Proceedings of the IASTED International Conference
WIRELESS AND OPTICAL COMMUNICATIONS
July 17-19, 2002, Banff, Canada

A MEMS based parallel packet switch

Claus Bauer
clausbauer@yahoo.com
Kofi Odame, Department of Electrical and Computer Engineering, Cornell University
kmo15@cornell.edu

Abstract— **This paper presents a parallel switch architecture based on MEMS technology. The architecture is explained and evaluated via analysis and simulation.**

Keywords— **Optical Packet Switching, Parallel Switch architecture, MEMS**

I. Introduction

The exponential growth of the Internet is driving the demand for higher transmission capacity and faster router architectures. While WDM technology is expected to meet those capacity requirements, the routers are likely to become a bottleneck. As electronic switching fabrics are not expected to match the capacities of future WDM based transport networks, optical switching cores are developed as an alternative ([9], [10]). With regard to their application to packet switches, all proposals face some common technological challenges.

1. No equivalent to electronic RAM is available in the optical domain ([4]).
2. Packet arrivals at different inputs are not synchronized ([12]).
3. Header evaluation is not practical in the optical domain ([12]).

Among the competing technologies, MEMS is considered as a promising candidate for commercial optical switches. However, MEMS based switches have not been considered for optical packet switches due to their long switching times of at least 0,5 ms ([10]) which are expected to cause large packet delays. In this paper we challenge the assumption that large switching times necessarily lead to unacceptable delays. We propose a parallel switch architecture which compensates for the long switching times by using parallel crossbar switching planes.

In [11] a parallel switch architecture is proposed to eliminate the need for speedup in the switching core. In [5] and [6] a parallel switch architecture is discussed where the switching planes work at a rate lower than the incoming and outgoing line rates. In contrast to our work, both approaches assume the switching time to be equal to zero. Whereas in [5] and [6] the forwarding speed is lower than the line speed, here the forwarding rate is equal to the line rate .

In MEMS based switches it is possible to reconfigure only a subset of the existing connections, while leaving other connections intact. We investigate if this property of MEMS based switches can be taken into advantage for the design of a scheduling algorithm. Because our investigation aims at evaluating the performance of the parallel switch in terms of delay, we do not deal with the problems 1 - 3 listed above, but assume packets to arrive in a synchronized way and header evaluation to be feasible. We assume packets to be buffered in the electronic domain.

In the next section we explain the switch architecture and its mode of operation. In section III we present the scheduling algorithms and derive upper delay bounds . The results of our simulations are presented in section IV and we give our conclusions in the final section.

II. Switch architecture and mode of operation

A. Architectural principles

Throughout this paper time is described via a discrete, slotted time model. We only consider packets of fixed length. A timeslot is defined as the time it takes to transport a packet over an incoming or outgoing link. In this paper $P_{i,j}$ denotes a packet that arrives at input i and is destined to output j. The long switching times decrease the throughput of the switch. In order to compensate for this, the architectures proposed here hold established connections open either for a fixed or dynamically determined large number of timeslots.

Switch architectures can be classified in those that use the capability of MEMS based switches to reconfigure only a subset of the existing connections and those that only allow to reconfigure the whole switch. In the latter case the operation of each parallel switching core can be divided in two phases:

$$T = \text{length of switching period in timeslots.}$$
$$M = \text{length of forwarding period in timeslots.}$$

We propose two architectures that differ in the degree of synchronization between the parallel planes: A simultaneous architecture where the forwarding and switching periods start simultaneously and the same configuration is always applied to all switches and a distributed architecture where the forwarding and switching periods of the switching cores are uniformly distributed in time and each switch decides its configuration individually. Finally we consider a switch architecture where subsets of connections can be reconfigured and the switching planes make individual forwarding decisions.

With a parallel architecture packets belonging to the same flow might be switched by different switching planes and not leave the switch in the order they have entered it. We describe how to maintain the flow order without having to

This paper was written when both authors were with the Tellabs Research Center

reorder packets at the outputs.

B. Simultaneous architecture

We use the parallel architecture proposed in [11] as depicted in figure 1. The input stage has N - inputs, each having an o-e converter and a de-multiplexer D_i. The output stage has N ports, each having a multiplexer M_i and an e-o converter. The switching stage is built of k-parallel NxN switching planes $S_1, .., S_k$ each of which consists of N virtual output queues (VOQ) at each input, an e-o converter at each input, the switch fabric, an o-e converter at each output and N output queues (OQ). Figure 2 shows the detailed buffer implementation. An optical packet arriving at an input is converted into electronics and sent to any of the k- input queues by the de-multiplexer. A packet scheduled to leave a virtual output queue is reconverted into an optical signal and sent through a switch. At the output of the switch it is reconverted to electronics and put into the output buffer of the switching plane. Once the multiplexer schedules the packet to leave the output buffer, it is reconverted to an optical signal and sent out. The parallel switches always apply the same configuration and start to reconfigure simultaneously. The scheduling algorithm takes into account all packets into all input buffers.

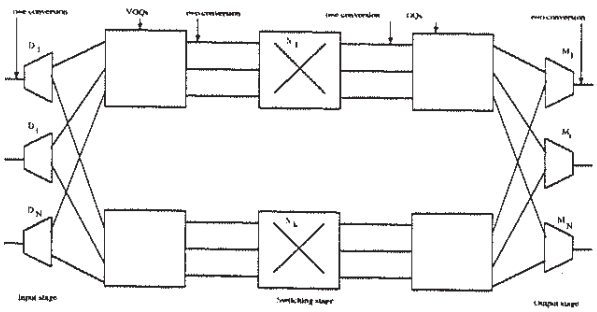

Fig. 1. Simultaneous switch architecture

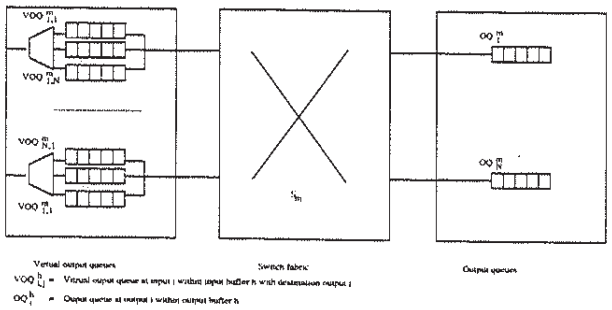

Fig. 2. Buffer design in the simultaneous switch architecture

The de-multiplexers distribute the arriving packets in a round robin fashion to the parallel switches. Sending the appropriate information from the de-multiplexer to the multiplexers ensures that the flow order is maintained at the outputs (see [11] for details).

C. Distributed architecture

The architecture consists of N inputs and N outputs identical to those in the simultaneous architecture. The switching stage consists of k input buffers B_m, a second layer of de-multiplexers $E_{i,j,m}$ at the head of each virtual output queue, e-o converters, the k- switching cores, o-e converters and k output buffers (see fig. 3 and 4). The second layer of de-multiplexers is introduced to take advantage of the individual switching decisions taking at each switching plane via a dynamic assignment of virtual output queues to the switching cores.

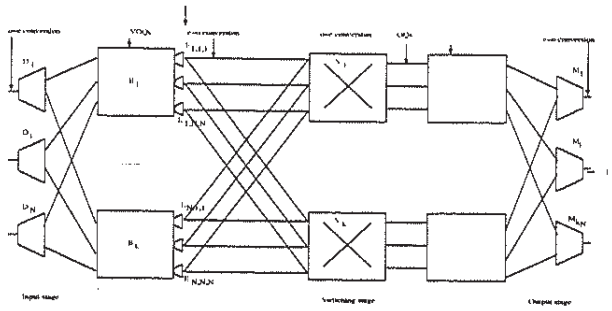

Fig. 3. Distributed switch architecture

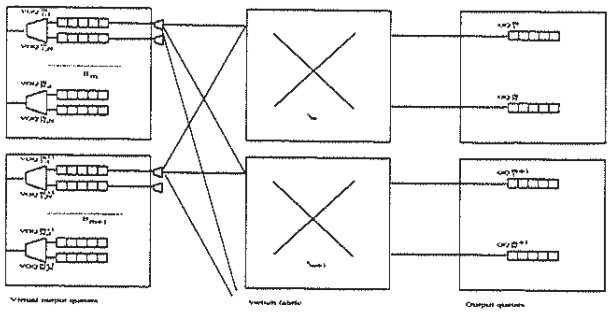

Fig. 4. Buffer design in the distributed switch architecture

In comparison, the simultaneous architecture assigns each input buffer statically to a switching plane. We motivate this design as follows. Assume that two packets $P_{m,j}$ and $P_{n,j}$ that compete for the same output j are buffered within the same input buffer B_h. In case of a static assignment, if the switch S_h configures the connection from input m to output j, the packet $P_{n,j}$ would remain in the buffer B_h. With a dynamic assignment, it could eventually be switched through another switch S_g that has established the connection from n to j. Now we consider the case of two packets $P_{i,j}$ and $P_{i,k}$ that compete for the same input i and are stored in the buffer B_g in the virtual output queues $VOQ^g_{i,j}$ and $VOQ^g_{i,k}$ respectively. The switch S_g decides to switch packet $P_{i,k}$. In case of a static buffer assignment $P_{i,j}$ remains in B_g. With a dynamic assignment $P_{i,j}$ could be switched through another switch S_h that has configured the connection from i to j. Note that in the first case of two packets that compete for the same output, a dynamic assignment of the group of all VOQs at one input i_0 in one buffer B_h, i.e. $VOQ^h_{i_0,j}$, $1 \leq j \leq N$ to a switching plane

would be sufficient. In contrast, the second case of two packets that compete for the same input requires the dynamic assignment of individual VOQs to a switching plane. A packet travels through the switch as with the simultaneous architecture the only change being that it can be forwarded by a de-multiplexer $E_{i,j,m}$ through any of the k switches.

Each switching plane decides its configuration individually. We distribute the starting times of the switching and forwarding phases of the switching planes uniformly over the time axis. We formalize this idea via a parameter R as follows:

R = time difference between the beginning of the switching periods of two switches $S_{i-1}(S_k)$and $S_i(S_0)$, where $S_i(S_0)$ is the first switch to start its switching phase after $S_{i-1}(S_k)$ has started its switching phase , $1 < i \leq k$.

From the definition of R follows that a switch starts a switching phase after every kR timeslots. As two consecutive switching periods are separated by a forwarding period of length M we obtain (see fig. 5)

$$M + T = kR. \tag{1}$$

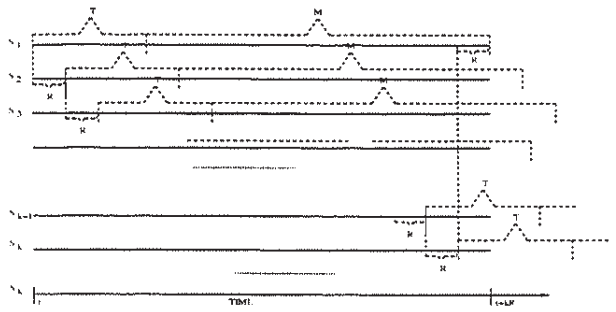

Fig. 5. Distribution of switching and forwarding phases in the distributed architecture

The de-multiplexers in the input stage distribute arriving packets in a round-robin fashion among the virtual output queues in the buffers B_m. To maintain the flow order we introduce a central control unit (CCU). The CCU also provides each individual switch with a complete picture of the packets buffered in all the VOQs which the switch uses as an input for its scheduling decision, as well as information about all existing and actually being configured input - output connections. The CCU knows the parameters M, T and R. The following information is sent to the CCU:

I Each multiplexer D_i in the input stage informs the CCU when it receives a packet.

II Each de-multiplexer $E_{i,j,m}$ informs the CCU when it sends a packet to a switch.

III Each switch informs the CCU about its chosen configuration at the beginning of its switching period.

The CCU uses this information to calculate the following parameters.

1. $K_{i,j}$ denotes the number of packets stored in all input buffers that will be switched from i to j. It is initially set to zero, increased by one when a packet $P_{i,j}$ enters the parallel switch and decreased by one when a packet $P_{i,j}$ is switched. $K_{i,j}$ is calculated by the CCU using the information I and II.

2. $T_{i,j}$ takes values between 1 and k and denotes the next input buffer from which a packet $P_{i,j}$ will be sent to one of the parallel switches. It always points to the input buffer containing the oldest packet $P_{i,j}$. It is increased by 1 modulo k each time a packet is switched from any of the k input buffers. $T_{i,j}$ is calculated by the CCU using the information II.

3. $M_{i,j}$ denotes a subset of the packets $K_{i,j}$. At each timeslot it denotes the number of packets stored in all input buffers that will be switched by a connection from i to j that either is already configured or is actually being configured. $M_{i,j}$ is initially set to zero and decreased by one when a packet $P_{i,j}$ is switched. When a packet $P_{i,j}$ enters the switch and $M_{i,j}$ is smaller than the number of the remaining switching opportunities of all either already configured or actually being configured connections from input i to output j, $M_{i,j}$ is increased by one. When a switch at the beginning of its switching phase decides that it will configure the connection from i to j, $M_{i,j}$ is increased by $\min(M, K_{i,j} - M_{i,j})$. $M_{i,j}$ is calculated by the CCU using the information I, II and III and the $K_{i,j}$.

The scheduling algorithm only takes into account the $K_{i,j} - M_{i,j}$, $1 \leq i,j \leq N$ packets in all VOQs for which a switching opportunity is not provided by connections that are already configured or are actually being configured. Each switch retrieves the required information from the CCU.

At the beginning of each timeslot the CCU knows from information III the $m_{i,j}$ switches $S_{a_1},..,S_{a_{m_{i,j}}}, 1\ m_{i,j} \leq k$ that have configured a connection between the input i and the output j. For each t satisfying $1 \leq t \leq u_{i,j} = min(m_{i,j}, K_{i,j})$ the CCU triggers the de-multiplexer at the head of $VOQ^s_{i,j}, s \equiv t - 1 + T_{i,j}(mod\, k), 1 \leq s \leq k$, to send a packet to the input i at switch S_{a_t} . This guarantees that packets leave the input buffers in the same order they have entered them.

A multiplexer M_j receives a list $L_{i,j}$ from the CCU for each connection from i to j configured by at least one of the switches actually in its active phase. $L_{i,j}$ is an ordered list containing the $u_{i,j}$ switches $S_{a_1},..,S_{a_{u_{i,j}}}$, through which a packet is sent in the current timeslot. The multiplexer M_j saves the lists $L_{i,j}$ in a FIFO list. When $L_{i,j}$ reaches the top of the FIFO list, M_j successively reads the packets from the output queues of the switches in the order $S_{a_1},..,S_{a_{u_{i,j}}}$. Then $L_{i,j}$ is removed from the FIFO list. This procedure ensures that packets keep their flow order also in the output queues and therefore in the whole switch.

We note that this architecture requires $O(N^2k^2)$ de-multiplexers and $O(N^2k^2)$ e-o converters.

D. Fully distributed architecture

This switch architecture is identical to the distributed architecture as shown in fig. 3 and fig. 4. The switching planes decide their switching configuration individually. Each switch runs a scheduling algorithm at the beginning of every timeslot. As a result the switch may either leave the current configuration intact or decide to reconfigure connections between a subset of inputs and outputs or reconfigure the whole switch. Because the concept of switching and forwarding phases does not apply to a fully distributed architecture, the $M_{i,j}$ needed in the distributed architecture are not calculated. The scheduling algorithm takes all $K_{i,j}$, $1 \leq i, j \leq N$ packets as an input.

Remark: We note that in the architectures presented here no component is required to work faster than line speed, i.e. a de-multiplexer and a switch forward at most one packet per timeslot and a multiplexer reads at most one packet per timeslot.

III. Scheduling algorithms and their packet delay bounds

A. Assumptions

In this section we propose different scheduling algorithms for the parallel switching planes and derive packet delay bounds in some cases. In the sequel we denote by $I_i(t_1, t_2)$ the number of packets arriving at an input i in the interval $[t_1, t_2]$. $O_j(t_1, t_2)$ is the number of packets destined for an output j in the interval $[t_1, t_2]$. If we used N parallel switches $S_1, .., S_N$ and connected the input i statically with the output $i + m(mod\, N)$ at switch S_m, every input would always be trivially connected to every output . Therefore in the following we assume $k \leq N - 1$.

Assuming leaky bucket flows and therefore avoiding oversubscription of any input or output, we require in the following

$$L_g \leq \alpha(t_2, t_1) + B \tag{2}$$

for $L \in \{I, O\}$, $0 < \alpha < 1$, $B > 0$, $1 \leq g \leq N$. In the following the problem to determine a switch configuration is solved by finding a matching of a bipartite graph (see [1]). We use the term matching opportunity to denote each occasion at which the scheduling algorithm is executed. Our first two scheduling algorithms belong to the class of maximal matching algorithms defined as follows:

Maximal matching: A policy applied by a scheduling algorithm is a maximal matching if at the end of any matching phase any packet buffered at any input i destined to any output j remains at the input only if during this matching phase some other packet has been transmitted from either input i and/or output j.

The first two algorithms can be applied to the simultaneous and the distributed architecture, whereas the third algorithm can only be applied to the fully distributed architecture.

B. Oldest Cell First (OCF)

The OCF maximal matching algorithm is defined in [2]:

We prove for the distributed architecture that a packet $P_{I,j}$ that arrives at time t will not be scheduled later than time $t + D$ where D will be calculated in the sequel. Suppose there exists a packet P arriving at input i at time t which is destined to output j and exceeds the delay bound D. Suppose further that P is the earliest packet to do so. The only packets which prevent P from being scheduled at the time $t + D$ either arrive at input i and/or are destined to output j. At the time t all packets that arrived before the time $t - D$ have already been scheduled by the choice of P. All packets that arrive after time t cannot block P from being scheduled by the definition of OCF. Therefore only packets that arrive in the time interval $[t - D, t]$ and either arrive at input i or are destined to output j compete with P. Using (2) the number of competing packets is estimated as follows:

$$\sum_{l=i \bigvee k=j} \alpha_{l,k}(t, t - D) \leq 2\alpha D + 2B. \tag{3}$$

Because a forwarding phase has M switching opportunities, we do not need more than $\left\lceil \frac{\alpha_{l,k}(t, t-D)}{M} \right\rceil$ switching phases to schedule $\alpha_{l,k}(t, t - D)$ packets. From (3) we derive an upper bound for the number of switching phases we need to switch all packets belonging competing with $P_{i,j}$:

$$\sum_{l=i \bigvee k=j} \left\lceil \frac{\alpha_{l,k}(t, t - D)}{M} \right\rceil \leq \frac{2\alpha D + 2B}{M} + 2N - 1 \tag{4}$$

Because in the interval $[t - D, t]$ there are at least $\frac{D}{R} - 1$ switching phases, in order to guarantee that all packets competing with $P_{i,j}$ will be switched before time $t + D$, it is enough to ensure that

$$\frac{2\alpha D + 2B}{M} + 2N - 1 \leq \frac{D}{R} - 1, \tag{5}$$

which for $M > 2R\alpha$ is equivalent to

$$\frac{2RB + 2MRN}{M - 2\alpha R} \leq D.$$

For the computationally simple case $\alpha = 1$ and $B = 0$ we derive from (1)

$$D \geq f(M) := 2N \frac{M^2 + MT}{M(k - 2) - 2T}, \tag{6}$$

for $M > \frac{2T}{k-2}$ and $k > 3$. Calculating the zeros of the derivative $f'(M)$ we find that $f(M)$ reaches its minimum at

$$M_0 = \frac{2T}{k - 2} + \frac{\sqrt{2kT}}{k - 2}, \tag{7}$$

such that

$$D := \left(\frac{k + 2}{(k - 2)^2 \sqrt{2k}} + \frac{4k}{(k - 2)^2} \right). \tag{8}$$

k\N	16	32	256	1024
4	0.04662	0.09325	0.74603	2.98415
8	0.00800	0.03200	0.12800	0.51200
15	0.00264	0.00529	0.04234	0.16938
31	-	0.00185	0.01483	0.05935
63	-	-	0.00601	0.02406
127	-	-	0.00263	0.01054
255	-	-	0.00121	0.00483
511	-	-	-	0.00228
1023	-	-	-	0.00109

TABLE I

UPPER DELAY BOUNDS AT THE INPUT FOR THE DISTRIBUTED ARCHITECTURE WITH THE OCF ALGORITHM

Table 1 shows values for D as calculated in (8) for $T = 2500$ and a timeslot length of 0.2 μs.

To calculate the input delay bound for the simultaneous architecture we note that during a switching phase maximally kM packets of a configured input - output connection are switched and that a new switching period starts every $M + T$ timeslots. Thus, instead of (4) and (5) we use the equations

$$\sum_{l=i \bigvee k=j} \left\lceil \frac{\alpha_{l,k}}{kM} \right\rceil \leq \frac{2\alpha D + 2B}{kM} + 2N - 1,$$

$$\frac{2\alpha D + 2B}{kM} + 2N - 1 \leq \frac{D}{M+T} - 1.$$

Arguing as above we obtain for $\alpha = 1$ and $B = 0$

$$2Nk\frac{2M^2 + MT}{M(k-2) - 2T} \leq D.$$

From (6) we see that the delay bound is k times larger in the simultaneous case than in the distributed case. To calculate the delay a packet can experience at the output, we apply the following lemma from [11]:

Lemma on output delay: For the simultaneous and the distributed architecture the following holds: If the maximal input delay is equal to D, the output delay is not larger than $kD + B$.

The *total_delay* a packet can experience at a switch is then

$$total\_delay \leq (k+1)D + B. \quad (9)$$

C. Longest Queue first (LQF)

The longest queue first algorithm [8] is defined analogously as OCF by using the queue length of the VOQs instead of a timestamp. We calculate a delay bound for the distributed and simultaneous architecture that holds uniformly for all maximal matching algorithms. Using the same notation as in the previous section, we see that only packets that have arrived in the interval $[t - D, t + D]$ can block the packet P from being scheduled before time $t + D$. Instead of (3) we obtain

$$\sum_{l=i \bigvee k=j} \alpha_{l,k} \leq 4\alpha D + 2B.$$

Arguing as in the previous section we derive delay bounds which for large k are more than twice as large as the bounds obtained with OCF.

D. Keep Connection Longest Queue first (KCLQF)

In each timeslot the algorithm leaves intact all connections for which there are packets in any of the corresponding VOQs. A modified LQF algorithm is applied to the remaining inputs and outputs. Instead of the queue length the algorithm takes the expected queue length after the configuration time of T timeslots as a weight. This is equal to the actual queue length minus the maximum number of packets that could be switched in the next T timeslots by all existing connections or those being actually configured. The approach chosen above does not allow us to derive delay bounds for the fully distributed architecture.

IV. SIMULATION RESULTS

The scope of the simulations is to understand the packet delay distributions for the various switch architectures.

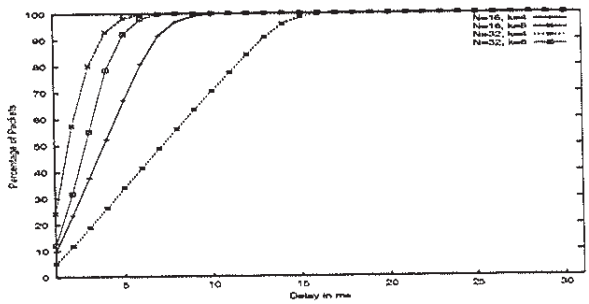

Fig. 6. The delay distribution for OCF for varying numbers of inputs/outputs N and varying number of switching planes k

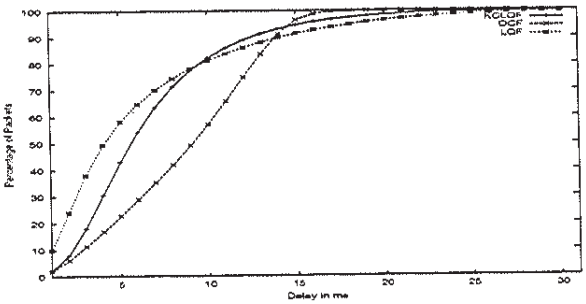

Fig. 7. The delay distribution for different scheduling algorithms with $N = 32$, $k = 4$

Simulations were run for the distributed architecture with the OCF and LQF algorithms and for the fully distributed with the KCLQF algorithm. No simulations were run for the simultaneous architecture because the performance analysis in section III suggests that the achievable

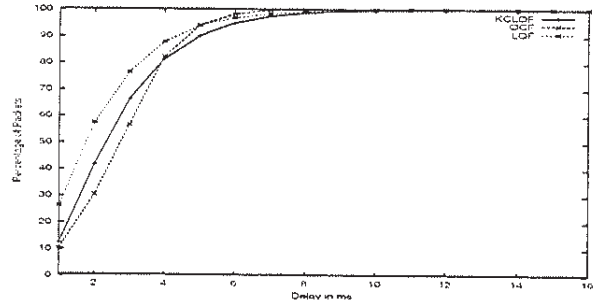

Fig. 8. The delay distribution for different scheduling algorithms with $N = 32$, $k = 8$

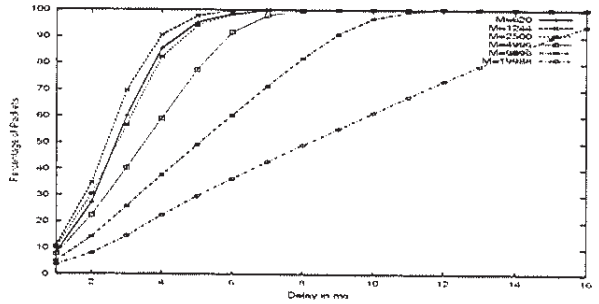

Fig. 9. The delay distribution for OCF as a function of M, for $N = 32$, $k = 8$

delays are much larger than those of the distributed architecture. The traffic arriving at each input was modeled as bursty as in [12]. The distribution of the traffic to the outputs was chosen either uniformly or according to a Zipf distributrion(see [3]). The traffic load at each input was close to 100% such allowing a comparison of the simulation results with the delay bound evaluation done for $\alpha = 1$ and $B = 0$ in section III. As in section II we assume a timeslot to be 0.2 μs and we set the switching equal to 0.5 ms ([7]) such that $T = 2500$. Increasing the number of switching planes does not always result in a decreased average delay. Past a point, the delay suffered by packets is drastically reduced at the input, but it is merely shifted to the output, since the output is receiving k packets and dequeuing only one per timeslot. Examples are shown for the OCF algorithm in fig. 6. Nearly all packets experience a delay that is significantly smaller than even the upper bound for the input delay as given in table I. It is even less compared to the *total_delay* given by (9), but much larger than the delays occuring at electronic switches. The figures 7 and 8 show that the OCF scheduling algorithm guarantees smaller delays than the LQF algorithm. The fully distributed architecture gives the largest delays. The better performance of the OCF algorithm can be expected as delay minimization is favored by the choice of the packet arrival time as the weight in the matching algorithm. The bad performance of the fully distributed architecture is due to the fact that connections with a high demand are maintained and prevent the configuration of connections with competing inputs or outputs. The performance of the distributed architecture depends on the the length of the forwarding period M. Fig. 9 shows that the packet delays are minimal if M is chosen between 620 and 1244. This is smaller than the value for $M_0 = 2500$ (see equation (7)) which minimizes the theoretic upper delay bound. All results hold with minor differences for both uniformly and Zipf distributed output destinations.

V. Conclusions

We considered the problem to design a switch architecture for switching technologies with large switching times. The idea of parallelism is proposed to compensate for the long switching times. Three types of architectures are discussed and upper packet delay bounds are established. Simulation results show that these delay bounds are much higher than the delay packets experience in practice. The experienced delays are however significantly larger than the delay bounds of existing electronic switches. In view of the delay requirements of realtime applications as voice, switches using the architecture proposed here might therefore only be used in the core network such that packets do travel through many of these switches.

References

[1] T. Anderson, S. Owicki, J. Saxe and C. Thacker, *High Speed Switch Scheduling for Local area Networks*, Proc. Fifth International Conference on Architectural Support for Programming Languages and operating Systems, Oct. 1992, Boston.

[2] A. Charny, *Providing QoS Guarantees in Input Buffered Crossbar Switches with Speedup*, Dissertation, Massachusetts Institute of Technology, 1998.

[3] I. Elhanany, J. Nir and D. Sadot, *A contention-free packet scheduling scheme for provision of quality-of-service in Tbit/sec WDM networks*,Optical networks magazine, issue 3, July 2000.

[4] D.K. Hunter, M.C. Chia, I. Andonovic, *Buffering in Optical Packet Switches*, Journal of Lightwave Technology, Vol. 16, No. 12, Dec. 1998.

[5] S. Iyer, A. Awadallah and N. McKeown, *Analysis of a packet switch with memories running slower than the line-rate*, IEEE Infocom 2000, Tel Aviv.

[6] S. Iyer and N. McKeown, *Making parallel switches practical*, IEEE Infocom 2001, Anchorage.

[7] A. Neukermans and R. Ramaswami, *MEMS Technology for Optical Networking Applications*, IEEE Communications Magazine January 2001, p.62 - 69.

[8] N. McKeown, V. Anantharam and J. Walran, *Achieving 100% Throughput in an Iput-Queued Switch*, IEEE Infocom 98, San Francisco.

[9] Y. Shani and N. Ben-Horin, *Advantages of smart photonic switches in enabling all-optical networks*, Lightwave, November 2000, p. 120 - 122.

[10] R. Ramasami and K. Sivarajan, *Optical Networks. A Practical Perspective*, Morgan Kaufmann Publishers, Inc, San Francisco , CA, 1998.

[11] S.S. Mneimneh, V. Sharma and K.Y. Siu, *On Scheduling Using Parallel Input-Output Queued Crossbar Switches With No Speedup*, IEEE Workshop on High Performance Switching and Routing 2001, Dallas.

[12] Z. Haas, *The "Staggering Switch Switch": An Electronically Controlled Optical Packet Switch*, Journal of Lightwave Technology, Vol. 11, No 5/6, May/June 1993, p. 925 - 936.