Reduced Complexity Input Buffered Switches

K. Kar† T. V. Lakshman∗ D. Stiliadis∗ L. Tassiulas†

† ECE Department
University of Maryland
College Park
Maryland, MD 20742

∗ Bell Labs
Lucent Technologies
101 Crawfords Corner Road
Holmdel, NJ 07733, USA

Abstract—**Input-Output buffered crossbars are popular building blocks for scalable high-speed switching because they require minimum speed-up of memory bandwidth. Scaling the design of crossbar switches to large capacities is limited by technology issues such as the reconfiguration of high-speed fabrics or power consumption. In addition, when crossbar architectures are used in the context of IP networks where packets are of variable size, they typically schedule and transfer fixed size *envelopes*. Incoming packets are fragmented into envelopes and this fragmentation can lead to, possibly large [1], loss of bandwidth and instability. This paper proposes a new method for switching variable sized packets over a crossbar switch that i) allows maximum utilization of switch bandwidth and avoids the fragmentation effect, and ii) allows designers to use much larger envelopes for transfering data over the fabric that minimizes the reconfiguration frequency of the fabric. Reducing the scheduling frequency makes implementation of complex schedulers practical and enables us to build ultra-fast switches incorporating optical technology that can provide bandwidth and delay guarantees.**

I. Introduction

The rapid growth of the Internet is creating an insatiable demand for bandwidth growth and the large scale deployment of high-speed access technologies, such as xDSL and cable modems, is likely to give further impetus to this trend. The deployment of Dense Wavelength Division Multiplexing (DWDM) technologies has expanded fiber transmission capacities to an extent that electronic routers remain as the main bandwidth bottlenecks. Consequently, there is need for building even faster routers. Moreover, the need to shift from a best-effort network to one providing Service Level Assurances (SLAs) requires routers and switches that support traffic management and protection mechanisms. This implies that switch fabrics used in routers must be able to provide bandwidth and delay guarantees in addition to being faster and faster.

Our main goal in this work is to show what are the main problems with current designs of high-speed fabrics and how one can design scheduling algorithms that can scale the capacity of high-speed electronic or even optical crossbars. We will also show, how by using appropriate scheduling algorithms, we can take advantage of recent developments in optical technology to develop high-capacity optical packet switches based on opto-electronic VLSI or Dragone Routers [9], [10], [11].

Let us consider a general model of a router or switch. The system consists of a set of port cards inter-connected through a central switching fabric. As the requirements for bandwidth and number ports increase dramatically, the number of port cards that a single switch/router must accomodate is increasing as well. It is most likely that port-cards will be distributed over several racks of equipment and their interconnection to the switch fabric will be either through serial high-speed electronic connections like Gigabit Ethernet or optical fiber connections.

A crossbar based architecture of this type, requires a central scheduler that controls the operation of the crossbar, by matching input and output ports for transmitting packets. There is a plethora of bibliography on applying this architecture in switches handling fixed size packets [7], [4], [5], [14], [22]. Something that does not become clear, however, from the high-level description of most of these architectures is the complexity of implementation when applied to very high capacity fabrics, as well as the communication overhead required between the port cards and the scheduler.

In order to avoid head-of-line blocking, most of these architectures are based on Virtual Output Queueing [7]. The central scheduler requires a complete knowledge of the state of the system. Or in other words, the scheduler must know whether a packet is waiting to be transmitted between any pair of input/output ports. Once the scheduler has collected this state information, it will i) match input and output ports, ii) program the crossbar switch and, iii) notify all input ports as to which packet to transmit to which output port. There are several issues that arise when one tries to implement such an architecture in a high-capacity crossbar. The rate of switching of the crossbar (or reconfiguration frequency) depends on the size of packets that are switched. Previous work has assumed that packet size is set equal to the minimum packet size of the network. In the case of IP networks this translates to 40 bytes. One can easily calculate that a switch fabric with 10Gbps ports must switch every 40ns and a fabric with 40Gbps ports must switch every 10ns.

Technology Issues

There are several problems that must be solved in such a design:

1. The communication between port cards and switch fabric must be done in the form of high-speed signals. Current technology choices are Gigabit Ethernet (PECL) or LVDS signals. The crossbar can either recover the clock of the signal and switch it digitally, or it can be a passive one that switches the signal without modification. In the first case, the central crossbar requires some limited memory to recover the digital signal and possibly switch it over a a parallel path. This function increases subtantially the power consumption of the switch to unacceptable levels. A Serializer/Deserializer (SERDES) device must be used in both the port cards and the switch fabric. State of the art CMOS technology requires at least 2.5 Watts of power for each 4 Gigabit-Ethernet Signals [**?**]. This means that a 64x64 switch fabric at 10Gbps will require at least 640 Watts of power just for the SERDES devices.

2. In the latter case, where the crossbar is passive, the transmitter and reciever devices must be re-syncrhonized every time the crossbar is switching. This clock re-synchronization operation takes usually several bit times. In the case of most common transcievers this time can be in the range of 10-20 ns or even higher. It can be easily seen that such an approach can not scale to very high-speeds (40Gbps or higher) where the crossbar must be switched every 10ns or less.
3. The central scheduler must be able to syncrhonize all the ports as well as the crossbar, so that they all switch at the same time. Because of the clock skews, one must account for extra margins to avoid loosing usefull data. As the switching frequency increases (i.e. 10 ns or less) this clock skew presents an additional design problem.
4. Recently, several algorithms have been presented that allow an input/output buffered crossbar to emulate an output buffered switch [2], [3], [6]. These algorithms require a speed-up of at least two, that when applied to high-speed fabrics it increase the switching frequency of the crossbar even farther. Therefore, for any real high-speed design this approach can not be usefull.

We can see from the above discussion, that scaling crossbar switches that use minimum size packets as their switching unit is almost technologically impossible or extremely expensive. In addition, to the technological problems, we have to consider a performance issue. Most previous approaches consider only issues related to the scheduling of fixed size packets, and do not address the problems caused by arriving traffic being composed of variable size packets with the local packet size distribution being generally not known.

Problems due to variable size packets

The incoming packet stream, in IP networks, is composed of variable size packets. However, crossbar switches use a fixed size unit of transfer, which we call *envelope*. This mismatch between arrival and transfer units requires packets to be fragmented into envelopes. A simple fragment-and-schedule packing of packets can lead to significant loss of fabric bandwidth when the packet size in not an integral multiple of envelope size. As an example, if the packet size is 1.5 times the envelope size and we use 2 envelopes to pack each packet, there is a 25% wastage in fabric bandwidth. If the packet size is half of the envelope size, then we have a 50% loss of fabric capacity. This problem was briefly addressed in [1] where it was suggested that increased speedup could be used to compensate for this bandwidth loss. If packets are always larger than envelopes, it can be easily seen that the maximum loss of bandwidth occurs when all the packets are of length $(L_e + 1)$, where L_e is the envelope-length. Then, as suggested in [1], we could use a $\frac{2L_e}{L_e+1} \approx 2$ times additional speedup to offset the bandwidth loss caused by mismatch of packet and envelope sizes. However, this is neither a satisfactory nor a general solution, since increasing the reconfiguration speed of the crossbar poses several implementation problems.

Figure 1 shows the average packet delays, obtained by simulation, for various envelope-sizes. We assume a 16x16 switch, using the 16 iterations of the iSLIP scheduling algorithm[7]. Assume that L_e denotes the envelope-size and L_p denotes the packet size. Packet size distribution is based on measurements from [8] with 60% of the packets being 44 bytes, 20% of the packets 552 bytes and the rest 1500 bytes. The minimum packet size is 44 bytes, denoted by $min(L_p)$ in the figure, while the average packet size is approximately 436 bytes, denoted by $mean(L_p)$ in the figure. Packet interarrival times are exponentially distributed and traffic is symmetric across all inputs and outputs. Delay is in terms of the transmission time of an average size packet on the input/output links.

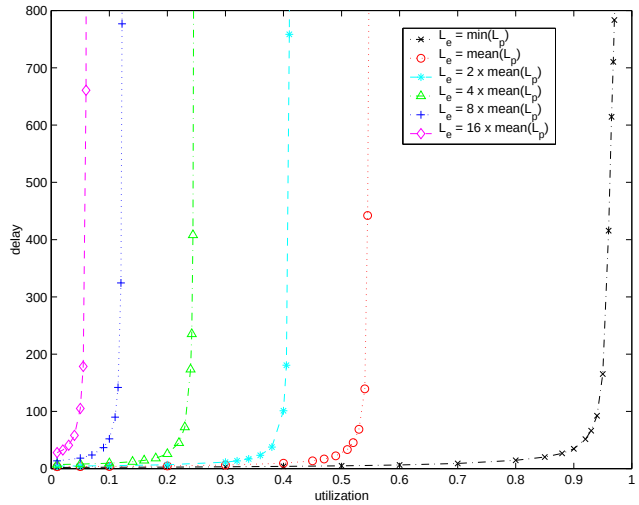

Fig. 1. Delay vs. Utilization for iSLIP for various envelope sizes

We can notice that contrary to the constant size packet case, for arbitrary packet sizes, the system may become unstable, particularly when the envelope size is much larger than the minimum packet size, as evident in Figure 1. From the figure, we see that when the envelope size is equal to the minimum packet size, we obtain almost 100% throughput, while when the envelope size is equal to the mean packet size or larger, the system becomes unstable at about 50% load or even less. It can be easily seen that if the speedup is κ and the link speed is r, and all the packets are of the same length L_p, the maximum load that can be supported is $\frac{\kappa L_p}{L_e} r$, for $L_e > L_p$, and $\frac{\kappa L_p}{\lceil \frac{L_p}{L_e} \rceil L_e} r$ for $L_e < L_p$.

Envelope size choice and tradeoffs

To reduce the bandwidth loss due to fragmentation, especially when the packet size distribution is bimodal between small and large packets as is often the case, the envelope size must be matched to the predominant small packet size. For IP networks, which have a large number of small packets due to TCP acknowledgments [8], the optimal envelope size will be 40 or 64 bytes (for Ethernet interfaces). However, for a very fast switch fabric, as we discussed earlier, such small envelope sizes result in almost impractically high scheduling and crossbar reconfiguration frequencies.

However, we can easily notice that if the envelope size were 16 times larger, i.e. 1K bytes, then we have 640ns (or 320ns with two times speed-up) for scheduling and reconfiguration. This makes it far more practical to implement sophisticated scheduling without pushing technology limits. The problem with large envelope sizes is the bandwidth loss evident from Figure 1. If an envelope packing and scheduling scheme were developed to

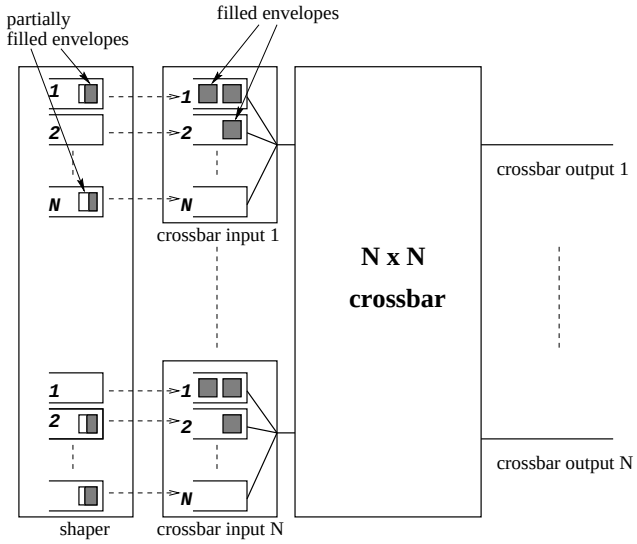

Fig. 2. A scheme for achieving 100% throughput in input queued switches for variable packet sizes

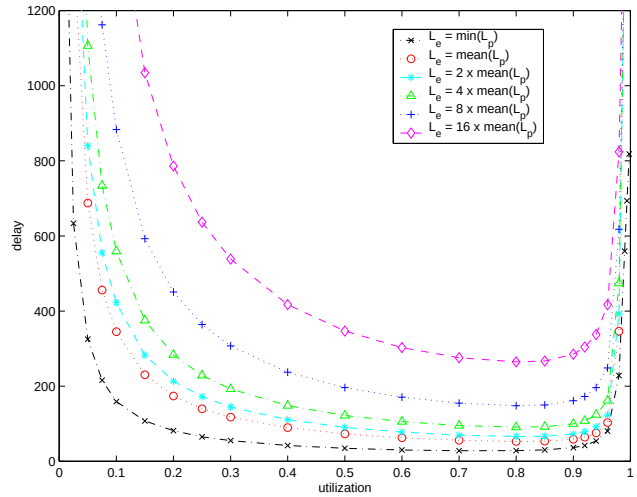

Fig. 3. Delay vs. utilization plot for the scheme shown in Fig 2

permit the use of large envelopes without bandwidth loss and without any practically significant degradation in QoS guarantees, then a major constraint in the design of very-fast crossbar switch fabrics would be removed. This is the key problem addressed in the paper.

Our approach

Based on the above discussion, what we would like to have in ultra-fast crossbars is the following:

1. Reconfiguration frequency that is independent of the switch bandwidth and saficiently long to allow the use of efficient scheduling algorithms and advanced transmission technology.
2. Efficient bandwidth management for variable size packets, without bandwidth loss and with QoS guarantees.
3. Power and space efficiency.

II. Achieving maximum throughput with large envelopes

In this section, we outline the basic idea of how we can avoid the problem of bandwidth loss due to the size mismatch between envelopes and packets, as described in the last section. This technique allows us to build fabrics that switch large envelopes, and they can thus use passive or optical crossbars. As described earlier, use of passive or optical crossbars results to lower complexity, higher capacity and lower power consumption. We develop these ideas further in the next section to design algorithms that can provide bandwidth and delay guarantees.

Note that loss of bandwidth occurs when some partially filled envelopes are transferred over the fabric. These correspond to the last fragment of a packet. The first step to prevent this loss then is to allow the packing of multiple packets into an envelope. This may require that a packet be fragmented at an arbitrary position to pack into an envelope and we assume this for the rest of the paper. Thus, if $L_p = 1.5L_e$, then we can fit 2 packets in 3 envelopes, and if $L_p = 0.5L_e$, we can fit 2 packets in 1 envelope.

We can avoid loss of fabric bandwidth by using the scheme shown in Figure 2. Here, packets arrive to the shaper and fill envelopes. Partially filled envelopes wait at the shaper until the arrival of packets that completely fill them. Filled envelopes become eligible for transfer across the fabric and are queued at the switch input ports till the scheduler makes them eligible for transfer. Thus all the envelopes scheduled across the switch are full envelopes. Figure 3 shows the avg delay vs. utilization plots for this scheme for various envelope sizes, when the fabric scheduler is iSLIP [7], and all other conditions are similar to those for Figure 1. As expected, the scheme gives 100% throughput for all the cases.

However, this simple approach has an obvious drawback: a packet may have to wait for a very long time (even for ever) in the shaper for the corresponding envelope to be filled. This explains the high delays seen in Figure 3 at low loads (under heavy load, most envelopes are immediately filled up, and thus do not need to wait at the shaper for long). When a particular port is lightly loaded we need to release partially filled-envelopes to prevent delay accumulation. However, these partially filled envelopes use bandwidth that can be used by other heavily loaded ports and in general cause unnecessary instability. Consider the situation where only two input ports have traffic destined to the same output port. Assume that port one is loaded 100% and that port two is lightly loaded (less than 25%). A fair scheduler like iSLIP or PIM gives equal opportunity to both ports. Since envelopes from the lightly loaded port are only partially full there is a large loss of capacity that could have been utilized by the loaded port.

A possible scheme for preventing delay accumulation is to incorporate a time-out for partially filled envelopes. Expiration of this timer causes the shaper to release partially filled envelopes. An intelligent setting of this timer is possible if the second-order statistics of the traffic is known. Without traffic assumptions, adaptive mechanisms are needed to configure the time-out value and it is not clear how this can be done. The main drawback of this mechanism is that it can not guarantee stability under all traffic patterns.

An alternative method to prevent delay accumulations, without using timeouts, is to use “excess bandwidth” to service par-

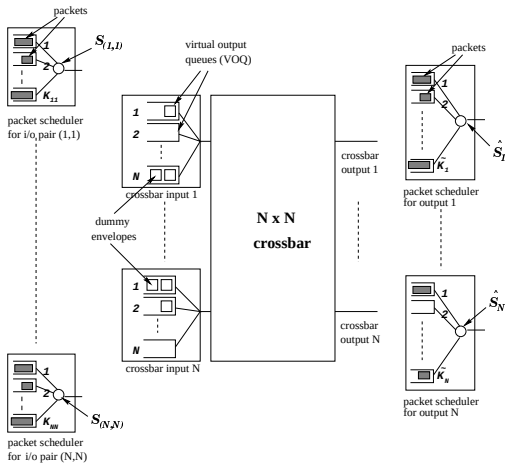

Fig. 4. The overall architecture of the switch for achieving bandwidth and delay guarantees

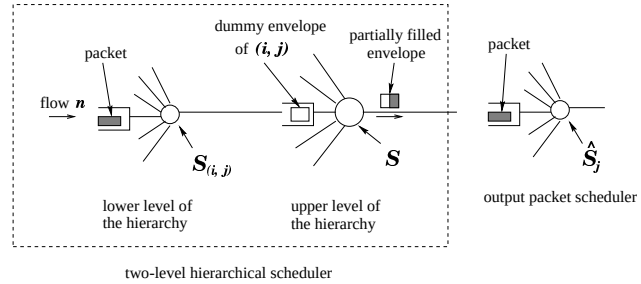

Fig. 5. Scheduling model of the architecture. The two-level hierarchical scheduler models the transmission of the packets over the switch fabric. The output packet scheduler models the transmission of the packets on the output links, after reassembly.

tially filled envelopes, i.e., when no switch input port has completely filled envelopes then the switch bandwidth is used to service partially filled envelopes waiting at the shapers. We use this idea later in the paper to reduce average delays.

III. Achieving bandwidth and delay guarantees

In this section, we first describe a switch architecture that can provide delay and bandwidth guarantees to the individual flows. We then derive a scheduling model for this switch architecture and use it to prove delay bounds for individual flows as a function of the scheduler properties such as service burstiness, and worst-case fairness index. Our goal in proving these bounds is to identify properties that the individual schedulers in our switch architecture must satisfy so as to enable the switch to provide good delay bounds despite using large envelopes. We do not give a specific choice of schedulers in this section. Specific scheduler choices so as to achieve delay bounds independent of the number of switch ports are given in the next section. Since providing delay guarantees requires that flows have bounded burstiness, we will assume that flows are leaky-bucket constrained, although a different measure of burstiness can be used as well.

A. Switch Architecture for Bandwidth and Delay Guarantees

Figure 4 shows the basic architechture that we consider. The switch has a crossbar fabric, and does virtual output queueing. For each input-output pair (i, j), there is a dedicated packet-scheduler $S_{(i,j)}$ (thus for a $N \times N$ switch, there will be N^2 such schedulers). The scheduler $S_{(i,j)}$ maintains separate queues for each individual flow between input i and output j, where the packets of the flow are buffered on arrival. When the crossbar arbiter allots a slot for the transfer of an envelope of (i, j) across the switch fabric, then the scheduler $S_{(i,j)}$ is responsible for scheduling an envelope-length of packets across the switch in that slot, choosing packets from the flows between input i and output j. Multiple packets can be transferred in a single envelope-slot, to reduce bandwidth wastage. The N^2 virtual output queues (VOQs) of the switch do not buffer real packets, but are used to buffer *dummy* envelopes. These dummy envelopes, each of length L_e, are just logical entities which are inserted into the VOQs in such a way that whenever the scheduler $S_{(i,j)}$ is backlogged with packets, the VOQ for (i, j) also remains backlogged with these dummy envelopes. Thus the presence of a dummy envelope in the VOQ for (i, j) indicates, to the crossbar arbiter, that there are some packets between input i and output j waiting to be scheduled across the crossbar fabric. The crossbar arbiter makes its scheduling decisions based on these dummy envelopes. Now let the crossbar arbiter decide to transfer a dummy envelope of (i, j) during the interval $[t, t + \tilde{T}_e]$, where $\tilde{T}_e$ is the length of a slot required to transfer an envelope of length L_e across the crossbar (if the crossbar has a speedup of κ, and the input/output link-speed is r, then $\tilde{T}_e = \frac{L_e}{\kappa r}$). Then, instead of transferring the dummy envelope in its scheduled slot, the scheduler $S_{(i,j)}$ is used to transfer real packets in that slot. Note that $S_{(i,j)}$ serves at a constant rate κr during that slot, and hence can transfer packets upto a total length of L_e during the slot. However, the slot may not be utilized completely, if all the queues at $S_{(i,j)}$ remain empty for some interval during the slot. Note that in the process of transferring packets across the switch fabric, the packets can be fragmented. Therefore, after an entire packet has been transferred, it is reassembled (the reassembly buffers are not shown in the figure), and placed in the per-flow queues at the corresponding output. The output packet scheduler, $\hat{S}_j$ (there are N of them, one for each output), is responsible for scheduling the packets of flows destined for output j, from the per-flow queues at the output onto the output link.

B. Scheduling Model for Switch Architecture

An abstract model of the overall switch scheduling system, as seen by the packets of a flow n between input i and output j, is shown in Figure 5. Here the crossbar is modeled as a single server S. The server is associated with N^2 envelope-queues, and serves up to maximum of N envelopes at the same time. During the time when an envelope of (i, j) is being served by S, the server $S_{(i,j)}$ serves packets of flows going from input i to output j.

As mentioned before, whenever $S_{(i,j)}$ is backlogged, the envelope-queue for (i, j) at S remains backlogged too. However, the converse is not necessarily true. When S is "busy serving the empty portion of an partially filled envelope" of (i, j), the envelope-queue for (i, j) is backlogged at S, but $S_{(i,j)}$ is not.

C. Delay Analysis and Required Scheduler Properties

Based on the above architecture we can analyze the end-to-end delay properties of a flow that goes through such a switch fabric. Interested readers are referred to [24] for a complete analysis. In our analysis we use the notion of Worst-Case Fairness Index (WFI), that represents the maximum time before serving two packets of a given flow [12]. We also use the theory presented in [13] and the notion of the Service Burstiness Index (SBI).

Assume that a flow n is leaky bucket shaped with burstiness parameters (σ_n, ρ_n). We state without proof the following Theorem:

Theorem 1: The total delay of a packet of flow n in the entire switch is upper-bounded by

$$\frac{\sigma_n + \gamma_n + \hat{\gamma}_n + L_{n,max}}{r_n} + \frac{\alpha_{(i,j)}}{r_{(i,j)}}$$

where $\alpha_{(i,j)}$ is the WFI guaranteed to the envelope-flow of (i,j) by the server S, and γ_n and $\hat{\gamma}_n$ are the SBIs guaranteed to flow n by the servers $S_{(i,j)}$ and $\hat{S}_j$, respectively, and $L_{n,max}$ is the maximum length of a packet of flow n.

In other words, the above theorem proves that the delay bounds that the switch can guarantee depend on the WFI and SBI of the schedulers. Typically, if the scheduling algorithms are correctly selected the term $\alpha_{(i,j)}$ will be dependent on the envelope size L_e, while γ_n and $\hat{\gamma}_n$ will be independent of L_e. Thus, typically the envelope size will influence the overall delay bound only through the term $\alpha_{(i,j)}$.

IV. Delay Guarantees under Various Switch Scheduling Algorithms

In this section, we use the results from the previous section, to compute packet delay bounds for several switch scheduling algorithms which have been shown in the literature to provide bandwidth guarantees to the envelope-traffic.

A. TDM scheduling

In [5], the authors propose a simple method, similar to time division multiplexing, for providing bandwidth guarantees. In this method, transmission time is split into *frames* of equal size and an integral number of envelopes can potentially be transmitted during each frame. Bandwidth for an aggregate flow between an input-output pair is guaranteed by reserving adequate number of envelopes for that flow in each frame (the procedure for computing the schedule is in [5], [14]). A new static schedule is calculated every time a new flow arrives or departs from the switch. If an input-output pair has no envelopes to transmit during its assigned slot in the frame, then the idle capacity is given to other ports using an iterative matching algorithm.

Treating the switch as a server, it is easy to show that the WFI guaranteed by the switch to the flows is $2F$ (F is the frame size) [17]. Corollary 1, gives the delay bound on a packet by replacing $\alpha_{(i,j)}$ by $2F$ and γ_n by the appropriate SBI of the packet scheduler $S_{(i,j)}$. Thus the delay of a packet will depend on the frame size. The frame size is determined by the minimum bandwidth allocation over the fabric and the number or fabric ports since every port must be able to receive at least the minimum bandwidth. Then, the delay bound is of the order of $O(N \times L_e)$ where N is the number of ports in the fabric.

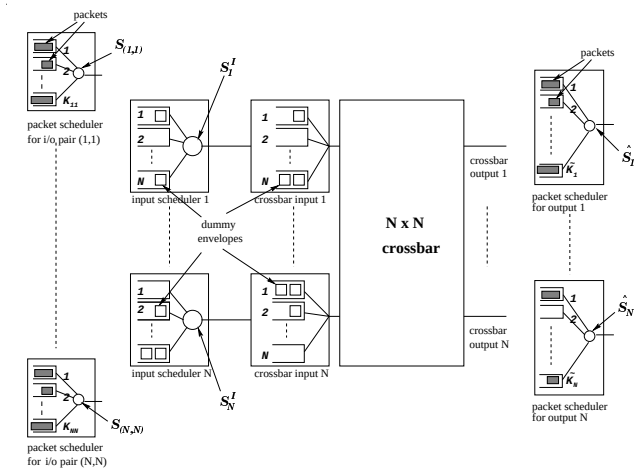

Fig. 6. The overall architecture of the switch for achieving bandwidth and delay guarantees on the packets for the output queueing emulation algorithms

Therefore, although the TDM approach provides a mechanism for guaranteeing delay bounds in an input/buffered switch with variable size packets, these bounds can be rather large since they depend both on the number of ports of the scheduler and the envelope size (i.e. for larger, and higher capacity fabrics, the delay bounds will increase). Ideally, we would like to design a scheduling algorithm where the delay bounds will be independent of the fabric size and will not grow substantially with increasing envelope size.

B. Output queueing emulator

Chuang et. al. in [2], and Stoica & Zhang in [3] presented switch scheduling algorithms which can mimic output queueing switches using an input-output buffered crossbar with a speed-up of 2. These algorithms apply only to constant size envelope switches, and in addition, they require that corresponding output buffered system that they emulate must be *monotonic* (i.e. future arrivals will not change the order of transmission of packets already in the queues).

Our scheme extends this approach for the case of variable size packets, as shown in Figure 6. In order to work correctly, the switch algorithms in [2], [3] require that at most one (dummy) envelope arrives at each input of the crossbar in a single envelope-time. To enforce these criteria, we introduce a scheduler at each input, as shown in Figure 6, such that it serves exactly one envelope each envelope-time. Each input scheduler maintains N envelope-queues, one for each output. The envelope-queue for (i,j) at the corresponding input scheduler is kept backlogged with envelopes whenever the packet-scheduler $S_{(i,j)}$ is backlogged with packets. An envelope for (i,j), on being scheduled by an input scheduler at input i, is placed in the queue for (i,j) at the crossbar inputs. When the crossbar allots a slot for the transmission of a envelope of (i,j), then instead of the transferring the envelope, the packet-scheduler $S_{(i,j)}$ is used to transfer packets across the switch in that slot, as we have described in the last section.

If we assume that the schedulers $S_{(i,j)}$ and $\hat{S}_j$ follow WFQ

[15], the schedulers S_i^I follow Shaped VC [19], and the scheduling algorithm that the crossbar emulates is WFQ, then we can derive and delay bound for any leaky-bucket shaped flow that is independent of the size of the switch N [24].

Corollary 1: If the packets achedulers at the input and the output follow WFQ, the input schedulers follow Shaped VC, and the crossbar emulates WFQ, then the total delay of a packet of flow n in the system is upper-bounded by

$$\frac{\sigma_n + L_{(i,j),max} + L_{j,max} + L_{n,max}}{r_n} + \frac{4L_e}{r_{(i,j)}}$$

The delay bound is independent of N, and the envelope-size affects the delay bound only through the term $\frac{4L_e}{r_{(i,j)}}$. Note, $\sigma_n \geq L_{n,max}$, where $L_{n,max}$ is the maximum packet size for flow n. Let $L_{n,max} = L_{(i,j),max} = L_{j,max} = L_p$. Let us assume that 100 flows of equal rate are multiplexed between input i and output j. Thus $r_{(i,j)} = 100r_n$. Thus the term $\frac{4L_e}{r_{(i,j)}}$ becomes comparable to the others in the delay bound only when L_e is equal to $100 \times L_p$ or more. Thus we can increase L_e (and hence reduce the scheduling frequency) considerably, compared to the maximum packet size, without affecting the overall delay bound significantly. Also, L_e can be made fairly large, still keeping the delay bound on a packet of the same order of magnitude as $(\frac{\sigma_n}{r_n} + \frac{L_{j,max}}{r})$, the delay bound for a packet of flow n in a completely output-buffered system.

V. Conclusions

We observe that fast scheduling is a key constraint which limits the scalability of crossbar switches which otherwise have appealing properties. We also observe that although optical switches have excellent properties in terms of power consumption and space requirements, they have long re-configuration times. In order for this technology to be useful, the scheduling algorithms used must account for a slow reconfiguration frequency. We also note that some of the recently proposed fabric schedulers do not satisfactorily take into account the fact that arriving packets can be of variable sizes and so cause bandwidth loss in the fabric. Motivated by these observations, we propose schemes where the scheduling decisions of the crossbar can be reduced considerably, yet maintaining delay and bandwidth guarantees, which for all practical purposes, are almost similar to that of output queued switches. Since slowing down the frequency of scheduling decisions also implies slowing down the reconfiguration frequency, our schemes enable the use of switch fabrics incorporating optical technology.

References

[1] D. Stephens and H. Zhang, "A distributed scheduling algorithm in high speed input queued crossbars," in *Proceedings of IEEE INFOCOM*, April 1998, pp. 282–290.

[2] S-T. Chuang, A. Goel, B. Prabhakar, and N. Mckeown, "Matching output queueing with a combined input-output queued swith," Tech. Rep. CSL-TR-98-758, Stanford University, 1998.

[3] I. Stoica and H. Zhang, "Exact emulation of an output queueing switch with a combined input output queueing switch," in *Proceedings of IWQoS*, 1998.

[4] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, "The Tiny Tera: A packet switch core," in *Proceedings of Hot Interconnects V*, August 1996.

[5] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High speed switch scheduling for local area networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, pp. 319–52, November 1993, Also as Digital Systems Research Center Technical Report, No. 99.

[6] A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in input buffered crossbars with speedup," in *Proceedings of IWQoS*, 1998.

[7] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, April 1999.

[8] Cooperative Associatian for Internet Data Analysis, "Packet size reports," http://www.caida.org.

[9] A. V. Krishnamoorthy and K.W. Goosen, "Optoelectronic-VLSI: Photonics integrated with vlsi circuits," *IEEE Journal in Selected Topics in Quantum Electronics*, vol. 4, no. 6, pp. 899–912, 1998.

[10] C. Dragone, C.A. Edwards, and R.C. Kestler, "Integrated optics nxn multiplexer and silicon," *Photonics Technology Letters*, vol. 3, no. 10, pp. 896–899, 1991.

[11] D.R. Hjelme, H. Story, and J. Skaar, "Reconfigurable all-fiber all-optical crossconnect node using synthesized fiber bragg gratings for both demultiplexing and switching," in *Optical Fiber Communication Conferece (OFC)*, February 1998.

[12] J.C.R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Proceedings of ACM SIGCOMM'96*, Palo Alto, CA, August 1996, pp. 143–156.

[13] R.L. Cruz, "Service burstiness and dynamic burstiness measures: A framework," *Journal of High Speed Networks*, vol. 1, no. 2, pp. 105–127, 1992.

[14] T. Rodeheffer and J.B. Saxe, "Smooth scheduling in a cell-based switching network," Tech. Rep. 150, Digital Systems Research Center, http://www.research.digital.com/SRC, February 1998.

[15] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case," *IEEE/ACM Transactions on Networking*, June 1993, pp. 344-357.

[16] L. Zhang, "Virtual Clock : A new traffic control algorithm for packet switching networks," *ACM Transactions on Computer Systems*, vol. 9, pp. 101-124, May 1991.

[17] D. Stiliadis and A. Varma, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms," in *Proceedings of IEEE INFOCOM '96*, pp. 111–119.

[18] J.C.R. Bennett and H. Zhang, "WF^2Q: Worst-case fair weighted fair queueing," in *Proceedings of INFOCOM 96*, March 1996, pp. 120–128.

[19] D. Stiliadis and A. Varma, "A general methodology for designing efficient traffic scheduling and shaping algorithms," in *Proceedings of IEEE INFOCOM'97*, April 1997.

[20] D. Stiliadis and A. Varma, "Efficient fair queueing algorithms for packet switched networks," *IEEE/ACM Transactions on Computers*, vol. 6, no. 2, pp. 175–185, 1998.

[21] N. Duffield, T.V. Lakshman, and D. Stiliadis, "On adaptive bandwidth sharing with rate guarantees," in *Proceedings of IEEE INFOCOM 98*, April 1998, pp. 1122–1130.

[22] N. McKeown, V Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proceedings of INFOCOM'96*, March 1996, pp. 296–302.

[23] "Backplane Transceivers", http://www.vitesse.com

[24] K. Kar, "Scheduling of Variable Size Packets in Input Queued Switches," MS Thesis, University of Maryland