Multicast scheduling of wavelength-tunable, multiqueue optical data center switches

Citation for the original published paper (version of record):
Keykhosravi, K., Rastegarfar, H., Agrell, E. (2018)
Multicast scheduling of wavelength-tunable, multiqueue optical data center switches
Journal of Optical Communications and Networking, 10(4)
http://dx.doi.org/10.1364/JOCN.10.000353

Multicast Scheduling of Wavelength-Tunable, Multiqueue Optical Data Center Switches

Kamran Keykhosravi, Houman Rastegarfar, and Erik Agrell

***Abstract*—The all-optical switching of multicast flows using star couplers and tunable transceivers is a promising solution for the emerging cloud data center applications. However, the limited tuning range of optical components on one hand and the buffer management challenges for multicast traffic delivery on the other pose a significant impact on the performance of optical multicast scheduling algorithms. Using only one queue per input port results in head-of-line (HOL) blocking and limits the throughput especially for bursty traffic patterns. As the number of possible multicast destinations grows exponentially with the switch size, allocating one queue per destination is not a feasible solution. To resolve HOL blocking, in this paper we consider only a handful of queues per switch input port and devise scalable scheduling algorithms that take into account transceiver tunability constraints. According to our Monte Carlo analysis of a switch with 64 ports and operating under bursty traffic, it is possible to improve the maximum achievable throughput by 44% when the number of queues per port is increased from one to eight. We show that the performance gains due to an increase in the queue count depend on the availability of the spectral resources. With the scarcity of wavelengths, an increase in the number of queues leads to diminishing returns.**

***Index Terms*—data center, multicast traffic, optical packet switching, multiqueue switch, scheduling, star coupler, tunability.**

I. Introduction

THE exponential growth of cloud traffic, the mushrooming of bandwidth-hungry applications, and the ever-increasing diversity of data center traffic patterns are posing overwhelming challenges to traditional electronic data center switching technologies in terms of scalability, power consumption, and resource management. In recent years, optical switching, providing for bit rate transparency, wavelength parallelism, and low energy footprints, has been proposed as a disruptive solution to address the requirements of next-generation data centers [1]–[5].

This work was supported by the Swedish Research Council under grant no. 2014-6230. The simulations were carried out on the resources provided by the Swedish National Infrastructure for Computing (SNIC) at C3SE.

This work was presented in part at the 2017 International Conference on Computing, Networking and Communications (ICNC).

K. Keykhosravi and E. Agrell are with the Department of Electrical Engineering, Chalmers University of Technology, Gothenburg 412 96, Sweden (e-mail: kamrank@chalmers.se; agrell@chalmers.se).

H. Rastegarfar is with the College of Optical Sciences, University of Arizona, Tucson, Arizona 85721, USA (e-mail: houman@optics.arizona.edu).

With a proper architectural design, optical switches can be made flexible to support a variety of traffic demands, including high-bandwidth, point-to-point connections and point-to-multipoint (i.e., multicast) traffic delivery. In fact, many data center applications today depend on multicast communication schemes [3], [6]. The MapReduce application is a prominent example that directs search queries to a set of indexing servers and multicasts executable binaries to a group of servers participating in cooperative computations [6]. Traffic multicasting in data centers helps to increase the throughput of bandwidth-hungry applications, reduce the completion time of delay-sensitive tasks, and save on the network communication resources and energy requirements as significant concerns in cloud data centers [6]–[9].

Supporting all-optical multicasting calls for an optical broadcast medium and proper spectral resource scheduling in the first place. A star coupler can provide such broadcast functionality and has been used to realize several optical multicast switch architectures [10]–[16]. Fig. 1 depicts a baseline optical multicast switch design comprising an $N \times N$ star coupler and interconnecting N nodes (servers) that are equipped with wavelength-tunable transceivers. The switch can be located at different tiers of the data center network, including on top of racks [5]. Each of the N nodes is equipped with a fast-tunable transceiver (capable of tuning over the range of available wavelengths in tens of nanoseconds) for data transmission/reception and a small form-factor pluggable (SFP) transceiver for interfacing with the switch controller [17]. Contending packets are stored at the transmit side of each node (i.e., input queueing). Depending on traffic demands, the switch controller instructs the nodes such that they tune to proper wavelengths for unicast or multicast transmission. As the coupler realizes a shared transmission medium, each transmitted signal is routed to all output ports. The designated receivers of a multicast traffic flow should all be tuned onto the wavelength of the transmitter node. To avoid collisions each active input port should utilize a distinct wavelength. In general, transmitter lasers are not required to be tunable; however, transmitter tunability is desired when the number of available wavelengths is smaller than the switch port count [16], [17].

The optical multicast switch is deployed in a time-slotted data center network. All nodes are directly connected to the switch controller and can be synchronized under its supervision. Synchronization is challenging and costly to achieve in optics and will be performed in the electronic domain (at the nodes connected to the input

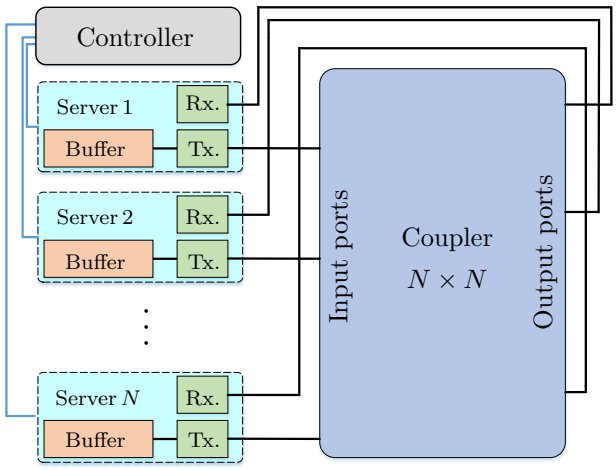

Fig. 1: An all-optical data center multicast switch based on an $N \times N$ star coupler.

ports of the star coupler). However, with the proper choice of fiber lengths, it is feasible to maintain the synchronous status of the packets in the optical domain. We consider an optical packet as a burst of IP packets with the same destination set that are aggregated and synchronized at the edge of the switch. The switching time slot is considered to be long enough to compensate for the synchronization, hardware reconfiguration, and scheduling overheads.

In an optical multicast switch, the number of wavelengths does not necessarily equate the port count, due to potentially a large coupler size and/or fabrication costs and constraints. For instance, a 128×128 star coupler [18, Ch. 4] requires transceivers with a 50.8 nm tuning range assuming 50 GHz channel spacing. However, commercial C-band transceivers can only cover 35 nm (i.e., over 1530–1565 nm). This mismatch is a significant feature in designing a multicast switch scheduling algorithm. While several efforts have tackled the problem of multicast scheduling in electronic packet switches [19]–[23], they only consider nonblocking switch fabrics that allow for simultaneous all-to-all communications, which is not a valid assumption under tunability constraints. Previous work on optical multicast scheduling, on the other hand, is either based on certain restrictive assumptions (e.g., fixed transmission wavelengths or fixed fan-out size) or huge computational burdens on the switch controller (e.g., due to handling a large set of status information as required by reservation-based algorithms) [24]–[27]. In our earlier work [17], we addressed these shortcomings by developing round-robin-based scheduling algorithms that can handle a mix of multicast and unicast traffic patterns in optical packet switches, taking into consideration the wavelength tunability constraints and a single queue per input port.

In this paper, we build on our previous work by exploiting multiple queues per input port to support both multicast and unicast traffic delivery in all-optical switches with wavelength tunability constraints. Buffers are used to resolve resource contentions in packet switches, and the buffering strategy employed has a significant impact on switch performance [28]–[31]. Locating the buffering resources at the output ports of a switch makes it difficult to scale due to the switch fabric speedup requirement. Dedicating only one queue to each input port of the switch, on the other hand, results in head-of-line (HOL) blocking and limits the network throughput. Exploiting multiple queues per input port is not a trivial task under multicast traffic as ideally an exponential number of queues would be required to completely resolve HOL blocking (i.e., $2^{N-1}-1$ queues for a switch with N output ports) [20], [21], [23]. Our focus in this work is to improve the performance of our optical multicast schedulers by resorting to a handful of queues at each input port, without compromising their simplicity. To our knowledge, this is the first effort to combine multiqueue buffering and wavelength tunability constraints for scheduling multicast traffic in optical data centers.

This work only studies the problem of multicast scheduling in a single coupler-based switch (as depicted in Fig. 1), which can achieve performance interesting for a data center environment. We do not study the data center network architecture design problem in this paper. However, the baseline multicast switch can be used as a key building block of the data center, for instance by replacing the electronic ToR switch for flexibility and energy efficiency [5]. As well, it can be employed as a building block of a wavelength-routing architecture to support distributed multicasting in a scalable fashion [16]. While the baseline switch is not scalable from a hardware point of view, we propose scheduling solutions that are simple enough to be adopted and scaled in more complicated switching architectures.

The rest of this paper is organized as follows. Section II details the multiqueue buffering and packet assignment strategy within the optical multicast switch. In Section III, we present two multicast scheduling algorithms that make use of multiple input queues per node and at the same time take into account the wavelength tunability constraints. The two algorithms differ in the way they grant resources to multicast demands. In Section IV, we discuss our performance analysis framework, including the examined traffic patterns in simulations. Section V discusses our simulation results for different tunability constraints, queue counts, fan-out values, and traffic patterns. Finally, Section VI summarizes and concludes the paper.

II. Multiqueue Buffering for Multicast Switching

In designing a proper buffering strategy for a switch, one should consider both the traffic and switch fabric properties. Under uniform traffic, the placement of one queue at each output port of a crossbar switch could lead to 100% throughput; however, an internal speedup is required to avoid buffer overflows [30]. In other words, the switch fabric should operate at a much higher speed compared to the line rate to prevent buffers from building

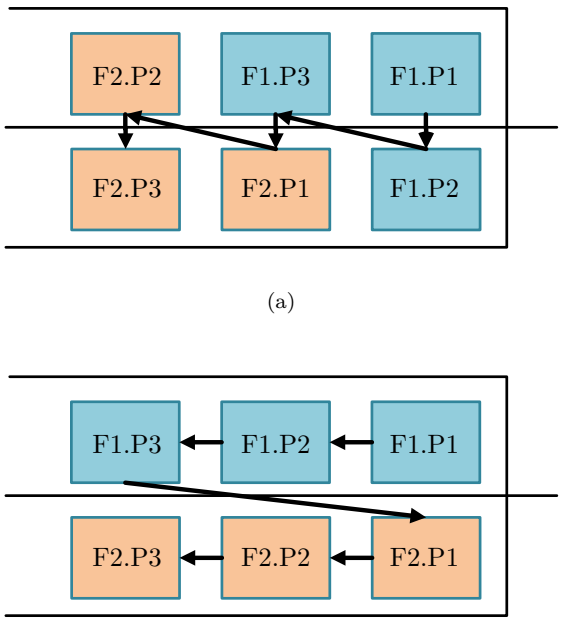

Fig. 2: Two policies for assigning multicast traffic to multiple input queues. Two flows comprising three packets are considered. Fi.Pj denotes the jth packet of the ith flow.

up. To avoid hardware complexities, it is preferred to place the buffers at the inputs of a switch. Placing a single queue at each input port of a switch leads to the undesirable effect of head-of-line (HOL) blocking. HOL blocking implies that as only the head-of-line packet in each queue is allowed to participate in the scheduling, the packets behind the HOL packet have to wait even though their destination ports are free and available [21].

To obviate HOL blocking, a number of techniques exist. One possibility is to look ahead into the queues; that is, instead of only participating the HOL packets in the scheduling, the first few packets per queue are considered for resource allocation. This could enable a non-HOL packet to preempt the packets in front of it. However, such a buffering technique is not helpful in the case of bursty packet arrivals as packets stored back-to-back have a high chance of being destined to the same destination, hence, leading to inadequate scheduling options. An effective solution to resolve HOL blocking in input-queued switches is to use multiple logical queues per input port (one per destination). These queues are called virtual output queues (VOQs) as they are logical and also correspond to output ports. Packets arriving on a switch input port and contending for a certain destination port will be stored in the VOQ corresponding to that destination. Under unicast traffic, each input port of an $N \times N$ switch should be equipped with $N-1$ VOQs to completely resolve HOL blocking. Note that if a packet is received on input port i of a switch, it is considered to not be destined to the output with the same index. Otherwise, the number of VOQs per port should be equal to the port count.

Unfortunately, it is not straightforward to resolve HOL blocking under multicast traffic. The number of possible destinations for a multicast packet in a switch grows exponentially with the switch size. In other words, in an $N \times N$ switch, an incoming multicast packet can be destined to at least one and at most $N-1$ destinations. This gives rise to $2^{N-1}-1$ possibilities. Assigning a VOQ per possible destination set makes the buffer management extremely challenging under multicast traffic.

In this work, we strike a balance between two extremes. Instead of using either only one or $2^{N-1}-1$ queues, we make use of a handful of queues per input port. Although the use of a limited number of queues per input cannot totally resolve HOL blocking, it does improve the performance. Unlike in VOQ switches, input queues are not preassigned to certain output ports and a policy is required for assigning packets to queues. The simplest approach is to assign packets to queues in a round-robin fashion. This method is demonstrated in Fig. 2(a) for two queues per input port and two flows each with three packets.

There are two major problems with the assignment policy in Fig. 2(a). First, packets within the same flow can be missequenced, e.g., in the case where the second packet of the first flow (F1.P2) is transmitted before the first one. Second, it does not necessarily result in destination diversity at the HOL position of the queues. As can be seen in Fig. 2(a), both queues have packets from the first flow at their HOL position. In such a case, only one scheduling choice exists, and the scheduler does not have the freedom to schedule packets from both of the flows.

To address the abovementioned issues, instead of assigning packets to queues in a round-robin fashion we do so with flows. If a newly arriving packet has a different destination set compared to the packet preceding it, it will be stored in the subsequent queue, which is determined by a round-robin pointer. Otherwise, it will be stored in the same queue as its previous packet. Fig. 2(b) illustrates this assignment approach. It can be noted that all packets belonging to the same flow are stored back-to-back, guaranteeing in-order packet delivery and providing more diversity at HOL positions. Unlike in Fig. 2(a), the HOL packets in Fig. 2(b) are from different flows and provide more scheduling options.

III. Multiqueue Scheduling Algorithms

We develop two efficient and simple-to-scale multicast scheduling algorithms that ensure high throughput, low latency, and fair operation. While the integration of heterogeneous services implies different quality-of-service (QoS) requirements for data center applications, in this work we do not consider the problem of scheduling for differentiated services and only focus on the scheduling of equal-priority flows. The problem of multicast scheduling subject to different QoS requirements is an important problem in the context of data center networks and is a potential area for future research.

This section is built upon our previous work in [17], where the problem of scheduling multicast packets under transceiver tunability constraints was studied. In general, the transceivers of an $N \times N$ multicast switch may not be able to tune to N different wavelengths. In other words, the number of available wavelengths, W, can be smaller than N. In [17] we proposed a suite of scheduling algorithms that work under this constraint and compared their performance. These algorithms, however, rely on a single queue per port and suffer from HOL blocking especially in the case of bursty traffic. The essence of this work is to enable multiqueue operation without overlooking the tunability constraints. We introduce two new scheduling algorithms: (1) greedy multiqueue algorithm (GMQA), and (2) multiqueue algorithm minimizing fan-out splitting (MAMFS). With the GMQA, the objective is to introduce a multiqueue scheduling algorithm that is simple, fair, and work conserving. To satisfy the fairness criterion, we design our scheduler based on a round-robin pointer, where the first packet is always transmitted completely, and the rest are transmitted to all of their free destinations until the resources are exhausted or all the packets are checked. With the MAMFS, we try to improve the performance of the GMQA by first trying to schedule the packets without fan-out splitting. Our motivation to do so is based on the results in the key paper of Prabhakar et al. [19], which suggests that avoiding fan-out splitting can result in improved throughput.

It is assumed that the destination set of each multicast packet is encoded into an N-bit string, which is included in the optical packet header. This bit string gets updated upon transmitting the packet to a subset of its destinations. According to Fig. 1, at the beginning of each time slot, each node transmits the destination sets of all HOL packets stored in its multiqueue buffer (that is Q strings of length N bits) to the controller via a dedicated control channel. Upon the completion of the scheduling decisions, the controller transmits to each of the nodes that are allowed to transmit, *i*) an integer number $1 \leq t^c \leq W$, where W is the number of available wavelengths, *ii*) an N-bit string, and *iii*) an integer number $1 \leq q^c \leq Q$, where Q is the number of virtual queues. Each transmitting node will tune its transmitter to the t^cth wavelength and transmit the HOL packet of its q^cth virtual queue. The destination set of this packet will be updated based on the received N-bit string from the controller (if the packet is completely transmitted, it will be deleted from the queue). The controller also sends to the receiving nodes an integer number $1 \leq r^c \leq W$. Each receiving node will tune its receiver to the r^cth wavelength. It is the responsibility of the controller to assign the wavelengths such that packets are delivered to their destinations without collision.

To better describe the different steps of our scheduling algorithms, we use an illustrative example. Fig. 3(a) provides an example of bursty, multicast traffic demands within a 4×4 switch where packets are buffered in two input queues per node. The destination sets are specified on each packet. As can be seen, there is a combination of unicast and multicast traffic in our study. Our algorithms operate such that there is no need to decouple the resource allocation procedures for these two traffic classes.

Both algorithms search among all of the queues and all of the nodes to find the first nonempty queue that belongs to a free transmitter, i.e., a transmitter that has not been scheduled at the time of search. The algorithms first search the HOL positions of a fixed queue index in all of the servers and carry out the search procedure for the subsequent queue indices if necessary. The following pseudocode of algorithm *Search()* illustrates the order of the search. Si.Qj denotes the jth queue of the ith server. The algorithm *Search()* is called multiple times during each time slot (with different initial search points) to find the first nonempty queue with a free transmitter. If all queues are empty, the output of *Search()* will be false.

Algorithm Search(I_Q, I_N)

Inputs: I_Q and I_N indicate the starting point of search; F_T : set of all free transmitters; queue status; Q: number of queues per port; and N: number of nodes.

Output: The server index, i, and queue index, j, associated with the first nonempty queue of a free transmitter. Returns *false* if no queue is found.

```
1: j ← I_Q
2: i ← I_N
3: loop:
4: if i ∈ F_T and Si.Qj is nonempty then return (i, j)
5: i ← (i mod N) + 1
6: if i = I_N then
7:     j ← (j mod Q) + 1
8:     if j = I_Q then return false
9: goto loop
```

For the scheduling algorithms to be fair, the starting point of the search operation should vary in each time slot. Two round-robin pointers are used to determine this starting point, namely, *Node_pointer*, where $1 \leq$ *Node_pointer* $\leq N$ and *Queue_pointer*, where $1 \leq$ *Queue_pointer* $\leq Q$. In the first round of search in each time slot, these pointers determine the initial values for the arguments of *Search()*, i.e, $I_N =$ *Node_pointer* and $I_Q =$ *Queue_pointer*. However, this is not the case for the subsequent search rounds.

A. Algorithm 1: Greedy Multiqueue Algorithm (GMQA)

The GMQA first looks for a nonempty queue by invoking the *Search()* algorithm. For each nonempty queue found, the HOL packet gets scheduled to be sent to all free outputs in its destination set. At the beginning of the scheduling process, the set of free outputs (F_R) includes all output ports of the baseline multicast switch, and will be gradually updated during the scheduling by the controller itself. Here, we first detail the scheduling tasks and then examine its performance according to the example in Fig. 3(a).

GMQA Steps:

1) *Initialize:* Let $F_T = \{1, \dots N\}$ be the set of all free transmitters and $F_R = \{1, \dots N\}$ be the set of all free receivers. Moreover, let I_Q = *Queue_pointer* and I_N = *Node_pointer*. Also, collect the HOLs' destination information.
2) *Select packets and schedule them:* Invoke *Search(I_Q, I_N)* to find the first nonempty queue. If the output is false, go to Step 4. Otherwise, let the output of search be (i, j). Let B be the intersection of the Si.Qj HOL destination set and all free receivers, F_R. If B is empty go to Step 3. Otherwise,
   a) Assign an unused wavelength to the transmitter (first-fit wavelength assigment policy);
   b) Tune all receivers in set B to the assigned transmit wavelength, so that they can receive the HOL packet in the next time slot;
   c) Remove i from the set of free transmitters F_T and also update the set of free receivers F_R.
3) If all wavelengths are used, or all receivers are occupied, or all servers are examined, go to Step 4. Otherwise, let $I_Q = j$ and $I_N = i$ and go to Step 2.
4) *Update:* Update the *Node_pointer* by incrementing it in a circular order. Moreover, if the updated value of *Node_pointer* = 1, also update circularly *Queue_pointer*. This concludes the scheduling process.

GMQA Scheduling Example : Here, we consider the example provided in Fig. 3(a) and go through the steps performed by the GMQA to schedule the packets. Assume *Node_pointer* = 1, *Queue_pointer* = 1, and $W = 4$. The following chart shows the order of the queues that are checked by the GMQA and the actions taken.

Order of Steps Executed by the GMQA
1: **S1.Q1:** The HOL packet is scheduled to be transmitted to outputs 3 and 4.
2: **S2.Q1:** Ignored since it is empty.
3: **S3.Q1:** The HOL packet is scheduled to be transmitted to output 2.
4: **S4.Q1:** Ignored since B is empty.
5: **S1.Q2:** Ignored since transmitter 1 is occupied.
6: **S2.Q2:** The HOL packet is scheduled to be transmitted to output 1.
7: The algorithm stops since all the outputs are busy.

Fig. 3(b) depicts the buffer occupancy status after the scheduling tasks have been performed and the corresponding packets have been dispatched.

B. Algorithm 2: Multiqueue Algorithm Minimizing Fan-out Splitting (MAMFS)

The fan-out is defined as the cardinality of the destination set of a multicast packet. With multicast traffic, fan-out splitting implies transmitting the same multicast packet over several time slots. Ideally a multicast packet should only be transmitted once through the switch fabric. However, depending on the available resources, fan-out splitting could be employed to deliver a multicast packet over several time slots. From an optical switching perspective, when the number of wavelengths is much smaller than the switch port count, it is desirable that each wavelength be utilized as much as possible by favoring multicast transmissions. In such a scenario, minimizing fan-out splitting results in a more efficient use of wavelength resources [17]. Moreover, by avoiding fan-out splitting, more complete packets can be transmitted, opening up the queue HOL spaces for new packets and reducing the penalties of HOL blocking.

In the MAMFS, fan-out splitting is avoided as much as possible to favor entire multicast transmissions. The MAMFS consist of two rounds of search among queues. In the first round, a HOL packet is scheduled if all of its destinations are free. The second round is based on the logic in the GMQA in order to fill up *all* output gaps by fan-out splitting.

MAMFS Steps :

1) *Initialize:* Same as in Step 1 of the GMQA.
2) *Scheduling without fan-out splitting:* Run *Search(I_Q, I_N)*. If the output is false, go to Step 5; otherwise, let it be (i, j). If all of the outputs in the destination set of the HOL packet of Si.Qj are free, perform the following steps; otherwise, go to Step 3.
   a) Assign a wavelength to the transmitter;
   b) Tune all outputs to that wavelength;
   c) Update the sets of free transmitters, F_T, and free receivers, F_R.
3) If all outputs are occupied, or all wavelengths are used, go to Step 5; else if all the servers are examined, go to Step 4; else, let $I_Q = j$ and $I_N = i$ and go to Step 2.
4) *Fill up the void spaces by fan-out splitting:* To schedule the free receivers, perform Step 2 and Step 3 of the GMQA.
5) *Update:* Same as in Step 4 of the GMQA.

MAMFS Scheduling Example: Here, we list the steps performed by the MAMFS in order to schedule the HOL packets in Fig. 3(a). Again, we let *Node_pointer* = 1, *Queue_pointer*=1, and $W = 4$.

Fig. 3(c) illustrates the buffer occupancy status after the execution of the MAMFS. Compared with Fig. 3(b), one can notice that both algorithms transmit to all four outputs. However, due to opposing fan-out splitting, two complete packets are transmitted with the MAMFS while this number is one for the GMQA. As a result, the effect of HOL blocking is reduced under the MAMFS as more new packets make their way to HOL positions.

Order of Steps Executed by the MAMFS

1: **S1.Q1:** The HOL packet is scheduled to be transmitted to outputs 3 and 4.
2: **S2.Q1:** Ignored since it is empty.
3: **S3.Q1:** Ignored since output 4 is busy.
4: **S4.Q1:** Ignored since output 3 is busy.
5: **S1.Q2:** Ignored since transmitter 1 is not free.
6: **S2.Q2:** Ignored since output 4 is busy.
7: **S3.Q2:** The HOL packet is scheduled to be transmitted to output 1.
8: **S4.Q2:** Ignored since output 1 is busy.
9: Since output 2 is free, the scheduler executes Step 2 of the GMQA. The steps performed by the GMQA are as follows.
10: **S1.Q1:** Ignored since transmitter 1 is not free.
11: **S2.Q1:** Ignored since it is empty.
12: **S3.Q1:** The HOL packet is scheduled to be transmitted to output 2.
13: Since all outputs are scheduled, the algorithm stops.

C. Scheduling Algorithm Properties

We examine our algorithms in terms of being scalable, fair, and work conserving, which are the essential requirements of a scheduling algorithm [19, Sec. II-C].

Complexity : It is well known that the scheduling of multicast traffic is an NP-hard problem even with fanout splitting [32]. However, we have been able to develop simple heuristics that offer acceptable performance. The time complexity of both of our scheduling algorithms is $\mathcal{O}(Q \times N)$ as they both should check all HOL packets to perform scheduling. Note that for each HOL packet the intersection of the destination set and all free receivers should be computed, which can be implemented in parallel via N XOR operations. Our algorithms are solely based on comparing binary sequences (via XOR) and no complicated algebraic operations are involved throughout scheduling. Therefore, the proposed algorithms are scalable due to their simplicity and can be applied to high-speed switches with large port counts. We note that in the worst case, the GMQA finalizes the scheduling after one round of search through the HOL packets (checking QN HOL packets), whereas the maximum number of rounds is equal to two for the MAMFS.

Fairness: A packet scheduling algorithm is considered to be fair if there exists a number $K > 0$, such that after K time slots, any arbitrary HOL packet is dismissed, regardless of the queue status [19, Sec. II-A]. A fair scheduling algorithm guarantees that no HOL packet starves for switching resources indefinitely. Note that with GMQA and MAMFS, a HOL multicast packet that is selected by the round-robin pointers (i.e., the first packet that is scheduled) will definitely be transmitted completely as all the receivers are free at the beginning of scheduling. Since the pointers are updated in a round-robin fashion, it takes QN time slots for the pointers to go over all HOL packets and return to their starting position. Therefore, after $QN-1$ time slots, any HOL packet is guaranteed to be transmitted to all of its destinations.

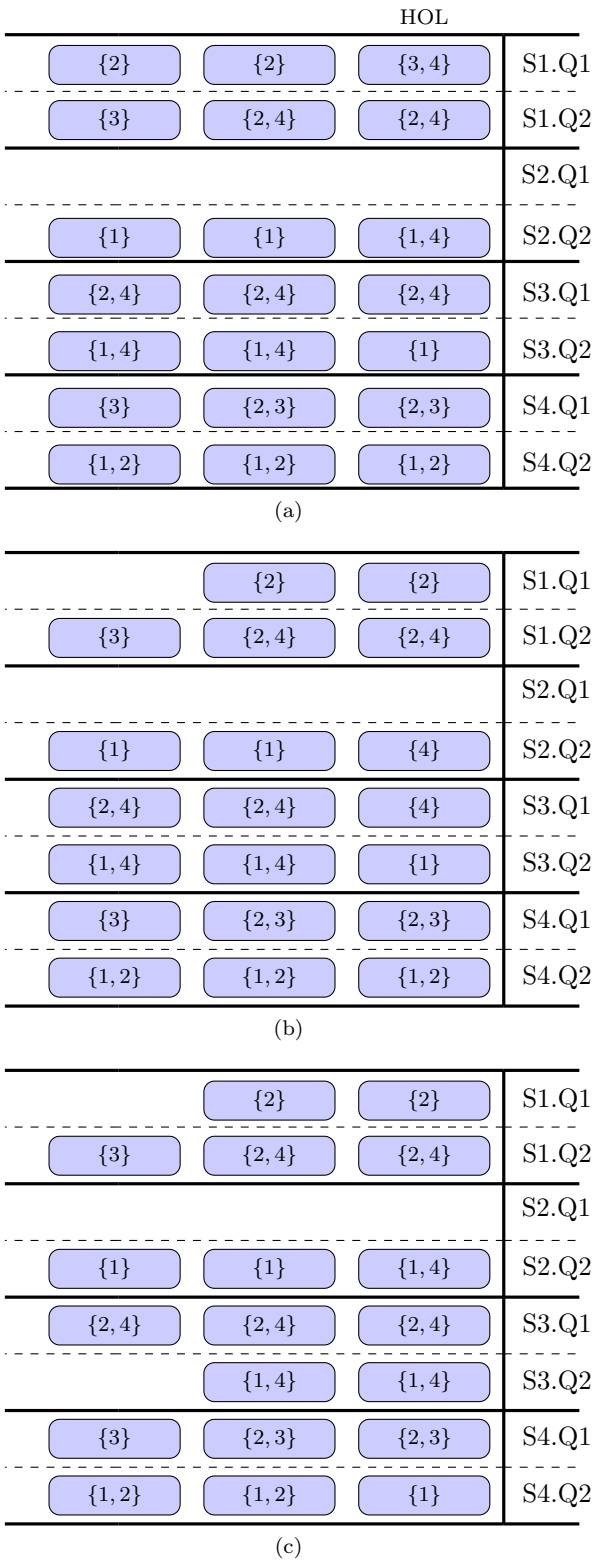

Fig. 3: Example buffer status in a 4×4 switch with two queues per server. Packets' destination sets are shown. Si.Qj is used to denote the jth queue in the ith server. S2.Q1 is assumed to be empty. (a) Initial buffer occupancy status, (b) Buffer occupancy status after the GMQA scheduling, and (c) Buffer occupancy status after the MAMFS.

Work Conservation: A scheduling algorithm is work conserving if it does not leave an output port idle as long as that output can serve some input traffic destined to it [19, Sec. II-A]. GMQA ensures that each packet is transmitted to all of its destinations that are free. In fact, it searches all HOL packets unless it runs out of wavelengths or free receivers, hence, work conserving. MAMFS is also work conserving as it simply runs GMQA at its last step.

D. Application in Data Center Networks

The topology in Fig. 1 and the scheduling algorithms presented in this section are developed for a single coupler-based switch. This switch can be deployed at the edge tier of a data center network. In other words, electronic ToR switches can be replaced with the optical multicast switch for improved flexibility and power efficiency. In a general setting, some coupler ports can be reserved to interface a second switching stage in a hierarchical optical network architecture. This way, the servers within a rack can directly connect to the multicast switch and the reserved ports enable the interconnection of optical ToR switches. By reserving a number of output (input) coupler ports, data center racks can be connected together to transmit (receive) the traffic to (from) other racks.

This concept has been illustrated in [16, Fig. 3], where multiple broadcast domains are interconnected via an arrayed waveguide grating (AWG) core to build an all-optical fabric with the capacity of $K(N-1)$ nodes, with K being the AWG port count. The resulting network is modular and requires distributed scheduling to support scalability. Hence, scheduling can be performed in two steps. First, the interdomain (inter-rack) traffic is scheduled and next the intradomain (intra-rack) traffic. While the work in [16, Fig. 3] disregards the problem of multicast traffic scheduling, our proposed algorithms in this paper can be applied to the second phase of scheduling (i.e. intradomain traffic scheduling).

IV. Simulation Setup

We conduct Monte Carlo simulations to evaluate the performance of our two multiqueue, multicast scheduling algorithms as proposed in Section III. We simulate the switch architecture of Fig. 1. The simulation parameters are summarized in Table I. A total of one million time slots are considered, with the first half only contributing to warm up. To make our simulation run with a reasonable speed, the maximum queue length is set to 1000 packets. This value is large enough as long as the system is stable (i.e., the packet delays are bounded). The number of nodes, N, is set to 64, and three numbers of wavelengths, W, are considered, namely, 16, 32, and 64. For each setting, four different values for the number of queues per input port, Q, i.e., 1, 2, 4, and 8, are studied.

Traffic Model: We examine the multiqueue scheduling algorithms under both uniform (Bernoulli) and bursty traffic patterns. In uniform traffic, independent Bernoulli processes are used to generate packet arrivals on switch input ports. A packet is generated every time slot with probability $0 \leq \rho \leq 1$, which also denotes the average number of generated packets per node per time slot. In a bursty traffic model, flows of correlated packets are generated assuming a geometric distribution for active and inactive periods. In this model, during each time slot, a node is either in the ON or the OFF state. During the OFF period, no packet is generated. In the ON period, packets arrive at every time slot and are all destined to the same set of receivers. The duration of the ON period, g_{on}, follows a geometric distribution with mean $E_{\text{on}} = 1/p_{\text{on}}$, that is

$$\Pr\{g_{\text{on}} = n\} = p_{\text{on}}(1-p_{\text{on}})^{n-1}, \tag{1}$$

where $n = 1, 2, 3, \ldots$, and $0 < p_{\text{on}} \leq 1$. In our simulations, we set $E_{\text{on}} = 16$. Given the average effective load (ρ) and the average ON time (E_{on}), the expectation of the OFF period, E_{off}, is calculated by $\rho = E_{\text{on}}/(E_{\text{on}} + E_{\text{off}})$. The OFF period duration follows the geometric distribution described in (1) with parameter $p_{\text{off}} = 1/E_{\text{off}}$.

In both traffic models, the fan-out is determined by the realization of a random variable with truncated geometric distribution with parameter q ($0 \leq q < 1$). That is,

$$\Pr\{\text{Fan-out} = n\} = \frac{(1-q)q^{n-1}}{1-q^{N-1}} \quad 1 \leq n \leq N-1 \tag{2}$$

The mean fan-out value can be calculated as [21]

$$E[\text{Fan-out}] = \frac{1}{1-q} - \frac{(N-1)q^{N-1}}{1-q^N}. \tag{3}$$

The parameter q is set to 1/2 ($E[\text{Fan-out}] \approx 2$) in the rest of this paper except in Section V-C, where $q = 0$ ($E[\text{Fan-out}] = 1$) and $q = 3/4$ ($E[\text{Fan-out}] \approx 4$) are studied. The destination set of each packet (or flow) is generated by a uniformly random selection of nodes from the set of all destination servers excluding the destination corresponding to the transmitter node itself.

TABLE I: Simulation parameters

Parameter	Value
Number of simulated time slots	$1,000,000$
Number of warm-up time slots	$500,000$
Coupler port count (N)	64
Wavelength count (W)	$16, 32, 64$
Number of queues (Q)	$1, 2, 4, 8$
Maximum queue depth	1000 packets

V. Simulation Results

The queueing performance of the presented switching structure can analytically be evaluated in a few special cases, under Bernoulli and unicast traffic. For instance, if $Q = 1$ and $W = N$ the problem reduces to the well-known input-queued switch problem [28], and if $Q = W = N$, it maps to the the M/D/1 queueing problem [33, Ch. 5]. In general, the problem is analytically intractable, especially considering the complexities of the multicast traffic patterns and the blocking properties of the switch fabric.

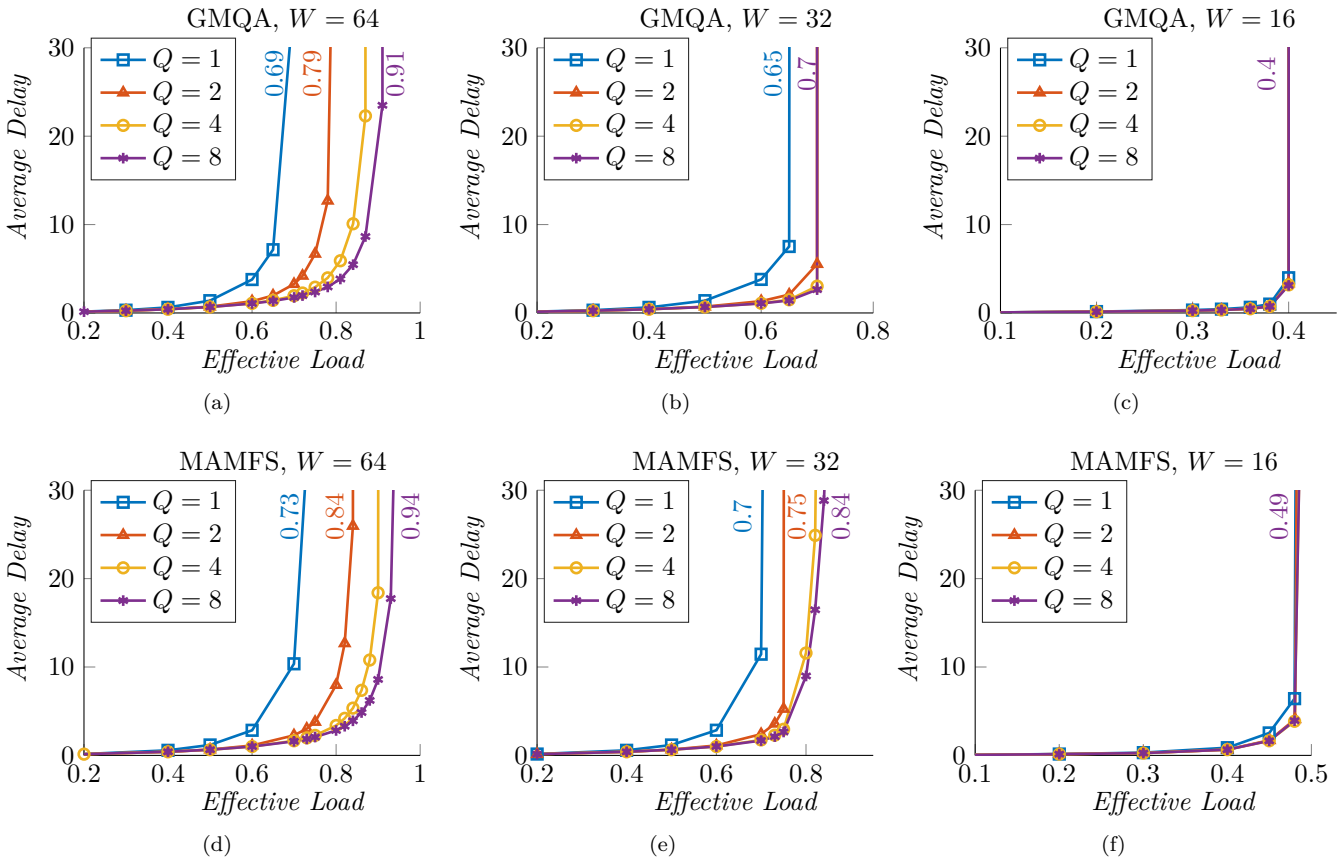

Fig. 4: Simulation results under uniform traffic: Average packet delays are depicted versus the effective load in a 64×64 switch operated by GMQA and MAMFS. Three different numbers of wavelengths (W) are considered, 64, 32, and 16. The number on the side of each curve with the same color denotes the load at which the average delay exceeds the maximum delay shown in the graphs (30 time slots).

Therefore, we perform several Monte Carlo simulations to assess the behavior of our scheduling algorithms. Our simulation results are presented in Figs. 4–7, which depict the performance of the proposed scheduling algorithms when the number of queues per input port is varied. In the special case of $Q = 1$, in [17], the proposed algorithms are compared with an adaptation of the weight-based algorithm (WBA) in [19]. It is shown in [17] that with $W = N$ the proposed algorithms have almost the same delay performance as WBA and that with $W < N$ our algorithms outperform the modified WBA. We will see that with an increase in the queue count, the switch performance keeps improving.

A. *Delay Performance under Uniform Traffic*

In Fig. 4, the average packet delay is depicted versus effective load under uniform traffic, considering the two scheduling algorithms, three values for W, and four values for Q. The delay of a packet is equal to the number of time slots that the packet has spent in the queue before being transmitted to output ports. The reported average values have been calculated by averaging the delay associated with all packets that make it to the output ports of the switch. The effective load is defined as the average utilization of the output ports. It is calculated by dividing the total number of packets that are received during a simulation run by the number of nodes times the number of simulation time slots. Both average delay and effective load quantities are collected from the steady-state phase of simulation runs.

With $W = 64$ (Figs. 4(a) and 4(d)), i.e., with full wavelength tunability, the performance of both the GMQA and the MAMFS improves as Q increases. This is clearly due to the scheduling diversity that can be provided through multiple queues. When the effective load is equal to 0.6, the delay of the GMQA reduces from 3.8 time slots for $Q = 1$ to 1.3 time slots for $Q = 2$, translating to about 65% reduction in the delay. For the MAMFS these numbers are 2.8 time slots for $Q = 1$ and 1.1 time slots for $Q = 2$. Moreover, the maximum throughput achieved by the algorithms increases significantly with an increase in Q. For the GMQA (MAMFS), the maximum throughput increases from 0.69 (0.73) with $Q = 1$ to 0.91 (0.94) with $Q = 8$, which is equivalent to 32% (29%) improvement. Although an increase in the queue count can improve the switch performance, the improvement achieved by increasing Q from 4 to 8 is minimal; i.e., the performance saturates beyond a small Q threshold. Given that the time complexity of our proposed algorithms is linear in Q and that we only need a small number of queues

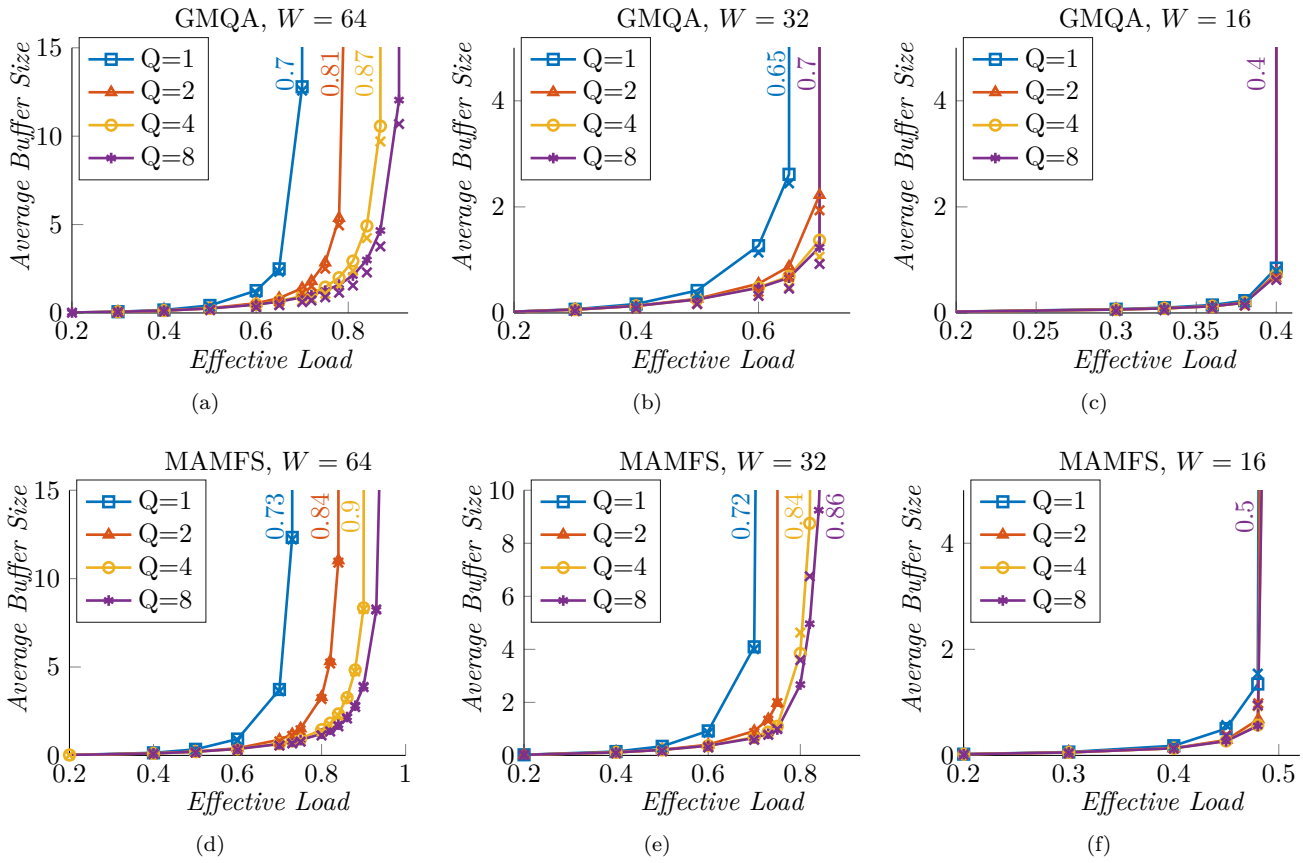

Fig. 5: Simulation results under uniform traffic in a 64×64 switch. The average buffer size is depicted against the effective load for the two scheduling algorithms and three numbers of wavelengths (W). For each setting, the average buffer size calculated by Little's formula is marked by crosses ("$\times$") marks with the same color.

per input port to schedule the multicast traffic, multiqueue buffering does not pose a significant scheduling overhead. In our analysis, $Q = 4$ is the sweet spot in the trade-off between performance gains and scheduling complexity.

With $W = 32$ (Figs. 4(b) and 4(e)), one can observe that the maximum throughput is less than that of $W = 64$, which is expected since there are less resources to be utilized. As with $W = 64$, both algorithms benefit from using multiple queues. However, the gains are less compared to the case of $W = 64$. For the GMQA (MAMFS), the maximum throughput is increased from 0.65 (0.7) under $Q = 1$ to 0.7 (0.84) under $Q = 8$. As the spectral resources are more scarce, the MAMFS offers a better performance compared with the GMQA since it aims at avoiding fan-out splitting as much as possible.

For $W = 16$ (Figs. 4(c) and 4(f)), which signifies the case of significant mismatch between the switch port count and wavelength count, approximately no improvement is achieved by deploying multiple queues. In general, when $W \ll N$, the effect of HOL blocking is minor even for $Q = 1$, and there is no added benefit in improving the HOL diversity through resorting to multiple queues. If $Q = 1$ and all queues are nonempty, the scheduler should choose at most W packets among the total number of N packets. This gives the scheduler enough diversity by itself. Therefore, deploying multiple queues is not beneficial in this setting. From the scheduler operation point of view, if $W \ll N$, then our proposed schedulers will most likely run out of wavelengths by only going through the first round of search on the pointed queue (pointed to by *Queue_pointer*) of each node and need to stop before looking at other queue indices. Hence, the impact of provisioning multiple input queues becomes insignificant.

Note that for an $N \times N$ switch the throughput can be upper-bounded by

$$\text{Throughput} \leq \frac{\text{E[Fan-out]} \times W}{N} \tag{4}$$

where W is the number of available wavelengths. However, (4) is only useful when its right-hand side is less than one. Since in our setup the average fan-out is approximately equal to 2, for $W = 16$ the throughput cannot exceed 0.5. This upper bound is almost achieved by the MAMFS in Fig. 4(f).

It can be observed from Fig. 4 that under uniform traffic, the MAMFS always outperforms the GMQA. This happens due to two reasons. First, reducing fan-out splitting counteracts the effects of HOL blocking. Second, reducing fan-out splitting results in exploiting the resources more efficiently. When the number of wavelengths is small, each wavelength should be utilized to transmit as many packets

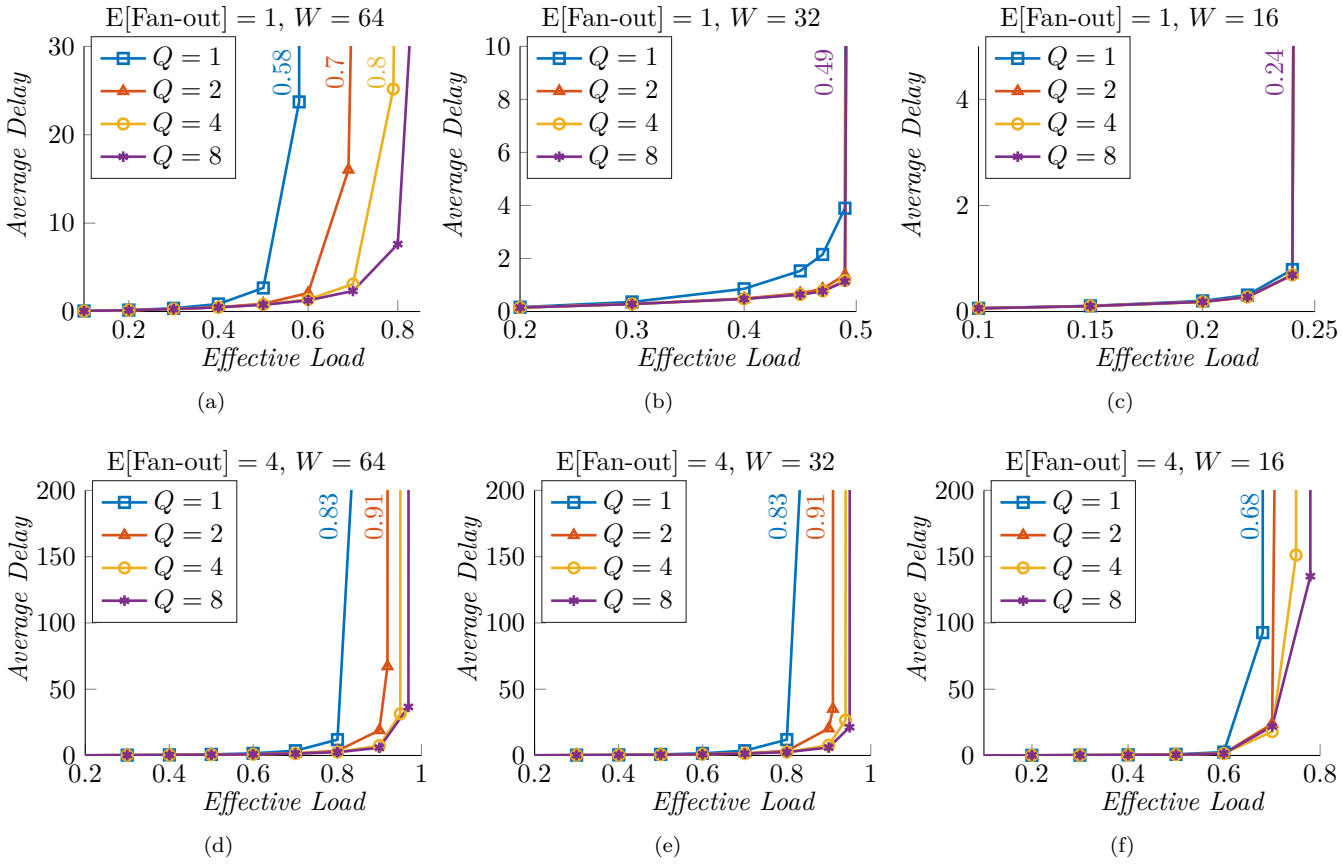

Fig. 6: Simulation results under uniform traffic with two different average fanouts, i.e., 1 and 4. Average packet delays are depicted versus the effective load in a 64×64 switch operated by MAMFS. Three different numbers of wavelengths (W) and four different numbers of queues are considered. The number on the side of each curve with the same color denotes the load at which the average delay exceeds the maximum delay shown in the graphs.

as possible at once. Therefore, the benefits of reducing fanout splitting become more substantial for smaller W. With $Q = 8$ and $W = 32$, the maximum throughput of the MAMFS exceeds that of the GMQA by 20%.

B. Buffer Occupancy under Uniform Traffic

The buffer management in our design involves dividing the available space of an input-stage electronic memory unit into multiple parallel virtual queues. In our simulations we consider a maximum queue size of 1000 packets, because this choice helps us to minimize the runtime of our codes while ensuring that no packet drops occur as long as the switch is stable. In fact, during each simulation time slot, the occupancy of each of the Q queues of each input port can be different depending on the number of packets that each multicast or unicast flow contains. All queues belonging to an input port share the same physical buffer space. Hence, statistical multiplexing can be achieved to save on the amount of required buffering resources.

The average buffer length can be used as a measure to determine the required memory size for each of our schedulers. It denotes the average number of stored packets per input port and is calculated by averaging the sum of all queue lengths at each node over all time slots and all nodes. In Fig. 5, the average buffer length achieved via Monte Carlo simulations is reported for uniform traffic with E[Fan-out] = 2. Comparing Fig. 5 to Fig. 4, we can observe the relationship between the average buffer length and the average delay, as governed by Little's formula

$$\text{Average Buffer Length} = \text{Average Delay} \times \text{Arrival Rate}, \tag{5}$$

where Arrival Rate is the average number of generated packets per input in a time slot and is denoted by ρ in Section IV. We calculated the average buffer length based on (5) using the average delays presented in Fig. 4. The calculated values are depicted by crosses in Fig. 5 and are in good agreement with the results based on Monte Carlo simulations. The results in Fig. 5 illustrate that, to achieve a certain throughput under multicast traffic, the total required memory can be reduced by partitioning the physical buffer space into multiple logical queues, which may have different occupancies.

C. Delay Performance under Unicast and Highly-Multicast Traffic

In this section, we study the impact of multicast fanout on the delay performance of the multiqueue switch. For the sake of brevity, we only focus on MAMFS in this

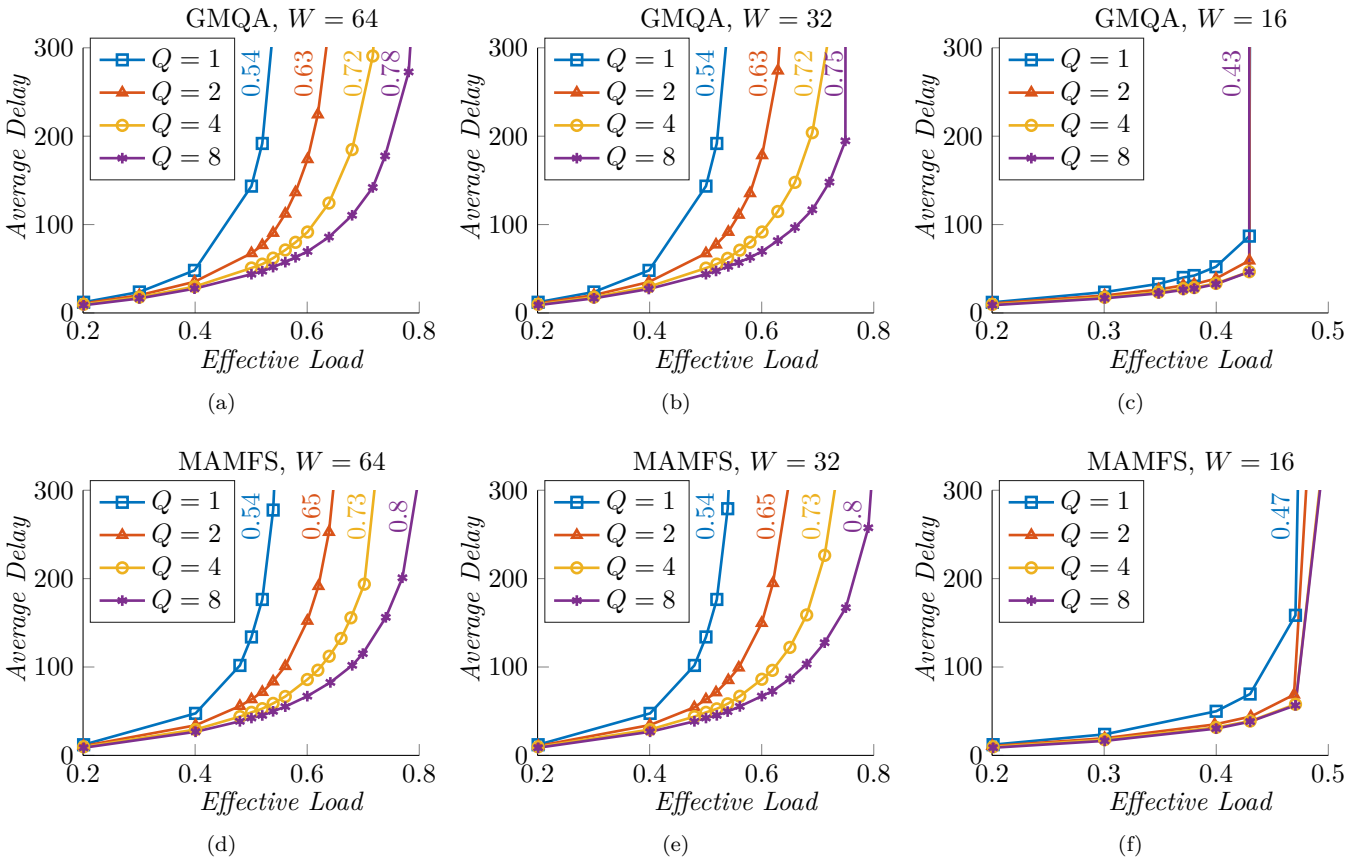

Fig. 7: Simulation results under bursty traffic in a 64 × 64 switch. The average packet delay is depicted against the effective load for the two scheduling algorithms and three numbers of wavelengths (W). The numbers on the side of the curves identify the loads at which the average delay exceeds 300 time slots, which is the maximum delay shown in the graphs.

section. Fig. 6 depicts the average delay versus the effective load for two different values of average fan-out, namely E[Fan-out] = 1 and E[Fan-out] = 4.

With the unicast traffic (E[Fan-out] = 1), full wavelength tunability ($W = 64$), and a single-queue architecture ($Q = 1$), as can be seen in Fig. 6(a), the maximum achievable throughput equals 0.58. This is in perfect agreement with the theoretical maximum throughput that can be achieved in input-queued switches [28]. A performance gain of 43% is introduced by increasing the number of queues from $Q = 1$ to $Q = 8$, which is due to decreasing the penalties of HOL blocking. With $W = 32$ ($W = 16$), it can be seen in Fig. 6(b) (Fig. 6(c)) that the maximum throughput is almost equal to the upper bound that can be calculated by (4), i.e., 0.5 (0.25).

With E[Fan-out] = 4, considerable performance gains can be achieved compared to the case of unicast traffic. While the delay values grow at high loads, the range of loads over which the switch has an acceptable performance can become much larger. The reason behind this behavior is two-fold: *i*) with a higher multicast degree, a larger number of output ports can simultaneously receive packets using a single wavelength, and *ii*) the diversity of the destinations increases at the HOL positions of the queues, counteracting the effects of HOL blocking. The former plays an important role when the number of wavelength resources is scarce (compare Figs. 6(c) and 6(f)) and the latter with large enough W. Comparing Figs. 6(b) and 6(e) (i.e., for $W = 32$ and $Q = 8$), the maximum throughput improves by 94% when the average fan-out is increased from 1 to 4.

D. Delay Performance under Bursty Traffic

Fig. 7 presents the simulation results under bursty traffic. These are more relevant to a data center network where temporal traffic and spatial correlations are common. Under bursty traffic, with $W = 64$ (Figs. 7(a) and 7(d)), both algorithms exhibit a similar performance. In fact with bursty traffic, the neighboring packets in queues have a high chance of bearing the same destination sets. Hence, avoiding fan-out splitting is not efficacious in reducing HOL blocking. Moreover, with $W = N$, there is no need for saving the resources by avoiding packet retransmissions.

As can be seen in Figs. 7(a) and 7(d), under bursty traffic and with $W = 64$, the performance of the GMQA and the MAMFS enhances by increasing Q. The maximum throughput for the GMQA (MAMFS) is 0.54 (0.54) with $Q = 1$ and 0.78 (0.8) with $Q = 8$, which translates to 44% (48%) improvement. At an effective load of 0.5, the average delay for the GMQA reduces from 143 time slots

with $Q = 1$ to 67 time slots with $Q = 2$, which corresponds to a 53% reduction. Increasing Q beyond 2 has a negligible impact on the average delay at this load.

Comparing Figs. 7(b) and 7(e) with Figs. 4(b) and 4(e), one can observe that increasing Q is more effective under bursty traffic. By assigning traffic flows to multiple queues as in Sec. II, the scheduling diversity is improved at the HOL position of input queues. This diversity of destinations is more significant for bursty traffic than for uniform traffic.

Finally, limiting the number of existing wavelengths to $W = 32$ under bursty traffic poses a negligible impact on the performance of the proposed algorithms (compare Figs. 7(b) and 7(e) to Figs. 7(a) and 7(d)). With $W = 16$ (Figs. 7(c) and 7(f)), the effects of using multiple queues is minimal. In this setting, the MAMFS approximately reaches the 0.5 upper bound on the maximum utilization. Moreover, by minimizing fan-out splitting, the MAMFS exploits the limited resources more effectively and outperforms the GMQA by 11%.

VI. Conclusion

We proposed round-robin-based scheduling algorithms for all-optical multicasting in coupler-based switches that are subject to transceiver tunability constraints and allow for exploiting multiple input queues per port. Our simulation results indicated that for scenarios in which the number of wavelengths is greater than or equal to one half of the switch port count, deploying multiple queues can significantly improve the delay performance. Moreover, we observed that the performance gains are more significant under bursty traffic in comparison to a uniform traffic pattern. Our analysis further suggested that avoiding fan-out splitting in combination with multiqueue buffering is an effective strategy for improving the performance of optical multicast switches, provided that spectral resources are not too scarce. In the future, we plan to develop scheduling algorithms that can handle the impact of spatial data center traffic correlations in addition to temporal correlations. Future work should also aim at developing multiqueue, multicast scheduling algorithms for multistage optical switch architectures.

References

[1] C. Kachris, K. Kanonakis, and I. Tomkos, "Optical interconnection networks in data centers: recent trends and future challenges," *IEEE Communications Magazine*, vol. 51, no. 9, pp. 39–45, Sep. 2013.

[2] H. Rastegarfar, L. A. Rusch, and A. Leon-Garcia, "Optical load-balancing tradeoffs in wavelength-routing cloud data centers," *Journal of Optical Communications and Networking*, vol. 7, no. 4, pp. 286–300, Apr. 2015.

[3] P. Samadi, V. Gupta, J. Xu, H. Wang, G. Zussman, and K. Bergman, "Optical multicast system for data center networks," *Optics Express*, vol. 23, no. 17, pp. 22 162–22 180, Aug. 2015.

[4] H. Wang, Y. Xia, K. Bergman, T. S. Ng, S. Sahu, and K. Sripanidkulchai, "Rethinking the physical layer of data center networks of the next decade: Using optics to enable efficient-cast connectivity," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 52–58, Jul. 2013.

[5] J. Chen, Y. Gong, M. Fiorani, and S. Aleksic, "Optical interconnects at the top of the rack for energy-efficient data centers," *IEEE Communications Magazine*, vol. 53, no. 8, pp. 140–148, Aug. 2015.

[6] D. Li, M. Xu, Y. Liu, X. Xie, Y. Cui, J. Wang, and G. Chen, "Reliable multicast in data center networks," *IEEE Transactions on Computers*, vol. 63, no. 8, pp. 2011–2024, Aug. 2014.

[7] Z. Guo, J. Duan, and Y. Yang, "On-line multicast scheduling with bounded congestion in fat-tree data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 102–115, Jan. 2014.

[8] W.-K. Jia, "A scalable multicast source routing architecture for data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 116–123, Jan. 2014.

[9] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "ESM: Efficient and scalable data center multicast routing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 944–955, Jun. 2012.

[10] F. Yan, W. Hu, W. Sun, W. Guo, Y. Jin, H. He, Y. Dong, and S. Xiao, "Nonblocking four-stage multicast network for multicast-capable optical cross connects," *Journal of Lightwave Technology*, vol. 27, no. 17, pp. 3923–3932, Sep. 2009.

[11] Y. Yang, J. Wang, and C. Qiao, "Nonblocking WDM multicast switching networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 12, pp. 1274–1287, Dec. 2000.

[12] W. Ni, C. Huang, Y. L. Liu, W. Li, K.-W. Leong, and J. Wu, "POXN: a new passive optical cross-connection network for low-cost power-efficient datacenters," *Journal of Lightwave Technology*, vol. 32, no. 8, pp. 1482–1500, Apr. 2014.

[13] M. Maier, M. Scheutzow, and M. Reisslein, "The arrayed-waveguide grating-based single-hop WDM network: an architecture for efficient multicasting," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 9, pp. 1414–1432, Nov. 2003.

[14] Q. Huang and W.-D. Zhong, "Wavelength-routed optical multicast packet switch with improved performance," *Journal of Lightwave Technology*, vol. 27, no. 24, pp. 5657–5664, Dec. 2009.

[15] Z. Guo and Y. Yang, "High-speed multicast scheduling in hybrid optical packet switches with guaranteed latency," *IEEE Transactions on Computers*, vol. 62, no. 10, pp. 1972–1987, Oct. 2013.

[16] H. Rastegarfar, L. Yan, K. Szczerba, and E. Agrell, "PAM performance analysis in multicast-enabled wavelength-routing data centers," *Journal of Lightwave Technology*, vol. 35, no. 13, pp. 2569–2579, Jul. 2017.

[17] K. Keykhosravi, H. Rastegarfar, and E. Agrell, "Multicast scheduling for optical data center switches with tunability constraints," in *Proc. IEEE International Conference on Computing, Networking and Communications (ICNC)*, Jan. 2017, pp. 308–312.

[18] S. Shimada, *Coherent Lightwave Communications Technology*. Springer Science & Business Media, 2012.

[19] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 855–866, Jun. 1997.

[20] S. Gupta and A. Aziz, "Multicast scheduling for switches with multiple input-queues," in *Proc. 10th IEEE Symposium on High Performance Interconnects*, Aug. 2002, pp. 28–33.

[21] D. Pan and Y. Yang, "FIFO-based multicast scheduling algorithm for virtual output queued packet switches," *IEEE Transactions on Computers*, vol. 54, no. 10, pp. 1283–1297, Oct. 2005.

[22] H. Yu, S. Ruepp, and M. S. Berger, "Multi-level round-robin multicast scheduling with look-ahead mechanism," in *Proc. IEEE International Conference on Communications (ICC)*, Jun. 2011.

[23] E. Schiattarella and C. Minkenberg, "Fair integrated scheduling of unicast and multicast traffic in an input-queued switch," in *IEEE International Conference on Communications (ICC)*, Jun. 2006, pp. 287–292.

[24] G. N. Rouskas and M. H. Ammar, "Multidestination communication over tunable-receiver single-hop WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 501–511, Apr. 1997.

[25] K. Naik, D. S. Wei, D. Krizanc, and S.-Y. Kuo, "A reservation-based multicast protocol for WDM optical star networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 9, pp. 1670–1680, Nov. 2004.

[26] E. Modiano, "Random algorithms for scheduling multicast traffic in WDM broadcast-and-select networks," *IEEE/ACM*

Transactions on Networking, vol. 7, no. 3, pp. 425–434, Jun. 1999.

[27] H.-C. Lin and C.-H. Wang, "A hybrid multicast scheduling algorithm for single-hop WDM networks," in *20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Apr. 2001, pp. 169–178.

[28] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications*, vol. COM-35, no. 12, pp. 1347–1356, Dec. 1987.

[29] M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1587–1597, Dec. 1988.

[30] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," in *Proc. ACM Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Aug. 2002, pp. 251–264.

[31] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet routers using optics," in *Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, Aug. 2003, pp. 189–200.

[32] M. Andrews, S. Khanna, and K. Kumaran, "Integrated scheduling of unicast and multicast traffic in an input-queued switch," in *Proc. 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3, Mar. 1999, pp. 1144–1151.

[33] L. Kleinrock, *Queuing systems, vol. 1: Theory.* New York, NY: J. Wiley and Sons, 1975.