).

We assume that the queue state on input i, output o in fig. 1 is also divided into P priorities (1=highest), i.e. $q_{i,o,p}$ describes this state. The mean rate $\lambda_{i,o,p}$ characterizes the arrivals into these queues.

Within one priority class p we define the number of bits b_p to use for the weight representation $f(q_{i,o,p})$, which corresponds to the *allowed* numeric range $0..(k_p - 1)$ with $k_p = 2^{b_p}$. This main parameters, *the number of bits per priority* represents the dynamic range and allows scaling from MSM ($b_p = 1$) to MWM ($b_p \to \infty$). The wider the dynamic range is, the better the delay performance will be. When dealing with finite wordlengths each partial weight $f(q_{i,o,p})$ must not exceed its bit width b_p. Thus these alternatives can be identified:

$$f_{eq}(q_{iop}) = q_{iop} \quad (1)$$

$$f_{sat}(q_{iop}, k_p) = \{ \begin{array}{cc} q_{iop} & if \quad q_{iop} < k_p \\ k_p - 1 & if \quad q_{iop} \geq k_p \end{array} \} \quad (2)$$

$$f_{compand}(q_{iop}) = \left\{ \begin{array}{r} 0 \; : \; q_{iop} = 0 \\ \lfloor \log_2(q_{iop}) + 1 \rfloor \; : \; 0 < q_{iop} < 2^{b_v} \\ 2^{b_p} - 1 \; : \; q_{iop} \geq 2^{b_v} \end{array} \right. \quad (3)$$

For a wider dynamic range (e.g. for bursty traffic) within the same limited wordlength a companding[7] function can be used. This maps integer numbers $0..2^{b_v} - 1$ to $0..2^{b_p} - 1$ by a monotonic nonlinear function with a decreasing slope and saturation beyond the interval $[0..2^{b_v} - 1]$. Eq. 3 has revealed[8] to be most suitable, because it can very well be implemented in hardware, since only the position of the highest 1 bit must be determined for the log_2 operation. It requires $2^{b_p} - 1 = \lfloor log_2(2^{b_v} - 1) + 1 \rfloor$ or $2^{b_p} - 1 = b_v$. As shown in section 5.3 the most effective value is $b_p = 3$ bit offering a dynamic range of $b_v = 7$ bit (fig. 7). When the numeric range exceeds a certain threshold, i.e. $q_{i,o,p} \geq k_p - 1$ with saturation or $q_{i,o,p} \geq 2^{b_v - 1} = 2^{2^{b_p} - 2}$ with companding, the weight doesn't increase further. That also means that congested port directions are treated fairly, because SIMP arbitrates equal-weighted ports in round-robin manner.

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

[8] minimizes the mean square error for exponential distributions

With weighted matching algorithms full throughput under admissible load can be achieved [4]. When there is an admissible load only of higher priority cell streams, an unbounded delay for all cells on one port can occur when there is no priority support in the arbiter, and lower priority (best effort) connections experience congestion (high queue weight $w_{i,o}$) on another port. There must be no influence of lower priority queues on higher priority performance, as known for a discrete priority system [19].

To be compatible with existing weighted matching algorithms $[q_{i,o,p}]$ is mapped to $[w_{i,o}]$ in a special way. With $w_{i,o} = \sum_{p=1}^{P} q_{i,o,p}$ there is no priority support. Priority support is achieved by applying this weighted summation:

$$w_{io} = \sum_{p=1}^{P} c_p \cdot f(q_{iop}), \qquad c_p = \prod_{q=p+1}^{P} k_q = \prod_{q=p+1}^{P} 2^{b_q} \quad (4)$$

The coefficients c_p are computed 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. For example,

$$P = 3; \forall p : b_p = 3 \Rightarrow k_p = 8$$
$$q_1 = 2 = 010_b, q_2 = 5 = 101_b, q_3 = 15$$
$$\Rightarrow f_{sat}(q_3, 8) = 111_b \Rightarrow w = 010101111_b$$

For a sufficient priority separation with f_0 the probability of a lower priority queue being filled with more than k_q cells must be low. For example, $q_{i,o,p}$ should not exceed $k_q \cdot q_{io(p-1)}$ at any time[9]. With saturation or companding separation is forced by construction.

[9] Fig. 6 shows an example where $Pr(q_{iop} > 2^8) < 5 \cdot 10^{-5}$

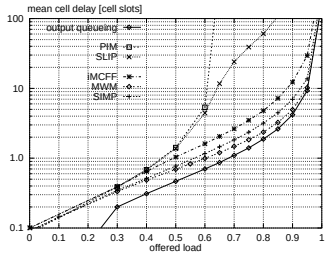

Figure 9: Std. scenario

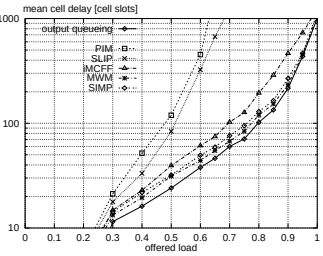

Figure 10: Burst scenario

strategy:	OQ	MWM	SIMP	iMCFF	MSM	RANDOM
w_{PCR+}/w_{PCR-}	100	12.68	18.73	137.2	3314	8624
d_{PCR+}/d_{PCR-}	1	0.128	0.191	1.367	32.5	86.68

Table 2: spread at $\rho = 0.9$ for asymmetric scenario

5 Arbitration Performance

Simulation results for the performance of SIMP and the reference algorithms output queueing, MWM, iSLIP[10], PIM and iMCFF have been obtained for an 16x16 switch with OPNET [20].

For the symmetric scenarios[11], each input port carries a total offered load of ρ, i.e. $\lambda_{i,o} = \rho/(M-1)$ for Bernoulli traffic.

Bursty traffic is used to represent VBR or ABR traffic. According to [21] we use a packet train model producing a burst of B cells (exponentially distributed) at full rate followed by empty cells such that the mean rate $\lambda_{i,o} = \rho/(M-1)$ is the desired fraction of the input rate. The *mean burst length* $\bar{B}$ is 32. This corresponds to the mean length of an ethernet PDU segmented into ATM cells.

5.1 Pure Arbitration Results

For one priority class the results in fig. 9 for PIM and iSLIP are as in [21]. Output queueing is the best case which we want to approximate with VOQ. The ideal arbitration for VOQ is MWM which most closely approximates output queueing. The performance of SIMP is in between iMCFF and MWM. For better visibility of the graphs, results for the *stable_marriage* and *WWFA* algorithms are omitted, because they are very close to iMCFF. For SIMP we can also provide delay quantiles (fig. 5) and the performance impact when latencies are present (fig. 8).

The burst scenario in fig. 10 shows the following characteristics: (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 ($i = 1$) 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. This is a first indication that weighted algorithms (iMCFF, SIMP, MWM) perform very well for typical traffic. Our results for SIMP showed no visible difference between using $b_p = 3bit$ with companding and infinite wordlength.

5.2 Asymmetric Input Traffic

Symmetries in the traffic matrix $[\lambda_{i,o}]$ exert much influence on the delay of the different streams. We study this with a scenario where rates of Bernoulli[12] traffic differ by a range $r = 100:1$ between the ports, but where the total load on each link is nevertheless equal to ρ. Let the normalized arrival rate matrix $[\lambda_{i,o}]$ have $\lambda_{i,[(i+j)modM]} = \rho \cdot a_j$ using asymmetry coefficients a_j. By establishing these traffic intensities we achieve that all neighbor ports are different by a factor f.

$$\begin{aligned} &a_0 = 0, a_1 = (f-1)/(f^{M-1}-1), a_j = a_1 \cdot f^{j-1} \ \ \forall \ \ j \neq 0 \\ &\lambda_{i,o}/\lambda_{[(i+1)modM],o} = f \ \ \forall \ \ i \neq o, [(i+1)modM] \neq o \\ &f = r^{-1/(M-2)} = (100:1)^{-1/(M-2)} \end{aligned} \tag{5}$$

The delay performance has been studied for a number of algorithms and weight bits. Here only SIMP is shown in fig. 11, with the rate group[13] as a parameter, because that is the only differentiator next to ρ.

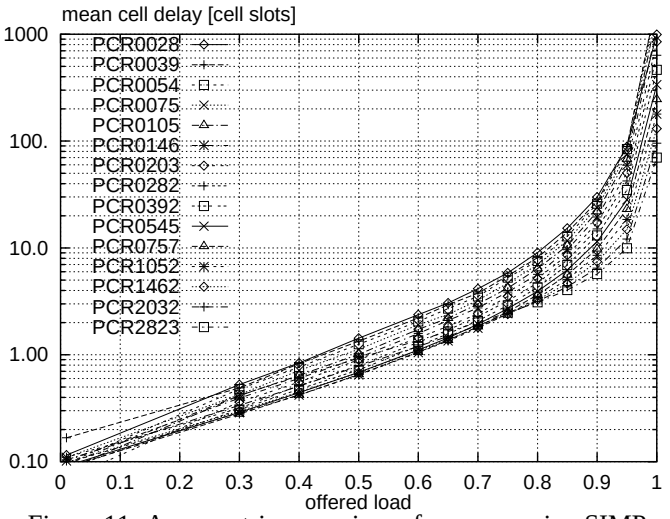

Figure 11: Asymmetric scenario performance using SIMP

b_p bits:	1	2	3	4-8	3+compand
w_{PCR+}/w_{PCR-}	1080	33.34	31.52	31.66	34.2
d_{PCR+}/d_{PCR-}	10.81	0.336	0.316	0.318	0.343

Table 3: spread at $\rho = 0.9$ for different wordlengths

For the comparison shown in table 2, $PCR+$ means the highest rate group (PCR2823), $PCR-$ the lowest (PCR0028). At $\rho = 0.9$ we see a delay spread (d_{PCR+}/d_{PCR-}) of around 0.2, with the highest rate group (PCR+) getting the lowest delay. We observe that unweighted algorithms have a much higher deviation in the delay and queue contents. Algorithms based on weighted matching tend to keep the queue length in each virtual queue for any output o on the same level because they prefer those with higher weight w_{io}. And in fact, the mean queue lengths are closer together for MWM, SIMP and iMCFF than for MSM and RANDOM. The big numbers for w_{PCR+}/w_{PCR-} with unweighted matching show that this is not a very good choice. This emphasises the necessity to use weighted matching in a switch. Using the knowledge $\lambda_{PCR+}/\lambda_{PCR-} = r = 100$ the results in table 2 are consistent with Little's Law

$$d_x \cdot \lambda_x = w_x \Rightarrow \frac{d_{PCR+}}{d_{PCR-}} \cdot \frac{\lambda_{PCR+}}{\lambda_{PCR-}} = \frac{w_{PCR+}}{w_{PCR-}} \tag{6}$$

The ratio d_{PCR+}/d_{PCR-} is equal one for pure output queueing (OQ). With SIMP or any other weighted algorithm the results are very close to that. In table 3 the results for restricted wordlengths for SIMP are shown, using saturation or companding. The essence is that with a few bits a good performance even in bursty and asymmetric traffic conditions can be achieved.

[10] here only 1 iteration is shown

[11] destination chosen symmetrically among the 15 remaining ports

[12] for bursty traffic the relative performance is very similar

[13] rate group is denoted PCRxxxx, corresponds to same a_j

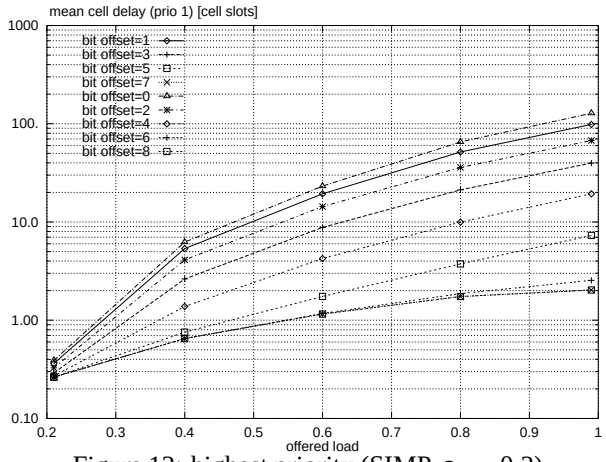

Figure 12: highest priority (SIMP, $\rho_1 = 0.2$)

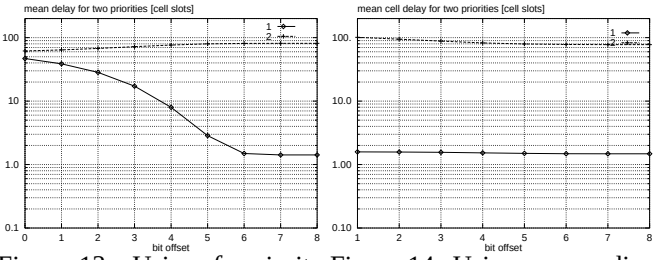

Figure 13: Using f_0 priority (SIMP and bursty traffic)

Figure 14: Using companding (SIMP and bursty traffic)

5.3 Priority Arbitration Results

Priority separation is obtained by the main parameter *priority weight factor* $k_q = 2^{b_q}$ (b_q shift bits). Two degrees of freedom in the choice of traffic parameters manifest in $\rho_{total} = \rho_1 + \rho_2$ 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[14] can be seen in fig. 12, where $\rho_1 = 0.2$ is held constant and ρ_2 is varied as bursty traffic. We see that the influence of the lower priority load on the higher priority performance is only two cell slots when seven or more bits are used with SIMP. As the CDF in fig. 6 shows, the probability of a weight overflow is low then. For MWM some more results can be found in [15].

The number of bits needed for a good priority separation can be obtained from fig. 13 and 14. For these results we studied 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. These values have revealed to be most expressive.

For bursty traffic and without saturation or companding about $b_p = 6$ bits are needed to separate the priorities in the best way. The higher priority traffic performance does almost only depend on its total load. The performance becomes totally independent when MWM is used [15], but with the SIMP approximation we still have some small coupling.

The deterministic separation by saturation avoids weight overlapping ($w_{io,p+1}$ cannot become higher than 2^{b_p}) but the weight dynamics is smaller because all possible values are in the interval $[0, 2^{b_p} - 1]$. With the same b_p companding is more effective. In fig. 14 we therefore observe the best separation. A three bit companded weight offers the best tradeoff between dynamics and priority separation.

6 Conclusion

Weighted arbitration algorithms are recommended for offering low delays under all workload conditions. Especially under other than symmetric loads unweighted algorithms do not offer full throughput and yield very high delays. We have proposed the SIMP algorithm as a useful approximation of maximum matching. Finite weight wordlengths are required for an implementation. This can be used seamlessly to support priorities. Performance results in various load conditions confirm the efficiency of the given algorithms.

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

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.

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

[8] 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.

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

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

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

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

[13] 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.

[14] 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.

[15] R. Schoenen, G. Post, and G. Sander, "Prioritized Arbitration for Input-Queued Switches with 100% Throughput," in *ATM Workshop'99*, 1999.

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

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

[18] 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.

[19] H. Takagi, *Queueing Analysis - Discrete-Time Systems*, vol. 3. ISBN 0-444-81611-9: North-Holland, 1991. RWTH: Bf9356-3.

[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.