Parallel Distributed Schedulers for Scalable Photonic Integrated Packet Switching

Paris Andreades
Optical Networks Group, Electronic & Electrical Engineering
University College London
London, UK
paris.andreades.09@ucl.ac.uk

Georgios Zervas
Optical Networks Group, Electronic & Electrical Engineering
University College London
London, UK
g.zervas@ucl.ac.uk

***Abstract*—As data centre traffic dynamics are changing, optical networking is becoming increasingly important for low-latency, high bandwidth intra-data centre communication. Nanosecond-reconfigurable, scalable photonic switch fabrics and advances in photonic integration are key enablers for optical packet switching. However, the control plane and in particular the switch scheduler is believed to be a critical factor on packet latency and scalability. To that end, we report a low-latency scheduler for Clos-network switch fabrics based on a fixed path assignment scheme and parallel and distributed path arbitration. Cycle-accurate network emulation results show nanosecond average latency at input port loads up to 60% of capacity for a 256x256 switch size. In comparison to previous work, the scheduler can now control a switch 8 times the size at double the input port saturation load for the same average latency. Scaling the switch from 16 to 256 ports shows only a small drop in saturation load from 70% to 60%. Also fairness on a per flow basis is demonstrated.**

***Keywords*—switch scheduling, Clos-network switch, optical packet switching**

I. Introduction

The ever increasing annual global data centre traffic is estimated to reach 20.6 zettabytes by 2021 and more than 70% of it will be due to intra-data centre communication. As a result, the data centre topology has now shifted from the traditional hierarchical tree to the leaf-spine (Fig. 1), in order to improve the network latency and bandwidth. Nonetheless, the fastest commercial electronic switches have a minimum latency on the order of 200 ns [1] and limited switching capacity. Optical switching has been proposed to address the network bandwidth requirements, however it is in the form of millisecond-reconfigurable optical circuits. Nanosecond-reconfigurable optical switch fabrics [2] have been demonstrated with photonic integrated circuit implementations scaling up to 128 ports [3]. At these sizes, such architectures could be used at the leaf layer as optical top-of-rack (ToR) packet switches [4, 5]. However, as their radix scales they can also be deployed at the spine layer, given their scheduler also scales while maintaining good latency and throughput performance, approaching the target area shown in Fig. 1. In the following sections we discuss our system concept and present a new parallel and distributed Clos switch scheduler with a fixed path assignment scheme. Its scalability, latency

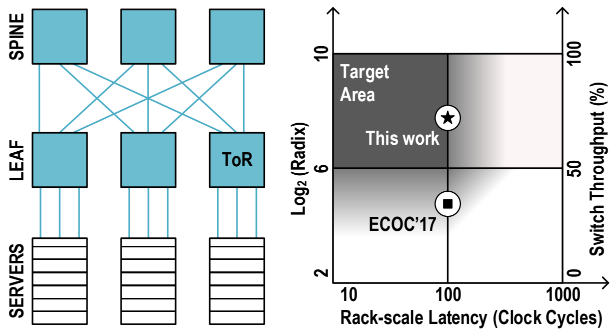

Fig. 1. Leaf-spine data centre network and target switch design area.

performance and fairness are evaluated under a synthetic workload in a cycle-accurate network emulator.

II. System Concept

The proposed optical top-of-rack (ToR) switching system is presented in Fig. 1. Packets are first buffered at the server network interfaces in first-in, first-out (FIFO) buffers. Every server network interface with a non-empty buffer issues a path request, for the head-of-line (HOL) packet in its buffer, to the switch scheduler shown in Fig. 2. There, path allocation takes place before the optical switch is reconfigured. Meanwhile, the HOL packet is held at the server interface for a configurable number of clock cycles before transmission. This allows the scheduler to complete allocation and switch reconfiguration by the time the optical packet arrives at the switch. The packet is segmented and modulated onto different wavelengths, using wavelength-division multiplexing (WDM), increasing the input port bandwidth bandwidth and reducing the packet head-to-tail latency. Furthermore, every packet is transmitted without waiting for a grant from the scheduler. In this way, the control delays for grant transport and grant synchronization and processing at the server interface are eliminated. In case a request fails allocation, the scheduler stores the packet in a corresponding FIFO buffer at the switch input port, according to its destination. Arranging the switch buffers as virtual output queues (VOQs) reduces queuing latency, compared to having a single FIFO, by eliminating the HOL blocking due to packets destined to different switch output ports. Any packet buffered at the switch will be delivered to its destination in a subsequent scheduling round, giving always priority to requests from the switch buffers over new requests from the servers, in order to

The work was partially supported by the EU's H2020 funded dRedBox project (no. 687632) and EPSRC project TRANSNET (EP/R035342/1).

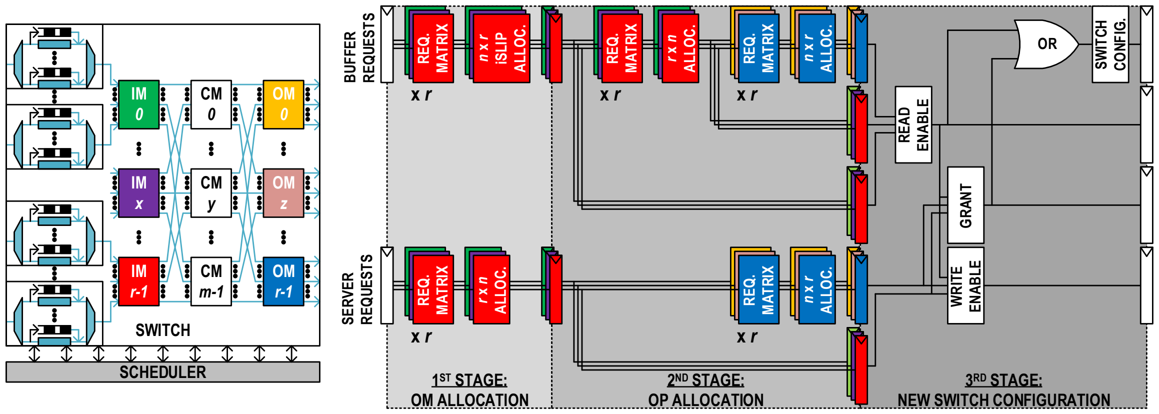

Fig. 2. The optical (m,n,r) Clos switch architecture and scheduler design. Color coding illustrates distributed allocation.

maintain packet ordering and also reduce the total end-to-end latency. A backpressure mechanism is in place to avoid packet loss; if any of the switch VOQs gets full, a signal is asserted to notify the corresponding server interface to temporarily halt transmission until there is space in the VOQ.

III. Scheduler Design

As opposed to the software looping algorithm for routing a Clos network [6], which in hardware would require multiple clock cycles to search and assign a free path, our scheduler attempts to allocate a path in a single clock cycle without searching the available paths. The scheduler is designed for $m = n = r = \sqrt{N}$ Clos fabrics, which are attractive for photonic integration. The input path requests are generated based on the input module (IM) and output module (OM) involved in a packet flow. More specifically, a packet arriving at input port i and destined to output port j would be requesting the path p according to the following equation:

$$p(i,j) = (\lfloor j/r \rfloor + \lfloor i/n \rfloor)\ \%\ m \qquad (1)$$

where $0 \leq i,j \leq N$-1. Using this path request scheme, an IM output port (or path) is reserved for packets to a specific OM and the OM-path reservation is different for each IM to guarantee no contention at the central modules (CMs). In order to assign a path in our space switch, an input port has to first be allocated the destination OM and then the destination port. As a result, this design uses only small n-bit and r-bit arbiters to perform allocation for each IM and OM. Therefore allocation for new server requests and allocation for switch buffer requests is each distributed and they both run in parallel, as shown in Fig. 2. Round-robin arbitration is used throughout the design.

A simplified block diagram illustrating the circuit design is shown in Fig. 2. The circuit is divided into 3 pipeline stages: (1) *OM allocation*, (2) *output port allocation* and (3) *new switch configuration*. In the first stage, for each IM, an r x n OM allocator is implemented as r n-bit arbiters, each choosing one out of at most n new requests for an OM. Similarly, for the VOQ buffer requests, an input-first n x r separable allocator [6] is implemented for each IM to select one OM request from every input port and then to select one input port for every OM. The iSLIP method [6] is used to stagger the priority pointers of the arbiters. In the second stage, for every OM, an n x r output-port allocator is implemented to choose one out of at most r new requests for every output port, only if those requests have won the OM allocation in the previous pipeline stage. In the case of the VOQ buffer requests an additional round of allocation is needed prior to OM allocation. This is because there could be up to n output port requests for an OM and hence an allocator is used to choose only one. The two allocators operate in a master-slave configuration divided in two pipeline stages. Only the wining requests in both allocators are considered for the final output port allocation. In both pipeline stages, the allocators for the VOQ buffer requests and the allocators for the new requests operate independently and in parallel. In the last pipeline stage the new switch configuration, including the VOQ buffer control signals (write-enable, read-enable), are generated. A path request from the switch buffers winning at all three rounds of allocation generates a read-enable signal, releasing the corresponding packet from the VOQ. Similarly, a new request winning at both allocation rounds and not contenting with a buffer request for any part of the path, generates a grant signal. Otherwise, a write-enable signal is asserted to store the incoming packet at the switch in the corresponding VOQ. The logical OR operation is performed on the grant and read-enable matrices to generate the new switch reconfiguration.

IV. Results and Discussion

A full rack-scale cycle-accurate emulation of our proposed optical switching system has been developed in SystemVerilog (hardware description language for register-transfer level digital design and hardware implementation) and used to capture the latency characteristics of our control plane and evaluate our Clos scheduler performance. The packet latency is measured under a synthetic workload. For an NxN switch, N independent packet sources are instantiated each generating 64-byte packets following a uniform random distribution and feeding a network interface. The probability of packet injection in any given clock cycle is given by a universal input port load parameter; an input port operating at load 100% of capacity would have packets arriving on every clock cycle. The switch therefore does not operate in time slots as packets from different sources are generated asynchronously. The packets are assumed to be wavelength-striped on 8 wavelengths and serialized at 25 Gb/s, yielding an input port capacity of 200 Gb/s. The packet destinations are uniform random. Also, N^2 VOQs per input port are instantiated to account for the queuing delay at the switch. Each VOQ has a depth of 128

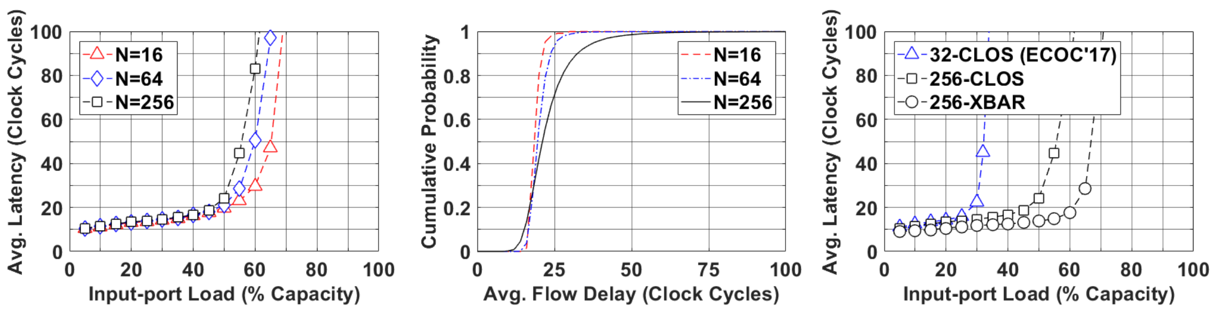

Fig. 3. Average end-to-end latency (a) for different switch sizes, (b) CDF for all flows and (c) for different scheduler designs.

packet entries, although depths as small as 8 entries may be used when backpressure is used in the emulation. For the end-to-end latency calculation, packets are first timestamped when generated and then once more when received. The minimum latency includes propagation delay (4 clock cycles), packet serialization delay (1 clock cycle), scheduling (3 clock cycles) and buffering (1 clock cycle) in each packet source and network interface.

Figure 3(a) shows the average end-to-end latency as a function of the input port load, for different *N*x*N* switch sizes. Here we consider the (4,4,4), (8,8,8) and (16,16,16) Clos configurations for *N* = 16, 64 and 256 respectively. At low loads, for all switch sizes, the average latency performance approaches the minimum end-to-end latency. As the load is increased, path contention increases and packets are queued in the VOQ buffers at the switch, and the queuing delay is longer for higher loads. Based on previous scheduler implementation results on FPGAs [4, 5], the new scheduler using only √N-bit arbiters should be able to achieve a 10 ns clock period, for all switch sizes. Hence, sub-microsecond latency performance is achieved for loads up to 60% of port capacity for *N*=256 and slightly higher saturation loads are shown for the smaller switch sizes, due to the lower contention probability. However, only a small 5% decrease in saturation load per 4x increase in switch size is observed, which demonstrates potential scalability to a 1024-port switch. This is due to the distributed allocation design in which the arbiters scale as √N. Figure 3(b) shows the cumulative distribution function of the average flow delay at an input port load of 50% of capacity, for N = 16, 64 and 256. Based on the figure, 90% of the flows in the 256x256 switch have an average delay less than 32 clock cycles compared to the total average delay of 25 clock cycles shown in Fig. 3(a). This demonstrates fair allocation of output ports to input ports at high loads even for the largest switch size. In Fig. 3(c), the average end-to-end latency for the 256-port Clos switch is compared to that of an equivalent crossbar switch, as well as to a 32-port (4,4,8) Clos which uses our previous scheduler design [5], based on random path requests. As shown in the figure, compared to our previous work, we can now achieve a 60% saturation load for a 256-port Clos switch; a 30% increase for a switch size 8 times larger than before. The significant performance gain is mainly due to (a) routing scheme, (b) distributed allocation which keeps the scaling penalty small and (c) VOQ buffer arrangement per input port which reduces queuing latency. The difference in latency between Clos and crossbar is a performance metric for our routing scheme since the crossbar is a single-stage fabric and requires no routing. The crossbar scheduler has only 2 pipeline stages and therefore its minimum latency is a clock cycle less in this network emulation, as shown in the figure. Also, being a strictly non-blocking fabric, the crossbar system has a lower contention probability which in turns leads to lower latency and higher saturation load compared to the Clos switches. Our Clos scheduler trades off contention probability for very low latency by not searching for an available path. Nonetheless, the results show only a small 10% routing penalty.

V. Conclusion

A new parallel distributed scheduler design has been presented for Clos-network switch fabrics. The design applies to 3-stage m = n = r = √N Clos fabrics which are attractive for fabrication unlike crossbar fabrics. Building the scheduler using only √N-bit arbiters makes the control design also scalable in comparison to our previous work. A cycle-accurate network emulator was used to evaluate the scheduler performance. The results show nanosecond switching for input port loads as high as 60% of capacity for a 256-port switch compared to only 30% for a 32-port switch using our previous scheduler design. The new design shows only a small 10% routing penalty compared to a crossbar switch.

References

[1] "Cisco Nexus 3548 switch performance validation," White paper, Cisco Systems, Inc., 2012.

[2] F. Yan, X. Xue, and N. Calabreta, "HiFOST: A scalable and low-latency hybrid data center network architecture based on flow-controlled fast optical switches," Journal of Optical Communications and Networking, vol. 10, no. 7, pp. B1-B14, July 2018.

[3] Q. Cheng, A. Wonfor, J. L. Wei, R. V. Penty, and I. H. White, "Demonstration of the feasibility of large port count optical switching using a hybrid MZI-SOA switch module in a recirculating loop," Optics Letters, vol. 39, no. 18, pp. 5244-5247, September 2014.

[4] P. Andreades, Y. Wang, J. Shen, S. Liu, and P. M. Watts, "Experimental demonstration of 75 ns end-to-end latency in an optical top-of-rack switch," in Optical Fiber Communications Conference and Exhibition (OFC), 2015.

[5] P. Andreades, and P. M. Watts, "Low-latency parallel schedulers for photonic integrated optical switch architectures in data centre networks," in European Conference on Optical Communication (ECOC), 2017.

[6] W. J. Dally and B. P. Towles, *Principles and Practices of Interconneciton Networks*. San Francisco, CA: Morgan Kaufmann, 2004.