Design of an Optical Packet Switch for Real-Time Applications

Jaemyoun Lee*, Juyoung Park*, Kyung-Joon Park†, and Kyungtae Kang*§
*Department of Computer Science and Engineering, Hanyang University, Korea
Email: {poken01, jypark, ktkang}@hanyang.ac.kr
†Department of Information and Communication Engineering, DGIST, Korea
Email: kjp@dgist.ac.kr

***Abstract*—Network switches are typically designed for best-effort Internet traffic. Most of existing studies have been focused on improving throughput and delay performance in an average sense rather than providing guaranteed delay bound that is critical for real-time applications. It has not been fully investigated how to design an efficient packet switching algorithm for real-time applications. In this paper, we propose a design framework for a real-time optical switch that is intended for use as an optical switch fabric. Our contributions are two folds: First, by introducing a clearance-time optimal switching together with clock-based scheduling, our switching design guarantees any feasible real-time traffic to be switched in two-clock periods. Second, we investigate key implementation issues of an optical packet switch such as packet size and buffering for real-time applications, and take account of these issues in design and performance evaluation of a switching algorithm. Our numerical study shows that the proposed switching algorithm provides a larger schedulability region with significantly reduced delay compared to the well-known iSLIP scheme.**

I. Introduction

In order to realize real-time networks in a proper manner, it is of critical importance how to design an efficient real-time switch. Though there have been extensive studies on design of network switches, most of them have typically focused on improving throughput and delay performance in an *average sense* rather than providing guaranteed delay bound that is critical for real-time applications [1], [2], [3].

In this paper, we study the design of an optical packet switch for real-time applications. We primarily focus on the problem of how to design an efficient real-time optical switch that can guarantee a bounded delay. In addition, we investigate key design issues of optical packet switches as a possible component for real-time embedded systems because optical switches have several benefits over electronic ones such as light weight and immunity to electrical noise. Furthermore, optical network-on-chip technology is actively investigated, which makes it possible for an optical switch to be used for real-time embedded systems. In particular, our contributions are as follows:

- We design a real-time optical switch with bounded delay under any feasible traffic. By introducing a clearance-time optimal policy together with clock-driven scheduling, *any feasible traffic* is *guaranteed* to be switched in *two-clock periods.*
- We investigate key implementation issues of an optical packet switch such as packet size and buffering, and properly take into account these issues in design and performance evaluation of a switching algorithm for real-time applications.

§Corresponding author.

Design of real-time switches have been recently studied in [4], [5]. In particular, an efficient switch design for real-time applications has been proposed in [5], where deterministic and periodic real-time traffic has been assumed in the design of a switching algorithm. Our work significantly differs from these studies in the sense that we introduce *no assumption* on traffic and design a switch algorithm with *delay guarantee for any feasible traffic*. In the meanwhile, it should be noted that our main focus is *not* on development of the switching theory itself. We are interested in exploiting the well-developed switching theory in order to design an efficient *real-time optical* packet switch.

The remainder of the paper is organized as follows: In Section II, we look into the design factors of a real-time optical packet switch. Then, in Section III, based on clock-driven scheduling, we design a real-time switching algorithm, which can guarantee the bounded delay for any feasible traffic. We perform a numerical study on switch design for real-time multimedia applications in an avionics system in Section IV. In this section, we take into account key characteristics of real-time multimedia traffic. Then, we verify the schedulability and delay performance of the proposed switch architecture. Finally, our conclusion with future research avenues is given in Section V.

II. Design of Optical Switches

In this section, we provide a brief background on optical switches. Then, we look into the key implementation issues in design of an optical switch [6], [7], [8]. In particular, we pay attention to issues in an optical *packet* switch [9].

A. Optical Packet Switches

The significantly increasing demand for optical network capacity has fueled the development of long-haul optical network systems, which employ wavelength-division multiplexing (WDM) to achieve tremendous capacities. Such systems can support tens to hundreds of wavelengths per fiber, with each wavelength modulated at 10 Gb/s or more [10].

Pure WDM only provides granularity at the level of one wavelength, which wastes capacity when compared to the use of fractions of a wavelength. When fractional wavelengths are combined with optical packet switching, packet streams can be multiplexed together statistically, making more efficient use of capacity and providing increased flexibility over pure WDM [6]. Wavelength is also used as a dimension inside the optical packet switch, in order to allow the optical buffers to be used efficiently and the switch throughput to be maximized.

Packet switches analyze the information contained in the packet headers and thus determine where to forward the packets. Optical packet-switching technologies enable the allocation of WDM channels on demand within microseconds. An optical packet switch is also able to accept increases in the transmission bit-rate, allowing the capacity of the transmission to be increased with only a minor impact on the switching nodes [11]. In addition, optical packet switching offers high speed, data rate/format transparency and configurability, which are important characteristics for making networks as future-proof as possible [12].

B. Design Issues in Optical Packet Switching

There are two design issues in optical packet switching: the lack of an optical capability for bit-level processing and the inefficiency of storing information in the optical domain.

Bit-level processing is required to read and interpret the packet headers. Packets that arrive at a packet-switching node are directed to the input interface of the switch, which extracts the routing information from the headers. This is then used to control the switching matrix, which actually performs the switching and buffering. The control of these functions is electronic because optical logic is currently too primitive to permit optical control. After switching, packets are directed to the output interface, where their headers are rewritten. The operating speed of the control electronics places an upper limit on the switch throughput. Although techniques exist to detect and recognize packet headers at gigabit-per-second speed [13], [14], it is still difficult to design electronic header to operate at sufficiently high speed to switch packets on the fly at every node [12]. For this reason, it is demanding that the size of a packet should be sufficiently large to offset the control overhead.

The issue of optical data storage affects the way in which packet contentions are resolved in an optical network. Contentions occur in the network switches when two or more packets require the same resource. A simple solution to this problem is to buffer the contending packets. Currently, fiber delay lines (FDLs) [12] are the only way to buffer a packet optically. Contending packets are sent to travel over an additional length of fiber and are thus delayed for a specific time.

Optical buffers that rely on FDLs have several disadvantages. First, FDLs are bulky and expensive, and a packet cannot be stored indefinitely on an FDL. Also, once a packet has entered an FDL, it cannot in general be retrieved before it emerges at the other end, after a fixed length of time. In other words, FDLs do not allow random access. In addition to these structural limitations, optical signals that are buffered using FDLs experience lose quality as they travel through the fiber. The length of each FDL, which is dictated by the duration of a packet, and the number of FDLs are therefore critical design parameters. Hence, as long as the contention can be resolved, it is obviously desirable to bound the number of buffered packets.

In summary, there are two key design issues in an optical packet switch. First, the size of a packet should be large enough to make the control overhead remain in a reasonable level. At the same time, the packet size should be chosen by considering the characteristic of traffic. Second, the buffering should be bounded as long as the contention can be handled, which can be accomplished if the switching delay can be bounded. In addition, buffering can be further reduced if the network operates in a synchronous manner. In the following sections, we will design a switching algorithm that can resolve these issues.

III. Design of a Real-Time Switching Algorithm

In this section, we design a real-time switching algorithm that can guarantee a bounded delay with any feasible traffic. We consider the widely-adopted approach of a virtual-output queue architecture, and introduce the concept of clock-driven scheduling as a virtual machine task. It should be noted that periodic traffic is assumed in [5] while we introduce *no assumption* on traffic characteristics except feasibility.

A. Crossbar Switch

We adopt a widely used $N \times N$ crossbar hardware fabric. At every time slot, each input port picks up a packet from one of its queues and sends it to its destined output port over a data bus. The switch fabric can turn on or turn off each crosspoints during runtime according to the switching logic. These crossbar connections can not be arbitrarily established, but are limited to the set of $N!$ permutation matrices where a permutation matrix is defined as $N \times N$ matrix composed of 0 and 1, with only one 1 in each row and column. To facilitate the switching logic, we assume that packet size of traffic is fixed, and the time for each packet transmission is denoted as one time slot.

Let queue (i, j) denote the input queue that holds packets from input i to output j. In addition, let $A_{ij}(t)$ and $Q_{ij}(t)$ denote the number of packets arriving to queue (i, j) and the current number of queued packets in queue (i, j) at time slot t, respectively. Then, the switching decision variable $S_{ij}(t)$ at time slot t can be defined as follows:

$$S_{ij}(t) = \begin{cases} 1, & \text{if crosspoint } (i,j) \text{ is turned on at } t; \\ 0, & \text{otherwise.} \end{cases}$$

Note that the switching matrix $[S_{ij}]$ corresponds to one of $N!$ permutation matrices because of the crossbar constraint described above. Each queue (i, j) of the switch is dominated by the following dynamic equation:

$$Q_{ij}(t+1) = \max\{Q_{ij}(t) - S_{ij}(t), 0\} + A_{ij}(t). \tag{1}$$

The maximum operation in (1) is introduced to cover the case when $Q_{ij}(t) = 0$ and $S_{ij} = 1$. Hence, qualitatively speaking, the design of a real-time switching algorithm corresponds to how to select the switching matrix $[S_{ij}]$ every time slot in order to satisfy the timing constraint.

Now, we look into the stability region of a crossbar switch. First, assume that input traffic is rate ergodic. Then, we can define the input rate of queue (i, j) as follows:

$$\lambda_{ij} := \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t} A_{ij}(\tau).$$

Then, it is well known that the stability region of the switch is given as the set of rate matrices that satisfy the following $2N$ inequalities:

$$\sum_{j=1}^{N} \lambda_{ij} \leq 1, i = 1, \ldots, N, \tag{2}$$

and

$$\sum_{i=1}^{N} \lambda_{ij} \leq 1, j = 1, \ldots, N. \tag{3}$$

This stability region implies that every queue in the switch algorithm can remain finite with probability one as long as the input rates are in the region. If these inequalities do not hold for any i or j, the corresponding queue will blow up with probability one. It should be noted that it is not a simple task to develop an efficient switching algorithm that can serve any traffic in the stability region. In fact, switching algorithms that can stabilize any traffic in the stability region are called *throughput-optimal*, which is an asymptotic property.

Now, we introduce the graph-theoretic approach for design of switching algorithms. It is well known that the problem of packet switching in crossbar fabrics can be transformed into that of finding matchings in bipartite graphs as follows: At each time slot, let G denote the bipartite graph with input ports and output ports on each side, respectively. Each port in the switch is considered as a node in the bipartite graph G. Hence, hereafter, *port* and *node* will be used interchangeably. Then, there exists an edge between input port i and output port j in G if there are packets waiting at input port i destined to output port j. A switching algorithm finds a matching M in G to determine which packets in input ports will be switched at each time slot. A scheduling policy is then to determine the matching M at each time slot, possibly based on the current state of the queues. In fact, the problem of finding a matching M corresponds to that of selecting the switching matrix $[S_{ij}]$ in (1).

For each input port i, let Q_i denote the total number of packets at port i. In a similar manner, let Q_j denote the total number of packets (at all the inputs) destined to output port j. Hence, if we let Q_{ij} denote the number of packets at input port i destined to output port j, we have $Q_i = \sum_{j=1}^{N} Q_{ij}$ and $Q_j = \sum_{i=1}^{N} Q_{ij}$. The queue Q_i and Q_j will be the *weight* of the corresponding node in the bipartite graph G.

B. Clock-Driven Scheduling as a Virtual Machine Task

A widely deployed approach for real-time virtual machine task (VM-task) is clock-driven scheduling, where a VM-task (L, C) represents that a real-time task is served C time units during each clock period of L time slots.

Let f_k^{ij} denote the k-th real-time flow from input port I_i to output port O_j, where $k = 1, 2, \ldots, K_{ij}$, and K_{ij} is the number of flows from I_i to O_j. In addition, f_k^{ij} is associated with a VM-task (L, C_{ijk}), i.e., flow f_k^{ij} has C_{ijk} packets to be served. Consequently, under clock-driven scheduling, the delay of f_k^{ij} will be bounded as long as the switching algorithm can guarantee the worst-case delay bound to forward all the C_{ijk} packets of f_k^{ij}.

Let C_{ij} denote the total number of packets to be forwarded from I_i to O_j. Then, we have $C_{ij} = \sum_{k=1}^{K_{ij}} C_{ijk}$. In addition to the stability condition in (2) and (3), for the VM-task sets $\{(L, C_{ijk})\}$, $i = 1, \ldots, N$, $j = 1, \ldots, N$, $k = 1, \ldots, K_{ij}$, we introduce the feasibility of real-time traffic for each clock time of L time slots as follows:

$$\sum_{j=1}^{N} C_{ij} \leq L, i = 1, 2, \ldots, L. \tag{4}$$

$$\sum_{i=1}^{N} C_{ij} \leq L, j = 1, 2, \ldots, L. \tag{5}$$

Those traffic that do not meet (4) and (5) are unschedulable, and we primarily focus on feasible traffic. However, as a remark, we will discuss later in the section how to extend our framework for those traffic that are infeasible in a clock period, but satisfies the stability condition in (2) and (3).

In summary, our proposed real-time switch serves each traffic flow as a VM-task, and each VM-task is served with a slot-by-slot switching algorithm that minimizes the clearance time for any feasible traffic. It should be noted that the presented clock-driven scheduling naturally makes the network operate in a synchronous manner, which is critical for an optical switch to reduce the buffering as explained in the previous section. We will explain the details of the proposed switching algorithm in the next section.

C. Clearance-Time Optimal Switching Policies

In order to design a switching algorithm with guarantee delay, we first introduce clearance-time optimal policies. In this setting, every queue in the system has initial packets of $Q_{ij}(0)$, and has no further arrivals, which is called *one-shot traffic*. Under this traffic condition, the clearance time is the time to serve every packets in the system. Switching policies that minimize the clearance time is called the clearance-time optimal policies.

As we have already explained in the previous section, because of the crossbar constraints, at most one packet can be switched at any port. Consequently, the clearance time, denoted by T_{clear}, is lower bounded by the maximum of the number of packets at a port as follows:

$$T_{clear} \geq \max\left(\max_i \sum_{j=1}^{N} Q_{ij}(0), \max_j \sum_{i=1}^{N} Q_{ij}(0)\right). \quad (6)$$

In fact, it has been known that this bound is tight, i.e., the minimum clearance time, denoted by T^*_{clear}, is equal to the right hand side of (6).

Now the question is how to design a switching algorithm that can achieve the minimum clearance time T^*_{clear}. To this end, we introduce a *critical-port policy* as follows: Given a bipartite graph G, node i is called *critical* if its weight is no smaller than any other nodes. Then, a matching M is a critical-port matching if it matches every critical port. Consequently, a scheduling policy is a critical-port one if it generates a critical matching in every time slot. It has been shown that a critical-port policy is clearance-time optimal as follows:

Proposition 1: A switching policy is clearance-time optimal if and only if it is a critical-port policy.

Proof: Here, for completeness, we provide a proof by closely following that of Proposition 1 in [3]. For any clearance-time optimal policy, at any time slot $s < T^*_{clear}$, every port i has $Q_i(s) \leq T^*_{clear} - s$. Otherwise, due to the crossbar constraint, the corresponding port cannot be cleared by T^*_{clear}. In a similar manner, it is clear that any ports with initial queue $Q_i(0) = T^*_{clear}$ will have $Q_i(s) = T^*_{clear} - s$. Consequently, it can be concluded that the weight (or queue) of the critical ports at $t = s$ is $T^*_{clear} - s$. If any of these critical ports are not served in time slot s, it is clear that the respective port cannot be drained by T^*_{clear}. Hence, every clearance-time optimal policy is a critical-port policy. Now, suppose we have a critical-port policy. Then, since any ports with the largest queues at any time slot are critical ports, those ports will be served and their queue will decrease by one. Hence, it is clear that critical-port policy corresponds to a clearance-time optimal policy. ∎

Remark By adopting a critical-port policy, we can guarantee that the clearance time of one-shot traffic can be minimized. In the next section, we present a design framework that can guarantee bounded delay for any feasible traffic.

D. Real-Time Switch with Clock-Driven Scheduling

In order to provide bounded delay for any feasible traffic, we incorporate the clock-driven scheduling with the clearance-time optimal scheduling in the following manner: The traffic arrived during one-clock period is buffered and served in the next clock period so that the one-shot traffic assumption of clearance-time optimal scheduling can be preserved. With this approach, by virtue of the clearance-time optimal property, any feasible traffic satisfying (4) and (5) is guaranteed to be served in two-clock periods. In other words, at the expense of an additional delay of one-clock period, our approach can ensure a deterministic delay bound of $2L$ for any feasible traffic.

Algorithm 1 Clock-based switching algorithm

```
1: for each clock period do
2:     Switch packets arrived in the previous clock by Algorithm 2
3: end for
```

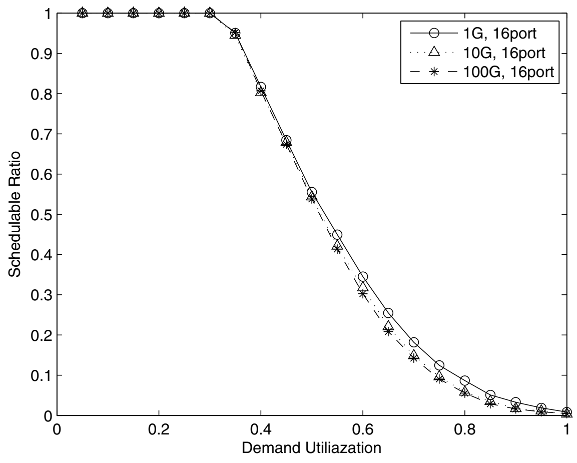
Algorithm 2 Lazy Heaviest Port First (LHPF) algorithm

```
1: INPUT: Any initial matching $M_0$
2: OUTPUT: LHPF matching $M^*$
3: // Initialization
4: $l \leftarrow 1$
5: // Iteration
6: loop
7:     if $M_{l-1}$ matches all nodes then
8:         $M^* \leftarrow M_{l-1}$
9:         BREAK
10:    end if
11:    Pick any highest unmatched node $i$ in $M_{l-1}$
12:    Find an augmented or absorbing path $P$ from $i$
13:    if $P$ exists then
14:        $M^* \leftarrow M_{l-1} \oplus P$
15:        $l \leftarrow l + 1$
16:    else
17:        $M^* \leftarrow M_{l-1}$
18:        BREAK
19:    end if
20: end loop
```

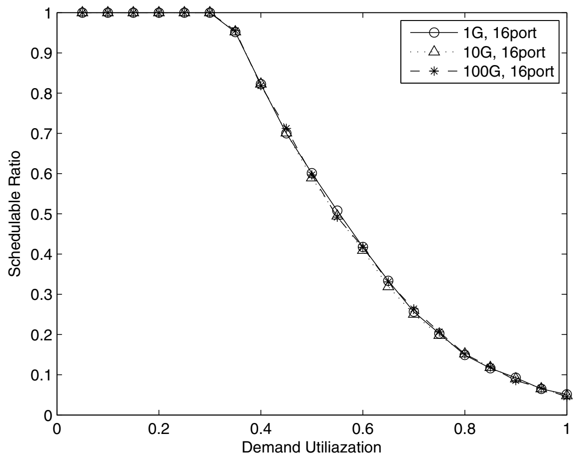
Among many possible realization of critical-port policies, we adopt the Lazy Heaviest Port First (LHPF) matching [3], which is defined as follows: First, threshold th of a matching M is defined as the lowest-possible positive integer such that the matching M matches all the ports with weight larger than or equal to th. For example, a perfect matching that switches every port having packets in the queue has $th = 1$. A matching is called an LHPF matching if it has the lowest threshold among all possible matchings. An algorithm that implements LHPF matching is given in Algorithm 2.

One advantage of LHPF is that it is throughput optimal [3]. The throughput optimality guarantees that every queue of the switch will remain finite with any traffic that satisfies the stability condition of (2) and (3). Consequently, even when the traffic does not satisfy the feasibility condition of (4) and (5), every queue can remain stabilized as long as traffic lies in the region of (2) and (3). Hence, the throughput optimality of LHPF further extends the schedulability region of the proposed switching algorithm. Our numerical study will further elaborate this issue in the next section.

We explain several graph-theoretic notions in Algorithm 2. For a given bipartite graph, the length of a path is defined as the number of edges the graph contains. For any given matching M and any node i not matched by M, an augmenting path from node i is defined as any odd-length path P whose every alternate edge is in M, has node i as one of its end points and an unmatched node as the other. An absorbing path from node i is defined as any even-length path P whose every alternate edge is in M, has node i as one end point, and the



(a) Schedulability region of iSLIP



(b) Schedulability region of the proposed shceme

Fig. 1. Schedulability region of iSLIP and the proposed scheme when the number of input ports is 16, and the port capacity is 1, 10, and 100 Gb/s.

other end point has weight larger than that of node i. Finally, for any given matching M and path P, we have $M \oplus P =: M - (M \cap P) + (M^c \cap P)$.

IV. Performance Evaluation

A. Comparison of the Schedulability Region

First, we look into the schedulability region of iLSIP and the proposed scheme in Fig. 1. In our simulation, the clock period is fixed to 1 ms. Consequently, any feasible traffic satisfying (4) and (5) is guaranteed to be switched in 2 ms, which is much smaller than the typical allowable end-to-end delay of 100 ms for multimedia video traffic. Each point in Fig. 1 is an average of 1000 simulation runs for a given switch setting. The per-port capacity of the switch is 10 and 100 Gb/s, and the number of input ports N is given as 8 and 16, respectively. The default value for the packet size is set to 10 Kbits. Note that we further evaluate the effect of the packet size on the switching performance (including switching overhead) in Section IV-C.

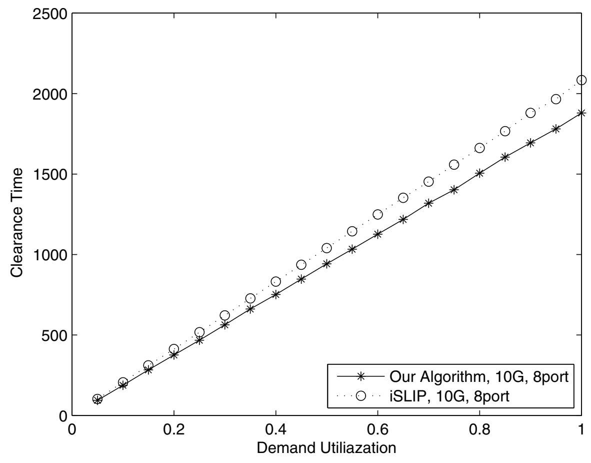


Fig. 2. Comparison between the clearance time of iSLIP and the proposed scheme when the number of input ports N is 8 and the port capacity is 10 Gb/s.

The demand utilization is the ratio between the average per-port traffic load and the per-port capacity. Note that the schedulability region, which depends on the traffic distribution among input ports, is less than one due to the cross-bar constraint explained in Section III-A. In other words, because of contention among ports, not every packet at an input port can be switched in a given time slot. For a given demand utilization, the corresponding number of packets are generated and assigned to a random input port.

As shown in Fig. 1, the schedulability region of the proposed approach is larger than that of iSLIP in all cases. In addition, as the number of input ports increases, the difference becomes smaller. This phenomenon may be due to the aggregation of random effects of uniform packet distribution among input ports. However, it should be noted that our primary application scenario of the proposed switch framework is an embedded system, where the number of input ports is relatively smaller than the typical Internet switch.

B. Clearance Time

In Fig. 2, we plot the clearance time (in time slot) of each scheme by increasing the demand utilization. Note that each point in Fig. 2 is an average value of 1000 simulation runs. Consequently, in addition to the guarantee of a bounded delay for any feasible traffic, Fig. 2 shows that our framework further provides a better average clearance-time performance than iSLIP.

C. Effect of the Packet Size on Switch Performance

Finally, we evaluate the effect of the packet size on switch performance. We consider the total execution time, which is defined as the service time by the switch plus the switching overhead. Here, switching overhead is the time to be taken for reconfiguration of the switch fabric. We consider 10 ns for the switching overhead of an optical switch. Since we use a fixed clock period of 1 ms, the total number of time slots in one

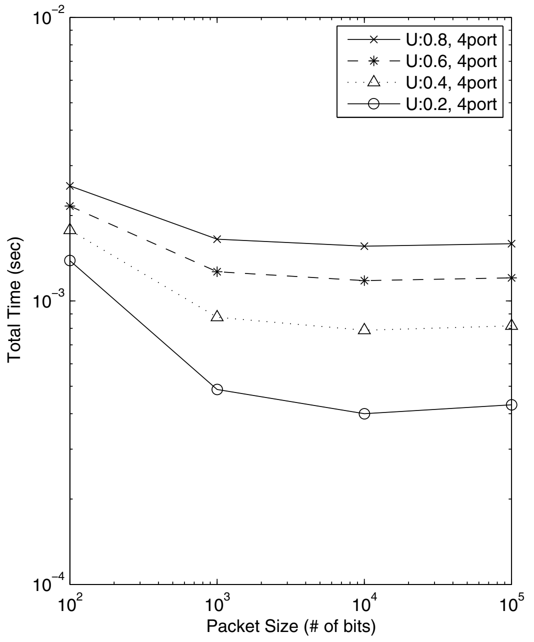


Fig. 3. Total execution time (service time plus switching overhead) versus the packet size when the number of input ports is 4 and the utilization is 0.2, 0.4, 0.6, and 0.8.

clock period will decrease as the packet size increases with a given per-port capacity. Qualitatively, the overall switching overhead in a clock period decreases with the packet size. However, because the decreasing granularity with the packet size may increase the service time. Our simulation result in Fig. 3 shows the total execution time as a function of the packet size for different values of demand utilization. The number of input ports is 4 in the figure. It is notable that the total execution time is not monotonically decreasing with the packet size. It first decreases with the packet size, but slightly increases with the packet size of 10^5 bits. Hence, if we consider the switching overhead of an optical switch, it might not be the best to adopt the largest-possible packet size. It will be a future research task to identify the optimal packet size for an optical packet switch in an analytical manner.

V. Conclusion

In this paper, we have proposed a design framework for a real-time optical switch on top of the concept of clock-driven scheduling used in [5]. Our framework has several distinguishing features above previous related studies. First, unlike previous work that primarily focused on periodic traffic [5], our proposed switching architecture can provide a guaranteed delay for any feasible traffic. Second, in our design, we explicitly take into account the realistic region of system parameters for an optical switch, which enables hardware-software codesign in practice. Our simulation study validates that our switch design gives better schedulability region and the clearance time than the iSLIP scheme.

References

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

[2] M. J. Neely, E. Modiano, and Y.-S. Cheng, “Logarithmic delay for $n \times n$ packet switches under the crossbar constraint,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 657–668, June 2007.

[3] G. R. Gupta, S. Sanghavi, and N. B. Shroff, “Node weighted scheduling,” in *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems (Sigmetrics 2009)*, June 15–19 2009.

[4] S. Gopalakrishnan, M. Caccamo, and L. Sha, “Switch scheduling and network design for real-time systems,” in *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2006)*, April 4–7 2006.

[5] Q. Wang, S. Gopalakrishnan, X. Liu, and L. Sha, “A switch design for real-time industrial networks,” in *Proceedings of the 14th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2008)*, April 22–24 2008.

[6] D. K. Hunter and I. Andonovic, “Approaches to optical Internet packet switching,” *IEEE Communications Magazine*, vol. 38, no. 9, pp. 116–122, September 2000.

[7] T. S. El-Bawab and J.-D. Shin, “Optical packet switching in core networks: Between vision and reality,” *IEEE Communications Magazine*, vol. 40, no. 9, pp. 60–65, September 2002.

[8] G. I. Papadimitriou, C. Papazoglou, and A. S. Pomportsis, “Optical switching: Switch fabrics, techniques, and architectures,” *IEEE/OSA Journal of Lightwave Technology*, vol. 21, no. 2, pp. 384–405, February 2003.

[9] L. Xu, H. G. Perros, and G. Rouskas, “Techniques for optical packet switching and optical burst switching,” *IEEE Communications Magazine*, vol. 39, no. 1, pp. 136–142, January 2001.

[10] P. B. Chu, S. S. Lee, and S. Park, “MEMS: The path to large optical cross-connects,” *IEEE Communications Magazine*, vol. 40, no. 3, pp. 80–87, March 2002.

[11] V. Eramo and M. Listanti, “Packet loss in a bufferless optical WDM switch employing shared tunable wavelength converters,” *IEEE/OSA Journal of Lightwave Technology*, vol. 18, no. 12, pp. 1818–1833, December 2000.

[12] S. Yao, B. Mukherjee, and S. Dixit, “Advances in photonic packet switching: An overview,” *IEEE Communications Magazine*, vol. 38, no. 2, pp. 84–94, February 2000.

[13] I. Glesk, K. I. Kang, and P. R. Prucnal, “Ultrafast photonic packet switching with optical control,” *Optical Express*, vol. 1, no. 5, pp. 126–132, September 1997.

[14] M. Murata and K. Kitayama, “Ultrafast photonic label switch for asynchronous packets of variable lengths,” in *Proceedings of IEEE INFOCOM*, June 23–27 2002.