The Tetris Switch

Stephen Suryaputra and Joseph D. Touch
USC/ISI
Marina del Rey, CA, USA
surya@isi.edu and touch@isi.edu

Joseph A. Bannister
Aerospace Corp.
El Segundo, CA, USA
joseph.a.bannister@aero.org

***Abstract*—This paper presents a packet switching architecture that replaces conventional random-access queues with a new kind of FIFO queues called variable speed conveyor (VSC) queues as part of a new contention resolution mechanism inspired by the Tetris video game. This paper presents the results of both quasipoission non-bursty and pareto bursty simulation analyses comparing the switch to a virtual output queued (VOQ) switch, as well as details of the architecture and a discussion of how it can be implemented in either electronics or optics. The switch achieves approximately 95% of the throughput of a VOQ switch under bursty and non-bursty loads, using as little as 100 kb of buffering under typical Internet distributions of variable-length packets.**

Keywords-optical; switch; packet; throughput; shifting

I. INTRODUCTION[1]

Random-access buffering in electronic switches can be viewed as a contention resolution mechanism, which works well and can achieve near 100% throughput. Unfortunately, it is not currently feasible to implement random-access buffers in the optical domain, therefore optical switches cannot achieve the same level of throughput performance as electronic ones. Our goal is to develop a packet switch design that closes that throughput gap. This paper presents the design of a packet switch that can time-shift packets in either the electronic or optical domains, called the Tetris[2] switch. We evaluate its throughput using simulation for multiple switch sizes, packet size distributions, and uniform switching matrix, and show that it can achieve performance approaching that of conventional electronic routers using virtual output queues (VOQs), which are currently the standard for random-access buffered switches.

II. TETRIS SWITCH

The Tetris switch performs contention resolution using packet time-shifting, rather than random-access buffering. To simplify the design, only forward time-shifts (*i.e.,* packet acceleration) are considered. Hence the switch is named after the Tetris game, which has a mode to accelerate the arrival of blocks to the bottom of the game window (*i.e.*, hitting the "space bar"). Shifting is possible only if there is room. If the input line is heavily loaded, there isn't much room. A packet can only be shifted up to the tail of the previous packet on the same input line. This is the same problem as HOL blocking in electronic switches. Thus, we use the same solution but without random-access buffering by *splitting traffic based on the output port*. Figure 1 illustrates this point. The top picture shows the packets lining up back to back on all four input ports. The bottom picture shows only the packets that are destined to an output port. When packets destined to the other output ports are eliminated, shifting forward is possible.

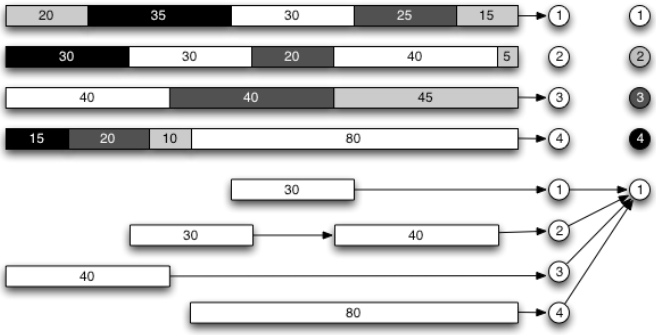


Figure 1: Illustration of the benefit of splitting traffic based on the output port.

The Tetris switch operates on a single electronic channel or optical wavelength and has several components:

- Input port demux that splits traffic based on the output port and informs the switch scheduler about the packets.
- Output traffic mux that combines packets from each input destined to the same switch output port. The combined output is back-to-back packets destined to a single output port. It is composed of a set of variable speed conveyor (VSC) queues together with a coordinated VSC queue delay controller and a simple passive muxing element.
- Variable speed conveyor (VSC) queues, which are a configurable set of switched delay lines (SDLs) [17, 18] using slow light buffers or electronic shift lines to implement controllable FIFOs. This FIFO isn't a true FIFO, but rather is a VSC queue, because it can be sped up or slowed down, but never stops shifting packets out (to a packet dump if no other space is available) [1].

[1] This material is based upon work supported by the U.S. Air Force, MILSATCOM Systems Wing SMC/MCX under the National Science Foundation Grant No. CNS-0626788 and the CIAN NSF ERC Grant No. EEC-0812072. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the Air Force.

[2] The term "Tetris" is a registered copyright of Tetris Holding, LLC. The term herein is used as shorthand for "Tetris-inspired", and has no direct relation to the game of the same name.

- VSC queue delay controller, which is an electronic control block that dynamically configures the speeds of the VSC queues.

The input port demux can be implemented optically using a 1×*N* unbuffered optical switch that is equipped with optical header processors, subcarrier multiplexing, or separating the header and payload on separate wavelengths [2]. The goal of this work is designing and evaluating the architecture; therefore, the detail on how to implement optical header processing is beyond our current scope.

The output traffic mux can be implemented optically using an *N*×1 unbuffered optical coupler preceded by a set of VSC queues. The VSC queues enable packets that cannot be shifted to continue travelling until they come to the mux, where they are dropped when contention exists.

Figure 2 shows a complete 3×3 switch based on these basic components.

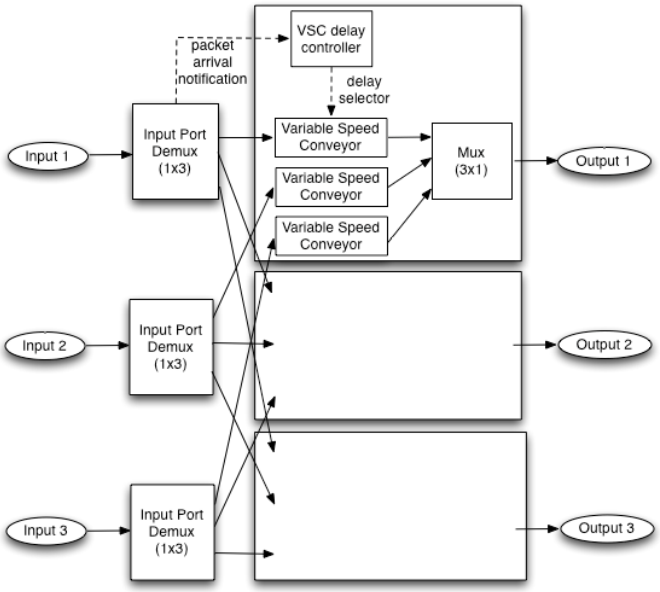


Figure 2: 3×3 Tetris switch architecture

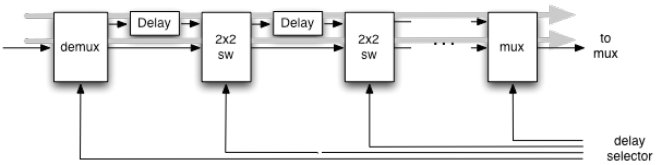


Figure 3: Implementation of a Variable Speed Conveyor (VSC) queue. The upper shaded line is the slow-path and the bottom shaded line is the fast-path.

The VSC queue delay has two internal paths, each with a different propagation speed. The first one is the *slow-path* consisting of slow-light buffers or electronic shifts. The second one is the *fast-path* consisting of fiber lines or electronic wires without delay. Shifted packets travel the fast-path while the other packets travel the slow-path. The VSC queue is illustrated in Figure 3. The slow-path is the default path until the VSC queue controller decides to shift the packets. Once the packets have been shifted, they stay on the fast-path until they arrive at the mux input.

The VSC queue controller has two phases: *look-ahead* and *shift*. In the look-ahead phase, it keeps track of the information about the packets in the set of VSC queues associated with a single output port. The collecting starts when the first packet enters the VSC queue. In the shift phase, the controller accelerates the packets forward. We can shift those packets because a VSC queue contains only in-flight packets from a single input port to a single output port.

Both phases correspond to regions of the VSC queue. The total delay in the VSC queue is divided into two equal-sized regions: the look-ahead region and the Tetris region. The look-ahead region is where the VSC delay controller examines the packet headers, determines what shifting is desired, and initiates shifting. The Tetris region is where the accelerated packets are shifted into, allowing packets to line up properly for the mux.

Packet scheduling in the VSCs occurs in rounds or batches, where there are gaps between the rounds. Each round is scheduled separately, and this reduces the complexity of the VSC delay controller. When a packet is still in the look-ahead region it initially travels the slow-path. Once the first packet arrives at the end of the look-ahead region, the controller shifts all look-ahead packets to the fast-path. This includes other packets that are still in the middle of the look-ahead region, and this operation shifts packets into the head of the input line to the mux. The available space for shifting is the Tetris region. Figure 4 shows both look-ahead and shift phase in separate diagrams. The top diagram shows the collecting period where packets destined to output port 1 are collected. In this example, packet A is the first to arrive at the end of the look-ahead region. This triggers the shifting operation. The bottom diagram shows the regions after shifting. The look-ahead region is freed (*i.e.*, containing no or fewer packets), and that region then becomes the next Tetris region if there is no spill and there is a packet that arrives at t15.

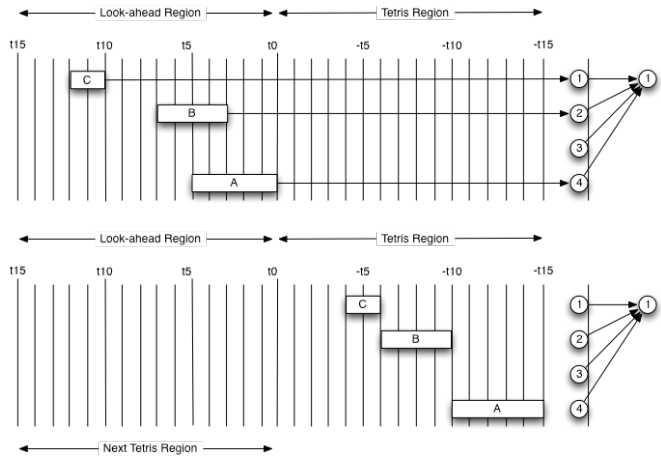


Figure 4: The two regions and phases of the switch scheduler.

This shifting algorithm is *earliest fit first*. It is defined in more detail as follows:

1. The controller keeps information on packets in the look-ahead region as an ordered set, sorted by arrival time. There is one in each controller, *i.e.*, one set per output port.
2. If the packet is first in the set, initialize the free space (F_S) to the size of the Tetris region in bytes. The number of bytes in the free space can be calculated from the delay (region size) and the port speed. Initialize the shifted time (T) to now + look-ahead time. This commences the operation of a round or batch of processing.
3. At time T:
   - 3.a. Initialize F_T to "now", *i.e.*, the time when the shifted packet will hit the switch port.
   - 3.b. Pick a packet from the set with the earliest arrival time.
   - 3.c. If the packet does not have any direct predecessor (*i.e.*, an earlier packet with the same input and output port that is still in the buffer) then:
     - 3.c.i. If the packet fits into F_S, shift the packet to F_T and subtract the length of the packet from F_S. Set F_T to the tail of this shifted packet.
     - 3.c.ii. Else (the packet does not fit), pick the next packet (if any) and repeat step 3.c.
   - 3.d. Else (there is a previous packet):
     - 3.d.i. Check if the tail of the previous packet is earlier than F_T (*i.e.*, the earlier packet is not blocking this packet). If yes, then go to step 3.c.i.
     - 3.d.ii. Else, pick the next packet, if any, and repeat step 3.c.
4. If there are still any packets in the set, drop those packets with arrival time less than the F_T. This happens to packets that arrive just before the end of the last shifted packet.
5. If there are still any packets in the set, re-initialize F_S and schedule another shifting operation (step 3) at time T (now + look-ahead time). The remaining packets in the set will be shifted along with other packets that may arrive between now and T.
6. If there isn't space to shift the packet forward, let the packet arrive at the mux at its usual time. It will be contending with the shifted packet, and one of them (randomly selected) will succeed.

We expect this switch can deliver close to 100% throughput in a uniform switching matrix regardless of the packet size distribution, provided the distribution of packets is similar across all input and output ports. For an $N \times N$ switch, the load from an input to an output port is $1/N$. The look-ahead delay is D bytes. The expected number of bytes from one input port to an output port is D/N. The expected total number of bytes destined to an output port from all input ports is D/N bytes × N ports = D bytes, which is also the Tetris region size. Thus the switch should not fall behind and will deliver close to 100% throughput.

There will be short-term variations in the input load that translate into the number of bytes from an input port to an output port being greater than D/N. The spill steps 4 and 5 handle this case by dropping the packets that miss the opportunity to shift and by reducing the size of the next (*i.e.*, subsequent) Tetris region. The number of bytes on the same input and output port pair will eventually be less than D/N. In the long term, close to 100% throughput should be achievable.

III. Performance Evaluation

We developed a simulator to validate the performance of the Tetris switch, and our analyses focus on the result of a 32×32 switch under both quasipoisson (non bursty) and pareto on/off (bursty) arrivals. Quasipoisson is a modified poisson arrival process that will not put overlapping packets on the same input port, and is thus more appropriate for switch analysis. Pareto on/off arrivals have been used to simulate higly variable arrival processes, including Web traffic [3]. The shape parameter α is set to 1.2 according to these findings. The mean burst size is 12500 bytes, the same burst size used for performance evaluation of optical burst switching [4].

Three packet length distributions are examined: fixed, exponential, and bimodal. For bimodal, 80% of the packets are 40 bytes long and the rest are 1500 bytes, based on recent observations [5]. The results presented in this section are mainly for the more realistic bimodal packet size distributions. Different packet size distributions will be noted explicitly when discussed.

The output port selection for quasipoisson arrivals is configured to be uniformly distributed. For pareto arrivals two variants are examined. In the first variant, called pareto-n, the output port is uniformly selected at the start of each burst and stays the same until the burst ends. For the second variant, called pareto-r, the output port is uniformly selected anew for each packet. From an output port point of view, pareto-n is more bursty.

The look-ahead and Tetris regions are each initially set to 500,000 bytes or 4 megabits (Mb), given the simulator configuration of 1 Gb/s per port. This is higher than the currently recommended router buffer size of 2.5 Mb, based on the switch size and speed being simulated [6]. For comparison, the performance of an unbuffered switch is shown in the same plot.

The simulator uses the standard Unix library *drand48()* function as its pseudorandom number generator. We run each simulation multiple times to achieve 95% confidence level with accuracy of 1% of the average. The number of runs, n, is determined using the following formula [7]:

$$n = \left(\frac{zs}{rx} \right)^2$$

where z is 1.96 from the normal probability distribution table, s and x are the standard deviation and the average from

preliminary runs, and *r* is the desired accuracy, which is 1% or 0.01. The numbers plotted here are the averages from those runs; the confidence intervals are omitted in the graphs because they are negligible. Each run lasts for 1 s of simulated packet processing time. To validate the parameters of our simulator, we ran a well-known input buffered switch architecture [8] for the same duration and checked the output of the simulator. Our simulator reported the same 58.6% throughput.

For a region size of 4 Mb and quasipoisson arrivals, the Tetris switch achieves close to 100% throughput for all the packet length distributions evaluated. For brevity, only the bimodal packet sizes are shown (Figure 5a). Packet drops do occur, but at a very low level. The result for pareto-r arrivals and bimodal packet sizes shows that the Tetris switch also provides noteworthy improvement over an unbuffered switch (Figure 5b). The result is similar to that of quasipoisson. Even though the traffic is bursty on the link, it is not bursty from an output port viewpoint. This is due to the uniform selection of output port for every packet in the burst in a pareto-r distribution.

In contrast, the result for pareto-n arrivals (also with bimodal packet sizes) shows that the Tetris switch does not achieve the desired near-100% throughput. It achieves only 83% throughput under 100% input load (Figure 5c). The throughput eventually goes to over 90% when the region sizes are increased to 64 Mb each. The decreased throughput (or, alternately, need for larger region sizes) is expected, however, because this traffic is bursty from an output port point of view.

Burstiness also affects the performance of an unbuffered switch. Figure 5c shows that pareto-n performs better on unbuffered switches than quasipoisson (Figure 5a) or pareto-r (Figure 5b). This is attributed to using a single output port for every packet in a burst. When an input port sends a burst to an output port, it does not send each packet to a different output port. In effect, this reduces the chance of having multiple input ports send to the same output port at the same time. Hence, the throughput of the unbuffered switch is improved for this kind of traffic.

Short-term variations in load do not have much effect on any of the throughputs indicated in Figure 5 because the Tetris region is far larger than the packets. Even with a uniform switching matrix, short-term variations of input load can cause the number of bytes destined to an output port to be greater than the number of bytes available in the Tetris region. This causes the Tetris switch to execute the spill steps and drop packets. The effect is more significant as the size of the regions decreases. Lowering the region sizes to 1 kilobit (kb) illustrates this effect (Figure 6).

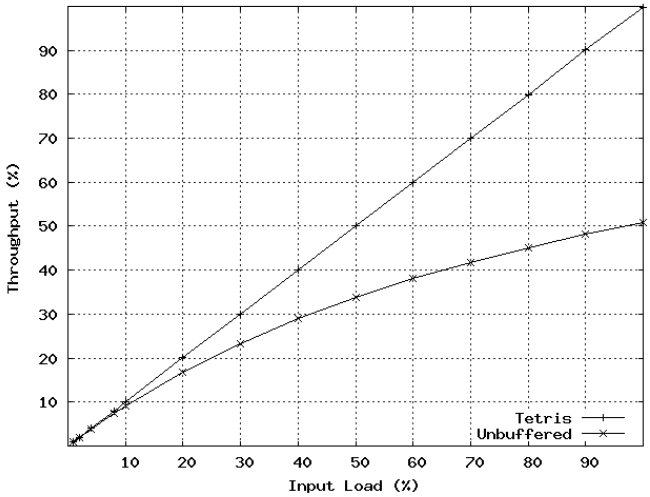
The Tetris switch throughput still is above 90% for quasipoisson arrivals even for a small delay for the packet length distributions considered (50 kb per region, *i.e.*, 6250 bytes or four 1500-byte packets). The performance level for bursty traffic is also lowered due to the prolonged variation of input load. For quasipoisson arrivals, the throughput drops significantly when the regions are smaller than 25 kb, or two 1500 byte packets (Figure 6a). A steep shift in the graph, or "cliff", is observed for both exponential and bimodal packet lengths. This is expected because there isn't any space to hold bigger packets, or smaller packets when a bigger packet occupies the Tetris region. The cliff shows a drop to the performance level of an unbuffered switch. Pareto-n arrivals, however, do not exhibit the same level of drop (Figure 6b).

An interesting small jump is observed for variable size packets when the region size is lowered from 10 kb to 5 kb or 2 kb. This is due to Step 6 in the algorithm. To maximize throughput when the region size is small, unshifted packets continue to propagate toward the mux (as per a conveyor queue, *i.e.,* the conveyors keep moving). When the mux is idle, the packet will be forwarded directly to the output port. When the mux is busy (*i.e.,* forwarding a different packet), the packet arriving at the mux will be dropped. The same operation is applied to shifted packets as well. Thus, when a large packet is being considered for shifting, smaller regions cause it to be unshiftable more often. The shifted packets are more often the smaller packets. The more unshifted packets hit the mux, the more they will be successfully forwarded. The shifted packets, which have smaller sizes, therefore, are more likely to be dropped. Thus, their throughput is slightly increased. When the regions are set to 1 kb, the mix of unshifted packets changes. There are more small packets that are unshiftable. This lessens the effect of unshifted packets monopolizing the mux.

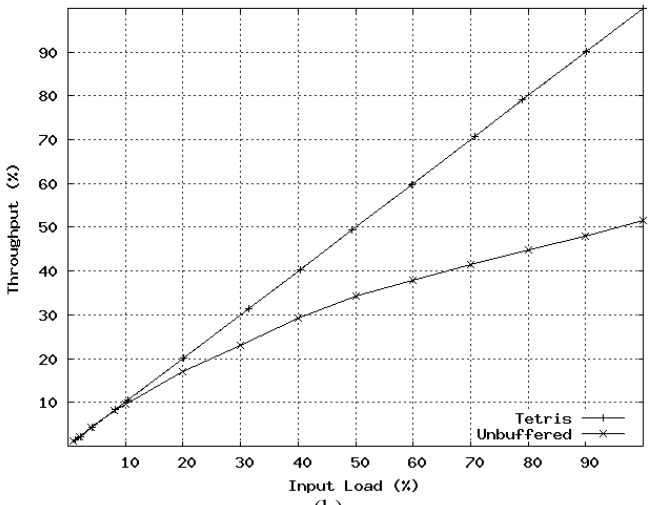
It is important to note that results in Figure 5 and Figure 6 are obtained by simulating the shifting of packets at arbitrary times. This is not realistic given the design of the VSC queue shown in Figure 3. However, placing the selector element at every smallest packet size interval should emulate shifting at arbitrary times. This was validated by simulating the discrete shifting points at every 320 bits (equivalent to one 40-byte minimal IPv4 packet) and 640 bits (two such packets) for equal region sizes of 50 kb.

The throughput drops from 92% to 88% to 85% for quasipoisson arrivals as the shifting point is changed (correspondingly) from continuous to 320 bits to 640 bits (Figure 7). The simulation output shows that the drop for pareto-n is also not severe, *i.e.*, from 68% to 66% to 65% for the same shifting points. Placing the selector element at larger intervals causes some voids in the VSC queues. Unlike some switches [9, 10], Tetris does not include void filling, thus the resulting throughput decreases.

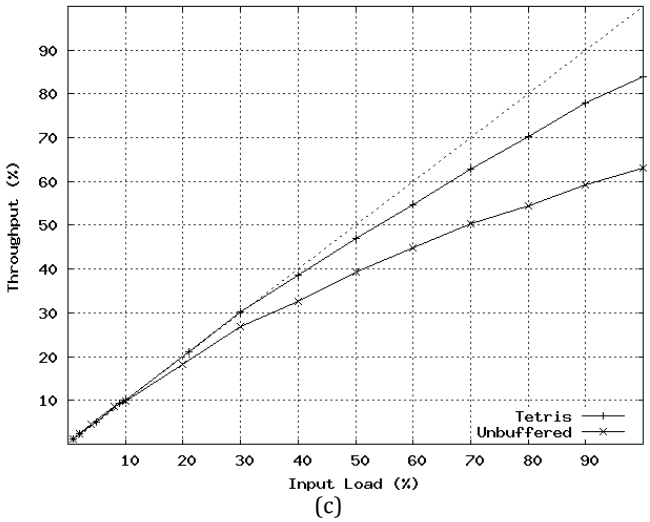
We also want to compare the performance of the Tetris switch to a conventional electronic switch, a VOQ switch with Parallel Iterative Matching (PIM) scheduling algorithm, modified so that it works on packets instead of fixed-size cells [11]. Figure 7 shows that the Tetris switch does perform similarly to the VOQ-PIM with 50 kb worth of buffering per virtual queue. The VOQ buffer size is selected to match the 50 kb Tetris region. To clarify the performance difference, the relative throughput of the Tetris switch is shown as a ratio to VOQ-PIM throughput in Figure 8. For bimodal packet sizes, the Tetris switch performs at 87 to 95% of the VOQ-PIM switch for quasipoisson arrivals and 93 to 98% for pareto-n arrivals.



(a)

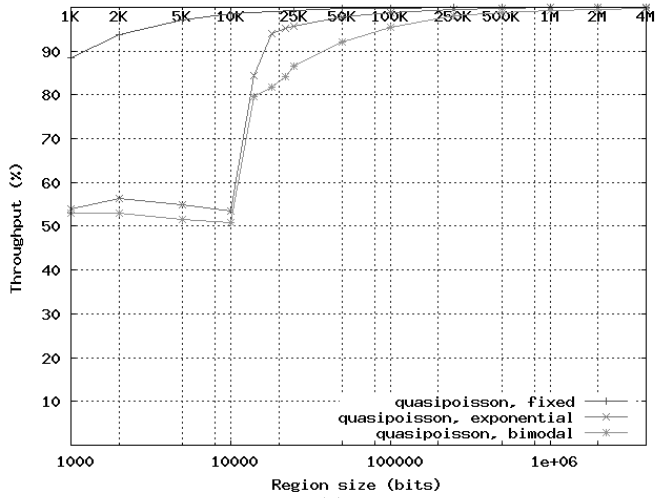


(b)

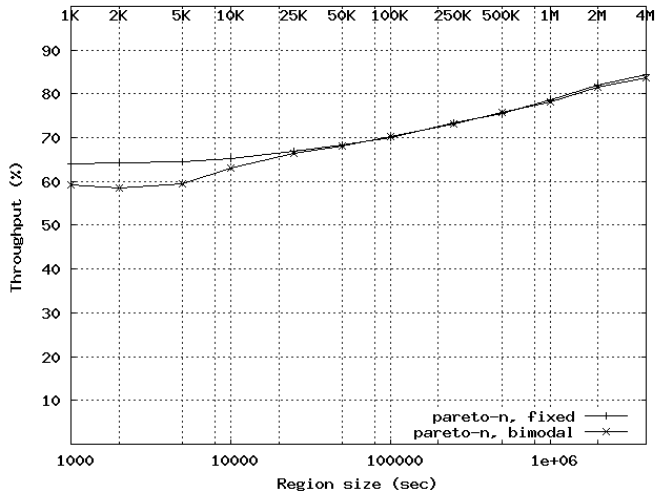


(c)

Figure 5: Throughput of a 32×32 Tetris switch for (a) quasipoisson, (b) pareto-r and (c) pareto-n arrivals. Look-ahead and Tetris region are each 4 Mb.



(a)



(b)

Figure 6: The throughput of a 32×32 Tetris switch as the look-ahead and Tetris region sizes get smaller at 100% input load. Both regions have equal size. The arrival processes are (a) quasipoisson and (b) pareto-n.

The Tetris switch does perform better than VOQ in terms of the average delay experienced by a packet especially at higher input loads (Figure 9). There is an interesting side effect in the delay experienced by a packet in a Tetris switch. For quasipoisson arrival and bimodal packet sizes, the delay drops when the input load increases from 0% to 10%. It then increases as the input load increases further. This is due to the shifting algorithm, which operates in batches. Step 1 in the algorithm collects packets in the look-ahead region until time *T*, which is the time when the first packet is at the start of the Tetris (or shifting) region. Step 3, the shifting operation, then starts working at time *T*. This step tries to shift all packets that are collected while the first packet is propagating through the look-ahead region. The first packet then experiences the whole delay incurred by the look-ahead region. Subsequent packets, however, do not incur the full amount because they are

starting to shift in the middle of the look-ahead region. This operation is a batch shifting operation. Thus, the more packets there are in the look-ahead region, the less is the delay. Hence, the average delay experienced decreases as load increases until the delay in the Tetris region becomes dominant. At that point, the average delay increases with the input load as expected.

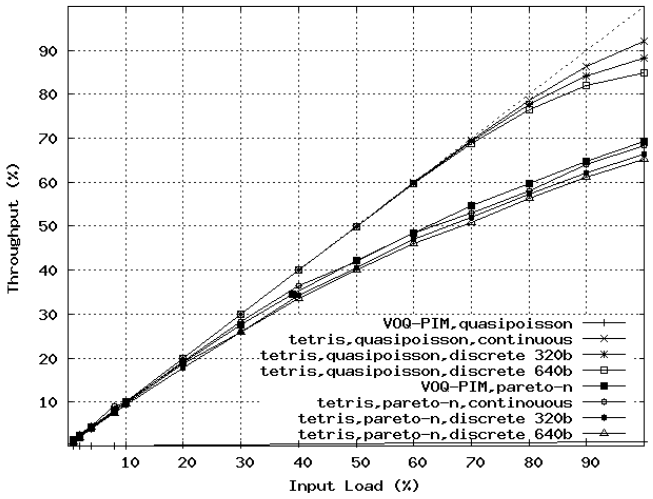
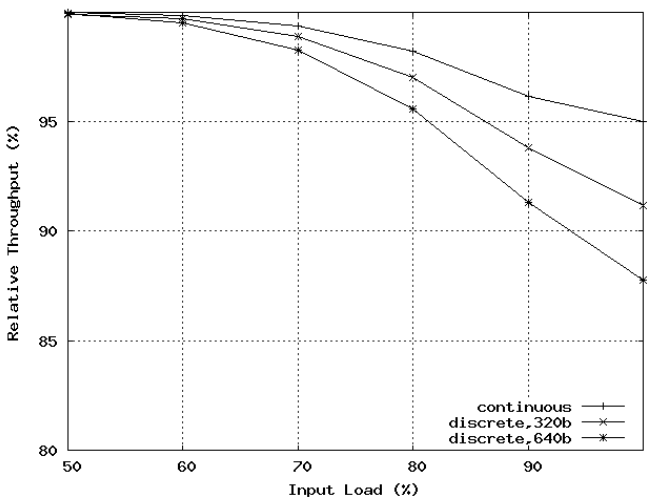


Figure 7: The throughput of a 32×32 Tetris switch with equal look-ahead and Tetris region sizes of 50 kb compared to VOQ-PIM. Tetris throughputs are for continuous and discrete shifting points (every 320 and 640 bits).

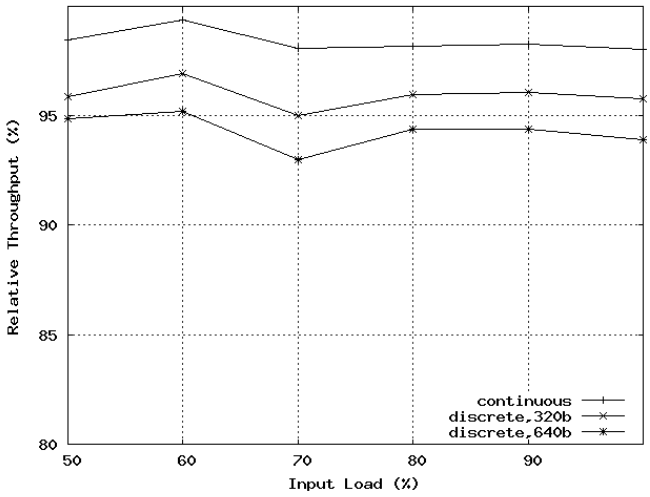
IV. TETRIS VARIANTS

The shifting concept used in the optical version of the Tetris switch can be applied to electronic switches as well. Instead of using slow light buffers as the delay component, the switch can use an analog delay line such as *Bucket Brigade Device* (BBD) [12]. Conventional electronic shift registers could also be used, but they use more transistors than BBDs. A recent development in non-volatile memory called *racetrack memory* supports the shifting concept, *i.e.*, *moving the data through a medium* [13]. An early example of this concept is the mercury delay line from the 1940s, in which data was represented as sound pulses moving through mercury confined to a metal tube [14]. The concept of moving a packet at variable speed through a medium, whether it is optical, electronic, or even mechanical, is the main working assumption in the Tetris switch. Thus, it is expected that the design is easily applicable to the electronic domain as well. Our future work includes a more detailed comparison of the Tetris switch with current electronic switches. We currently only compare it to VOQ with the PIM scheduling algorithm.

The combination of VSC queues with a mux and a scheduling mechanism is the core of the Tetris switch, and can be useful separately. We call this a Tetris mux, and it can be beneficial in so-called 'access' networks, *i.e.*, networks used to aggregate traffic from a large number of lower-speed networks to a higher-speed core link.



(a)



(b)

Figure 8: Throughput of a Tetris switch in Figure 7 shown as relative to the VOQ throughput using PIM scheduling algorithm: (a) quasipoisson and (b) pareto-n.

V. PRIOR WORK

Electronic switches have been studied extensively. The well-know relevant results are:

- Head-of-the-line (HOL) blocking limits the throughput of electronic switches to 59% [8].
- Virtual output queuing (VOQ) eliminates HOL blocking and achieves near-100% throughput [15].

These two results are for fixed packet sizes and a uniform switching matrix. They inspire and inform the basic design of the Tetris switch architecture.

In an earlier work we analyzed and simulated an un-buffered switch for variable length packets, whose achievable throughput is only 50% [16]. This forms the baseline of the throughput gap between the optical (unbuffered) and the electronic (buffered) switches. We also simulated a switch that

is capable of looking ahead beyond HOL packets, called the *precognition* switch. This switch selects packets that maximize the number of bytes transferred. Disappointingly, this switch improves performance by only a few percent [16]. This highlights that the improvement achieved by the Tetris switch is the result of the shifting operation, not just the ability to look ahead in the arriving packet stream and optimally select drops as done in the precognition switch.

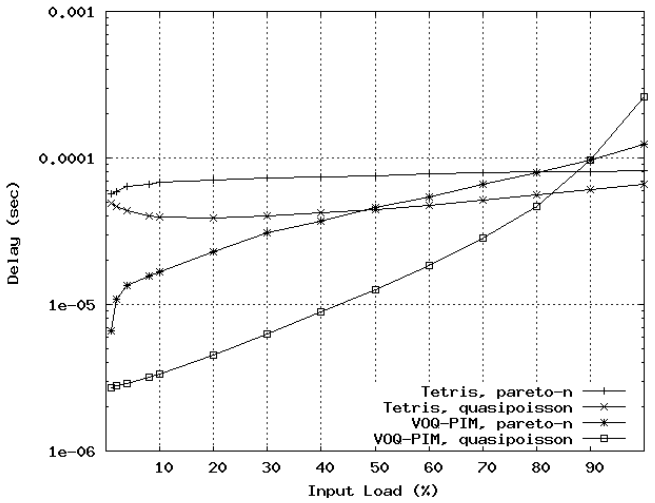


Figure 9: Average delay of a Tetris switch compared to VOQ PIM algorithm for quasipoisson and pareto-n arrivals. The Tetris switch is using look-ahead and Tetris region sizes of 50 kb each. VOQ switch buffer is 50 kb per virtual queue.

This paper focuses on switch designs capable of optical implementations. There are multiple approaches to contention resolution in optical packet switches. The most prevalent approach is a buffering scheme called switched delay lines (SDL) [17, 18]. SDLs use fiber delay lines (FDLs) as the delay component in the buffer. However, FDLs are not practical because of the length of fiber required to implement even a very small delay. For example, a 100 ns delay requires 20 meters of fiber [19]. Recent developments in the area of slow light optical buffering are promising because they require only 50 cm^2 for 400 kb buffering [20]. It is expected that a capacity of 1 megabit (Mb) can be easily supported. The optical version of the Tetris switch would be expected to use SDLs and slow light buffers as its delay elements.

The Staggering switch [21] is an example of a switch that uses FDLs for buffering. A 32×32 staggering switch achieves 97% throughput under 100% input load. However, it assumes that the packets are fixed size and packet arrivals are synchronized. We want to deduce its performance for variable length packets by looking at the results for bursty cell traffic. Unfortunately, results are provided for only a 4×4 switch. Although this shows 97% throughput, a 32×32 switch is expected to have much lower performance. Our simulation shows that increasing switch size lowers throughput, in general. Others have reported the same observation [8, 15]. The Tetris switch performance evaluation herein uses variable length packets (bursty and non-bursty) and asynchronous packet arrivals. These two operating conditions increase the probability of collision beyond what was considered for the staggering switch.

Slow light buffers are also being considered in a combined input and output buffered optical switch [22]. Instead of using the buffer as a bit-delay element, whole packets are buffered. The simulation results of this switch show good performance even with bursty traffic. However, the switch uses multiple wavelengths and was evaluated for only a 4×4 configuration. An optical Tetris, on the other hand, could use fixed-delay slow light buffers and does not rely on wavelength division multiplexing.

The switch with large optical buffer (SLOB) architecture extends the idea of the staggering switch to create a significantly larger buffer [23]. FDLs of multiple lengths are used to form its delay, from zero delay to a delay of m-1 time slots. This first stage of configurable delay cascades to a subsequent stage where the delay starts from m to $(m\text{-}1)m$ time slots. The following stage starts from m^2 to $(m\text{-}1)m^2$. These cascaded stages are used as the output queues for a cell-based switch. Tetris uses VSC queues as delay, and can use the SLOB approach when a longer delay is necessary. The notable difference is that in SLOB, a packet can go through a delayed path then a non-delayed path and come back to the delayed path. In the Tetris switch, once the packet enters a non-delayed path, it stays in that path until it is muxed (or dropped), and as a result its packet switching latency is lower than in SLOB.

A paper by Shiramizu *et al.* [24] describes a buffering architecture in which each input has a dedicated buffer in the output. In a sense, this is an optical VOQ. Each buffer is equipped with multiple FDLs. The number of parallel FDLs is the capacity of the buffer in terms of the number of storable packets. Different from a VOQ, each output has a FIFO buffer manager that arbitrates the packets in the optical domain. When there is a packet being transmitted to an output port, other packets that reach the end of the FIFO are recirculated back. Thus, a packet can potentially be recycled and delayed for an arbitrary amount of time.

Our approach is similar to Shiramizu's because the Tetris switch also uses the concept of VOQs. However, the Tetris switch senses potential collision while the packets are still in flight and time-shifts the packets to avoid collision, instead of relying on parallel FDLs and recirculation. Tetris switch VSC scheduling takes into account the time of arrival of packets, which thus requires an electronic controller.

Asynchronous optical packet switches with variable length and bursty traffic also have been studied [9, 10]. They use the same traffic model shown herein (pareto on/off) with similar parameters except for the mean burst size. Their approaches rely on multiple wavelengths for contention resolution, which yields good performance. The studies also explore a utilization problem created by the discrete nature of FDLs. This is solved by a technique called *void filling*, which allows the next packet on the input port that is destined to the same output port to be scheduled behind the previous packet even though there is another packet on a different input port that arrives earlier than that next packet. As noted earlier, the Tetris switch does not

currently rely on this mechanism, and operates on a single wavelength.

A recent study by Appenzeller *et al.* [6] shows that the buffering requirement for an IP router is $RTT / \sqrt{n}$ where *RTT* is the average round trip time of a flow passing the link and *n* is the number of flows carried by the link. In our scenario the typical value of RTT is 250 ms and the number of flows is 10,000. So, the buffering requirement can be as low as 2.5 ms. On a 1 Gb/s link, this translates to 2.5 Mb. We thus use this number as a starting point for our performance evaluation and then also consider lower values that would be more practical for slow light buffers.

Optical burst switching (OBS) is another relevant approach to optical packet switching [4]. Like OBS, the Tetris switch also relies on a time offset between the arrival of the packets and the arrival of the control signals in asimilar manner to OBS. OBS uses this offset to avoid buffering (by deflecting the bursts) and to reserve switch connections along the path of a burst through a switch. The Tetris switch uses the time offset to determine which packets to shift forward. Another difference is that OBS focuses on the performance of an optical network with multiple wavelengths, whereas the optical Tetris switch is intended to support packets on just a single wavelength, *i.e.*, the Tetris switch enables true optical packet switching, and does not rely on wavelength switching for its performance. Enhancing OBS with FDLs is a viable alternative to resolving output port contention solely with wavelengths [25, 26]. The latter study employs tunable FDLs to accommodate diverted bursts, possibly for several recirculations. The Tetris switch avoids recirculation and does not have to calculate complicated schedules that allow packets to emerge from the FDL at just the right time.

VI. Concluding Remarks

The Tetris switch provides a promising approach to optical packet switching, and can also support electronic switching, using a new mechanism for packet multiplexing using variable speed conveyor queues. It achieves high performance for uniformly distributed traffic with a simple scheduling algorithm. This performance can be maintained with as little as 50 kb of delay on each region (a total of 100 kb of delay) for non-bursty traffic. With 50 kb region sizes, the use of slow light buffers are feasible For both bursty and non-bursty traffic, the Tetris switch enjoys most of the benefit of VOQ without the need for random-access memory, which has proven extremely difficult to implement with optical technology.

References

[1] Coffman, E.G., Jr., E. Gelenbe, and E.N. Gilbert. Analysis of a conveyor queue in a flexible manufacturing system. In *ACM SIGMETRICS Perform. Eval. Rev.*, 14(1): 204-223, 1986.

[2] Papadimitriou, G.I., C. Papazoglou, and A.S. Pomportsis. Optical switching: switch fabrics, techniques, and architectures. In *J. of Lightwave Technology*, 21(2): 384-405, 2003.

[3] Crovella, M.E., M.S. Taqqu, and A. Bestavros. *Heavy-tailed Probability Distributions in the World Wide Web*. Chapman and Hall, 1998.

[4] Qiao, C. and M. Yoo. Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet. *J. of High Speed Networks*, 8(1): 69-84, 1999.

[5] Sinha, R., C. Papadopoulos, and J. Heidemann. Internet Packet Size Distributions: Some Observations. USC/ISI Tech Report 643, 2007.

[6] Appenzeller, G., I. Keslassy, and N. McKeown. Sizing Router Buffers. In *ACM SIGCOMM 2004*. Portland, Oregon, pp. 281-292.

[7] Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*: John Wiley & Sons, 1991.

[8] Karol, M.J., M.G. Hluchyj, and S.P. Morgan. Input Versus Output Queueing on a Space-Division Packet Switch. *IEEE Trans. on Communications*, COM-35(12): 1347-1356, 1987.

[9] Hunter, D.K. and I. Andonovic. Approaches to Optical Internet Packet Switching, *IEEE Communications Magazine*. 38(9): 116-122, 2000.

[10] Tancevski, L., *et al*. A new scheduling algorithm for asynchronous variable length IP traffic incorporating void filling. In *Optical Fiber Communication Conference and International Conference on Integrated Optics and Optical Fiber Communication OFC/IOOC*. 3:180-182, 1999.

[11] Ge, N., M. Hamdi, and K.B. Letaief. Efficient scheduling of variable-length IP packets on high-speed switches. In *GLOBECOM*, 2: 1407-1411, 1999.

[12] Analog Delay Devices. Available from: http://mysite.du.edu/~etuttle/electron/elect39.htm.

[13] Parkin, S.S.P. Racetrack Memory: The Future Third Dimension of Data Storage. *Scientific American*. 2009.

[14] Eckert, J.P., Jr. A survey of digital computer memory systems. *IEEE Annals of the History of Computing*, 20(4): 15-28, 1998.

[15] McKeown, N.W. *Scheduling Algorithms for Input-Queued Cell Switches*. Ph.D. Dissertation,University of California at Berkeley, 1995.

[16] Suryaputra, S., J. Touch, and J. Bannister. The Case of a Precognition Optical Packet Switch, In *IEEE INFOCOM Workshop on High Speed Networks*, pp. 1-6, 2009.

[17] Chlamtac, I. and A. Fumagalli. An Optical Switch Architecture for Manhattan Networks. *IEEE J. on Selected Areas in Communications*, 11(4): 550-559, 1993.

[18] Yao, S., B. Mukherjee, and S. Dixit. Advances in Photonic Packet Switching: An Overview. *IEEE Communications Magazine*. 2000.

[19] Tucker, R.S., P.-C. Ku, and C.J. Chang-Hasnain. Slow-Light Optical Buffers: Capabilities and Fundamental Limitations. *J. of Lightwave Technology*, 22(12): 4046-4066, 2005.

[20] Tucker, R.S. The Role of Optics and Electronics in High-Capacity Routers. *J. of Lightwave Technology*, 24(12): 4655-4672, 2006.
[21] Haas, Z. The'Staggering Switch': An Electronically Controlled Optical Packet Switch. *J. of Lightwave Technology*, 11(5/6): 925-926, 1993.
[22] Haijun, Y. and S.J.B. Yoo. Combined input and output all-optical variable buffered switch architecture for future optical routers. *IEEE Photonics Technology Letters*, 17(6): 1292-1294, 2005.
[23] Hunter, D.K., *et al*. SLOB: a switch with large optical buffers for packet switching. *J. of Lightwave Technology*, 16(10): 1725-1736, 1998.
[24] Shiramizu, Y., J. Oda, and N. Goto. All-optical autonomous first-in--first-out buffer managed with carrier sensing of output packets. *Optical Engineering*, 47(8): 085006-8, 2008.
[25] Gauger, G. Dimensioning FDL buffers for optical burst switching nodes. In *Optical Network Design and Modeling*. pp. 117-132, 2002.
[26] Merchant, K., *et al*. Analysis of an optical burst switching router with tunable multiwavelength recirculating buffers. *J. of Lightwave Technology*. 23(10): 3302-3312, 2005.