HELIOS: A High Energy-efficiency Locally-scheduled Input-queued Optical Switch

†Shunyuan Ye, ‡Yanming Shen, †Shivendra Panwar
†ECE Department, Polytechnic Institute of NYU
‡School of Computer Science and Technology, Dalian University of Technology, China
{sye02@students, syanming@photon, panwar@catt}.poly.edu

ABSTRACT

Fast growing traffic for both the Internet and within data centers has lead to an increasing demand for high-speed switching systems. In this paper, we propose a fully distributed scheduling algorithm with an $O(1)$ complexity, for a switch with an optical switching fabric. The inputs only use local queue information to make their scheduling decisions, and the switch consumes much less power than an electronic switch. Therefore, we call the switch HELIOS: High Energy-efficiency Locally-scheduled Input-queued Optical Switch. HELIOS can achieve 100% throughput for any admissible Bernoulli i.i.d traffic. To our knowledge, this is the first distributed scheduling algorithm to guarantee 100% throughput for an input-queued optical switch.

1. INTRODUCTION

Internet traffic has been increasing at a very fast pace. Emerging websites and applications demand ever increasing bandwidth. For fast storage-access networks like a data center, thousands of servers have to exchange bulk data at a very high speed. Under this fast growing traffic demand, electronic technologies may not be able to support the demands of packet switches in the near future. Photonic technologies, on the other hand, can provide very high bandwidth and reduce the power consumption. But an all-optical switch is still far from feasible due to the fact that it is very difficult to buffer packets in the optical domain. Therefore, future switching systems may have a hybrid architecture, which exploits both electronic and optical technologies. In this paper, we adopt a switch architecture with a *broadcast-and-select* optical switching fabric. The fabric can be as simple as a WDM fiber ring [1], which is shown in Fig. 1.

Each linecard is equipped with a tunable laser as the transmitter (TX) and a fixed wavelength receiver (RX). For example, the TX of linecard 1 can transmit in any wavelength, but its RX can only receive packets transmitted using wavelength λ_1. If linecard 1 wants to send a packet to linecard 4, it has to tune its laser to λ_4, and make sure that no other linecard is transmitting using this wavelength. Otherwise, there will be a *collision*. We assume that the system is slotted, and all the linecards are synchronous. The packet arrival process is Bernoulli i.i.d..

ANCS'10, October 25-26, 2010, La Jolla, CA, USA.

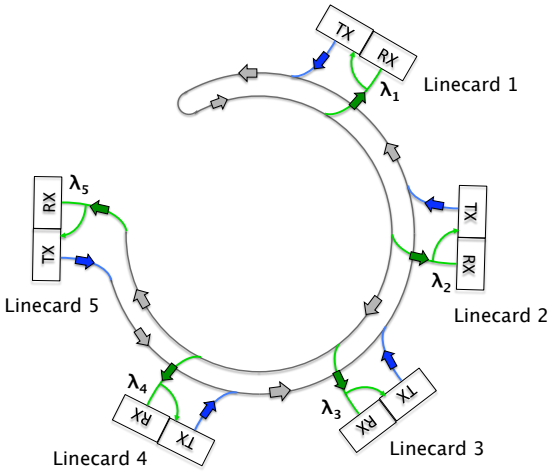

Figure 1: Switch Architecture

2. HELIOS SCHEDULING ALGORITHM

In order to avoid collisions, we require that each linecard has a module called a λ-*monitor*. It can detect which wavelengths are being used in the fiber, by detecting energy across all wavelengths from the received signal. The linecard architecture is shown in Fig. 2.

Though HELIOS is distributed and has O(1) complexity, it can achieve 100% throughput (the proofs are available on request). The intuition is that since there is at most one arrival in a time slot for each input, a schedule with a heavy weight (defined as the sum of a concave function of queue lengths at the linecards) will continue to be heavy over a few time slots. So the schedule of the previous time slot provides some information that we can utilize. HELIOS adds (or removes) edges to (or from) the schedule of the previous slot with a probability which is a function of the queue size. After the system converges, the schedule generated at every time slot is very close to the one with maximum weight, and therefore it can stabilize the system.

Following the HELIOS algorithm, each input i has to keep track of the schedule of the previous time slot, i.e., for which j was $S_{ij}(n-1) = 1$. At the beginning of time n, generate an input/output permutation $\mathbf{H}(n)$ by using a Hamiltonian

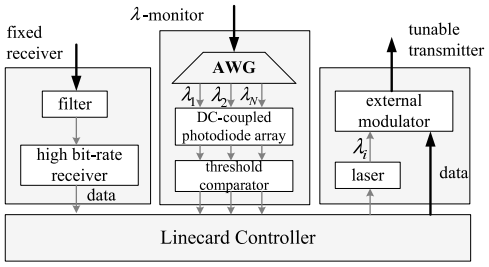

Figure 2: Linecard Architecture

walk. A Hamiltonian walk can be generated with a simple distributed algorithm with a time complexity of $O(1)$ [2]. After $\mathbf{H}(n)$ is generated, each input can make its scheduling decision following the algorithm below.

The HELIOS Scheduling Algorithm

At each input port i, assume (i,j) is selected by $\mathbf{H(n)}$, i.e., $H_{ij}(n)=1$.
- For any (i,k) that $k \neq j$:
  (a) $S_{ik}(n) = S_{ik}(n-1)$.
- If (i,j) was in $\mathbf{S}(n-1)$:
  (b) $S_{ij}(n)=1$ with probability p_{ij};
  (c) $S_{ij}(n)=0$ with probability $\overline{p}_{ij}=1-p_{ij}$.
- Else, if there is no j' such that $S_{ij'}(n-1)=1$ (input i was not transmitting at time $n-1$) , and output j was not receiving a packet at time $n-1$ (input i knows this from λ-monitor),
  (d) $S_{ij}(n)=1$ with probability p_{ij};
  (e) $S_{ij}(n)=0$ with probability $\overline{p}_{ij}=1-p_{ij}$.
- Else,
  (f) $S_{ij}(n)=0$.

p_{ij} is defined as $p_{ij}=\frac{\exp(w_{ij}(n))}{1+\exp(w_{ij}(n))}$, where $w_{ij}(n)$ is a function of the queue length at input i for output j. In order to make sure that the system can converge as the Glauber dynamics [3], we define the weight function as: $w_{ij}(n)=f(x)=\frac{\log(1+x)}{\log(e+\log(1+x))}$ [4].

3. AN EXAMPLE

In Fig. 3, we give an example of the schedule evolution over time. Even though the optical switch does not have a physical crossbar, we still use it in the example since it can better illustrate the matching.

Assume that at the beginning, as shown in Fig. 3(a), $\mathbf{S}(n)=\{(1,1),(2,3)\}$ and $\mathbf{H}(n)=\{(1,3),(2,1),(3,2)\}$. For input 1, since $(1,1)$ and $(1,2)$ are not selected by $\mathbf{H}(n)$, $S_{11}(n)=S_{11}(n-1)=1$ and $S_{12}(n)=S_{12}(n-1)=0$. $(1,3)$ is in $\mathbf{H}(n)$, but input 1 knows that one of its neighbors, which is $(1,1)$, was in the schedule. Thus, according to the case (f) of HELIOS, $S_{13}(n)=0$. The case of input 2 is similar. For input 3, $(3,2)$ is selected by $\mathbf{H}(n)$. Input 3 already knows that crosspoints in the third row of the crossbar were not active. From the λ-monitor, it also knows that output 2 was not receiving any packet in the previous slot, namely, crosspoints in the second column were not active. So none of the neighbors of $(3,2)$ was active at time $n-1$. According to case (d) and (e) of HELIOS, input 3 can turn on $(3,2)$ with a probability p_{32}, which is a function of Q_{32}. In the example, input 3 decides to make $(3,2)$ active. The new schedule $\mathbf{S}(n+1)$ is shown in Fig. 3(b).

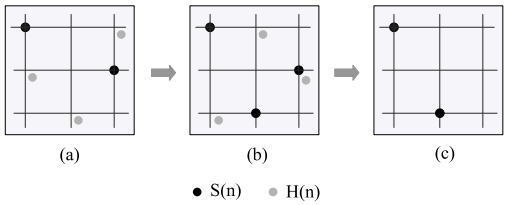

Figure 3: Schedule Evolution Example

In the next slot, $\mathbf{H}(n+1)=\{(1,2),(2,3),(3,1)\}$ (Fig. 3(b)). For input 1 and 3, they keep the schedule of the previous time slot, as their active crosspoints are not selected by $\mathbf{H}(n)$. For input 2, the active crosspoint $(2,3)$ is selected. It has to make a decision for $(2,3)$ to stay active or not according to case (b) and (c) of HELIOS. In the example, $(2,3)$ becomes inactive, and the new schedule is shown in Fig. 3(c).

4. EVALUATION

In order to evaluate the delay performance of HELIOS algorithm, we have run simulations for different scenarios. Uniform and two non-uniform traffic scenarios are studied, including a) Lin-diagonal: Arrival rates at the same input differ linearly, i.e, $\lambda_{i(i+j \pmod N)}-\lambda_{i(i+j+1 \pmod N)}=2\lambda/N(N+1)$; b) Hot-spot: For input port i, $\lambda_{ii}=0.5\lambda$ and $\lambda_{ij}=(1-0.5)\lambda/(N-1)$, for $i \neq j$ (λ is the traffic load). Sample results are shown in Fig. 4. We can see that when traffic is not heavy (i.e., $\lambda < 0.8$), the delay performance is reasonable, i.e. under $100\mu s$. We are assuming that the line speed is 100Gbps and each packet has a size of 64 bytes.

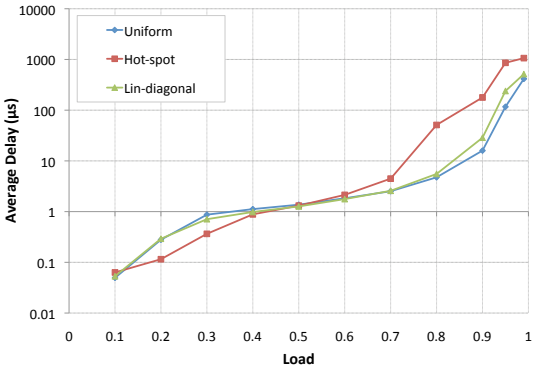

Figure 4: Simulation result, N=16

5. REFERENCES

[1] A. Bianco, E. Carta, D. Cuda, J. M. Finochietto, and F. Neri. A distributed scheduling algorithm for an optical switching fabric. In *Proc. of IEEE ICC*, Beijing, China, May 2008.

[2] A. Nijenhuis and H. S. Wilf. *Combinatorial Algorithms for Computers and Calculators* . Academic Press, 1978.

[3] S. Rajagopalan and D. Shah. Aloha That Works. *submitted*, Nov. 2008.

[4] S. Ye, Y. Shen, and S. S. Panwar. DISQUO: A Distributed 100% Throughput Algorithm for a Buffered Crossbar Switch. In *Proceedings of IEEE Workshop on High Performance Switching and Routing*, Dallas, Texas, June 2010.