Sirius: A Flat Datacenter Network with Nanosecond Optical Switching

Hitesh Ballani Paolo Costa Raphael Behrendt Daniel Cletheroe Istvan Haller
Krzysztof Jozwik Fotini Karinou Sophie Lange Kai Shi Benn Thomsen Hugh Williams
Microsoft Research

ABSTRACT

The increasing gap between the growth of datacenter traffic and electrical switch capacity is expected to worsen due to the slowdown of Moore's law, motivating the need for a new switching technology for the post-Moore's law era that can meet the increasingly stringent requirements of hardware-driven cloud workloads. We propose Sirius, an optically-switched network for datacenters providing the abstraction of a single, high-radix switch that can connect thousands of nodes—racks or servers—in a datacenter while achieving nanosecond-granularity reconfiguration. At its core, Sirius uses a combination of tunable lasers and simple, passive gratings that route light based on its wavelength. Sirius' switching technology and topology is tightly codesigned with its routing and scheduling and with novel congestion-control and time-synchronization mechanisms to achieve a scalable yet flat network that can offer high bandwidth and very low end-to-end latency. Through a small-scale prototype using a custom tunable laser chip that can tune in less than 912 ps, we demonstrate 3.84 ns end-to-end reconfiguration atop 50 Gbps channels. Through large-scale simulations, we show that Sirius can approximate the performance of an ideal, electrically-switched non-blocking network with up to 74-77% lower power.

CCS CONCEPTS

• **Networks** → **Data center networks**; **Network protocols**; **Network architectures**.

KEYWORDS

Optical Switches, Datacenter Networks, Fast Tunable Lasers, Nanosecond Switching, Vertical Integration, Scheduler-less design

ACM Reference Format:
Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. 2020. Sirius: A Flat Datacenter Network with Nanosecond Optical Switching. In *Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM '20), August 10–14, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3387514.3406221

SIGCOMM '20, August 10–14, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7955-7/20/08…$15.00
https://doi.org/10.1145/3387514.3406221

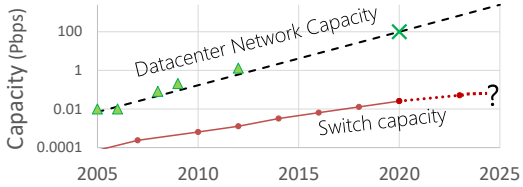


Figure 1: Datacenter network capacity (and traffic) [70] is growing faster that the doubling of switch capacity every two years (which, in turn, is expected to slowdown beyond 2024).

1 INTRODUCTION

The Moore's law for networking–electrical switches doubling their bandwidth every two years at the same power and cost–has allowed datacenter operators to scale up their network across generations. However, this impressive free scaling of switches still lags behind the doubling of cloud traffic roughly every year [70]. The increasing gap, shown in Fig. 1, results in high power, cost, and latency. For example, a large datacenter today would ideally be equipped with a 100 Pbps non-blocking network which would consume a prohibitive 48.7 MW, more than the 32 MW allocation for an entire datacenter.

Looking ahead, the problem is expected to worsen due to emerging cloud workloads that, unlike today's software-driven workloads, are hardware-driven. Examples include distributed training for deep neural networks (DNNs) atop accelerators like GPUs and TPUs [30], and disaggregation of resources like GPUs, FPGAs and non-volatile memory. The network thus needs to operate at hardware speeds, providing even higher bandwidth and ultra-low latency. Worse yet, even the free scaling of CMOS-based electrical switches is expected to taper-off due to the slowdown of Moore's law [5, 52, 64]. So the cost and power of electrically-switched networks could worsen every generation beyond the next four years.

We present Sirius, an all-optical datacenter network that provides the abstraction of a single high-radix switch connecting thousands of nodes with high bandwidth and with end-to-end reconfiguration at nanosecond-timescales. Instead of a hierarchy of electrical switches, Sirius uses a single layer of gratings, a very simple and passive building block with no moving parts and no power. Each grating diffracts or routes incoming light to an output port based on its wavelength. It can thus be used as a physical-layer switch by equipping a node's transceivers with tunable lasers that can change the wavelength used to carry the data, effectively using the wavelength as a proxy for the destination address. The nodes can be servers or rack switches. With a server-based deployment, server uplinks use the tunable transceivers to encode data onto lightwaves while with a rack-based deployment, servers connect to electrical rack switches whose uplinks use the tunable transceivers.

A key aspect of Sirius' design is the drive for very fast reconfiguration. This is motivated by the bursty nature and high fan-out of emerging cloud workloads that necessitate fast switching (less than 10 ns, §2.2). The speed of physical-layer reconfiguration in Sirius is dictated by the tuning latency of the laser. Since off-the-shelf tunable lasers take milliseconds to change wavelengths, we design and fabricate a custom tunable laser chip and demonstrate a tuning time of less than 912 ps, paving the way for nanosecond optical switching. Another key challenge is the lack of buffering inside the optical network, which requires scheduling traffic to avoid in-network contention. Since using an explicit scheduler to collect demands and compute schedules at nanosecond timescales and at datacenter scale would impose high overhead, if at all practical, Sirius uses a "scheduler-less" design with a static schedule that connects nodes to each other in a round-robin fashion. To accommodate any traffic pattern atop this static schedule, Sirius uses *load-balanced routing* [12, 72] whereby traffic from each node is detoured uniformly through other nodes on its way to the destination.

Sirius' design has the following advantages. First, its topology is "flat". By eliminating expensive and power-hungry switches and transceivers in the core network, it provides high-bandwidth at significantly lower power and cost (§5). With a rack-based deployment, it can connect up to 25,600 racks–6× the size of a large datacenter today. Second, by allowing reconfiguration at nanosecond-timescales, it emulates the packet-by-packet switching offered by today's electrically-switched networks (§6), thus showing the viability of supporting a broad set of workloads with an all-optical core. In contrast, previous optically-switched architectures [14, 25, 34, 43, 49, 57, 74] typically rely on a separate electrically-switched networks for latency-sensitive workloads due to the micro-to-millisecond granularity of switching. Third, by eliminating buffers inside the optical network and by careful management of the buffering at the nodes themselves, it achieves very low and predictable latency (§7). Finally, the network's core is passive with no dependency on CMOS components, which makes it robust (§4.5) and future proof as it does not need to be upgraded across generations.

Beyond ultra-fast laser tuning and scheduling, achieving these properties also required solving several systems challenges. For example, end-to-end reconfiguration at nanosecond granularity relies on very accurate time synchronization across the nodes and accounting for the varying physical delays between them. A key theme that allowed us to tackle these challenges is the tight codesign of mechanisms across the entire cloud network stack, starting from the choice of the switching technology and topology to the mechanisms used for scheduling, synchronization, clock recovery, and congestion control. For example, the deterministic latency provided by the passive core and the periodic connection of nodes by the schedule underlies the design of the decentralized time-synchronization mechanism that achieves an accuracy of ± 5 picoseconds and its congestion control algorithm that ensures very low queuing.

Overall, this paper makes the following contributions:

- We propose Sirius, a scalable and flat optically-switched network for datacenters that achieves nanosecond-granularity reconfiguration in a cost- and power-efficient fashion.
- We present a novel design for tunable lasers that uses disaggregation to achieve sub-nanosecond laser tuning and implement it atop a custom photonics integrated circuit (PIC).
- We codesign and implement a few novel technologies necessary for end-to-end reconfiguration at nanosecond-granularity, including mechanisms for fine-grained time synchronization and clock-and-data recovery.
- We codesign and implement a new congestion control protocol that leverages Sirius' features to ensure very low queuing and hence, very low tail latency.
- We demonstrate a small-scale Sirius prototype using off-the-shelf tunable lasers to connect 4 FPGA-based nodes with an end-to-end reconfiguration latency of 100 ns. We further improve it with our custom laser, achieving an end-to-end latency of 3.84 ns.
- Through large-scale simulations, we demonstrate that Sirius can closely match the performance of an ideal non-blocking network with up to 74-77% lower power.

[This work does not raise any ethical issues.]

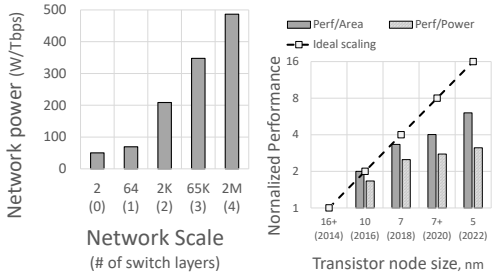
2 MOTIVATION

Datacenter networks today use Clos-based, hierarchical topologies built using low-port-count or low-radix electrical switches. This allows them to scale-up to arbitrary sizes simply by adding more network layers. However, the use of hierarchy imposes a "scale tax", i.e., an increase in network power, cost and latency as the network scales up. Fig. 2a shows how the network power per unit bisection bandwidth increases as it is scaled in size by adding layers of hierarchy: connecting two nodes directly with an optical transceiver plus fiber consumes only 50 Watts/Tbps. However, assuming 64-port switches (400 Gbps per port) [8], connecting more than 65 K nodes in a large datacenter would require four layer of switches, with the additional switches and transceivers adding up to 487 Watts/Tbps. Extrapolating from historical data on intra-datacenter traffic, such a datacenter would ideally require 100 Pbps of bisection bandwidth today (see Fig. 1). However, the power for such a network is a prohibitive 48.7 MW (487 Watts/Tbps x 100 Pbps). The cost and latency of the network are similarly hurt as network layers are added.

2.1 Future trends indicate a perfect storm

Operators have traditionally coped with the scale tax by oversubscribing the network to cap its power and cost. This comes at a price though: it increases fragmentation of resources like network-attached storage and worsens the tail latency further from tens to hundreds of microseconds [32]. Till now, this has been an acceptable trade-off as workloads today are software-driven and software stack latencies are much higher than network latencies. However, this is primed to change with workloads like distributed DNN training and disaggregation of hardware resources that are expected to drive future datacenter growth. In these workloads, traffic is generated and consumed by hardware directly and hence, is expected to grow even faster than the doubling every year. For e.g., while CPUs struggle to saturate 100 Gbps links, state-of-the-art GPUs can process 2.4 Tbps of network traffic [54]. Furthermore, by eliminating the stack latency, it makes network latency the next bottleneck for application performance. This motivates the need for ultra-low network latency.

Apart from the downward pressure due to new workloads, there is also upward pressure as we approach the limits of free scaling of



(a) Power/bisection b/w **(b) CMOS Scaling**

Figure 2: (a) Scale tax—as the network is scaled by adding hierarchy, the total power worsens significantly. (b) Slowdown of CMOS scaling is expected to hurt network power (and cost).

CMOS-based electrical switches. This is because of two key metrics whose increase is becoming hard to sustain: i) the I/O bandwidth off the switching ASIC due to limits on SERDES density and speed, and ii) the transistor density on the ASIC itself. Fig. 2b shows that, as the CMOS node size reduces below 7nm, the power and area gains are far from the historic doubling every generation. For SERDESes, the problem is worse as they have a significant analog component which scales even more poorly, which, in turn, impacts transceiver scaling too. As a result, the cost and power of switches and transceivers beyond two generations is unlikely to stay constant. Higher bandwidth switches could still be constructed through a hierarchical collection of smaller ASICs but at the expense of power and cost. Thus, the scaling of electrically-switched networks would worsen every generation thereafter.

2.2 Why Fast Optical Switches?

The shortcomings of electrically-switched networks, coupled with a poor prognosis on scaling, has driven the development of several optical switching technologies. MEMS-based optical switches have been particularly popular in optical architectures for datacenters. However, the reconfiguration latency of these switches, varying from milliseconds [10] to tens of microseconds [29], is much slower than the packet-granularity reconfiguration offered by electrical packet switches to accommodate small network flows that may only last hundreds of nanoseconds. Looking ahead, datacenter traffic patterns are changing with scenarios like key-value stores and memory disaggregation resulting in *very bursty* workloads, i.e., most of the bytes are in short transfers. We analyzed network traces from a production cloud service across two days in Mar 2019 and found the workload to be dominated by small packets [20]. Over 34% of the packets comprise less than 128 bytes while 97.8% of the packets are 576 bytes or less. Similarly, over 91% of the packets generated by Facebook's in-memory cache are 576 bytes or less [80]. Furthermore, these workloads have a *high fanout* with an endpoint communicating with many destinations at the same time, particularly at high load.

Supporting these workloads–with high burstiness and fan out–while maintaining high network utilization requires much faster reconfiguration. For example, an endpoint sending 576 B packets to different destinations would be ideally served by switching between the destinations every 92 ns (with 50 Gb/s optical channels in today's 400 Gb/s ports). Thus, to ensure less than 10% switching overhead, the reconfiguration period, during which no data can be transferred, should be shorter than 9.2 ns. We further conducted simulations to understand the impact of reconfiguration latency on flow completion time (FCT) and found that, at high load, the FCT grows sharply beyond a reconfiguration latency of 10 ns (Fig. 11). Thus, we target an end-to-end reconfiguration latency of less than 10 ns to achieve performance comparable to an ideal network.

3 BUILDING BLOCK TECHNOLOGIES

We begin with a description of the two main optical technologies that we use as building blocks in Sirius.

3.1 Wavelength gratings (AWGR)

Sirius' core comprises gratings called Arrayed Wavelength Grating Routers (AWGR), a passive optical component with many input and output ports. As shown in Fig. 3a, wavelengths from a given input port are diffracted or "routed" cyclically across the output ports. AWGRs are based on the same technology as optical components (arrayed waveguide gratings) that are commonly used in wide-area networks for multiplexing and demultiplexing of multiple wavelengths onto fiber.

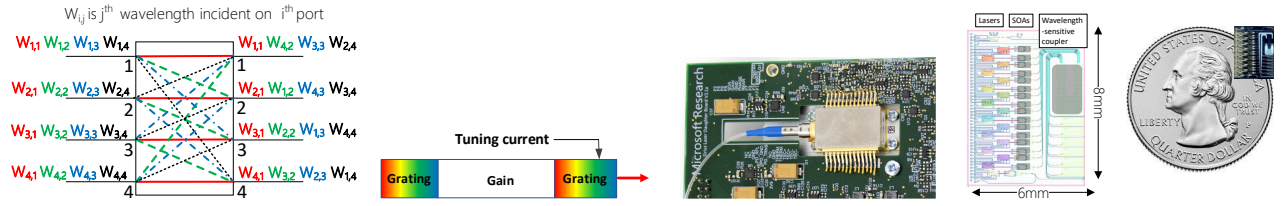
Apart from providing all-to-all connectivity between their ports at the physical layer, a big advantage of AWGRs is their passive nature with no mechanical or electrical parts and, hence, no power consumption. This also makes them very robust. Furthermore, the routing of a wavelength is agnostic to the modulation format used to encode the data onto it. This means, unlike electrical switches, AWGRs need not be upgraded every network generation. Over the past decade, significant advances have been made in chip-scale AWGRs promising low cost fabrication at high volume. AWGRs with around hundred ports are available commercially [53] while prototypes with 512 ports have been demonstrated [19].

3.2 Tunable lasers

A standard semiconductor laser comprises a gain section that generates a range of wavelengths and a grating section that allows one of these wavelengths to exit the laser. In a tunable laser (Fig. 3b), the grating section can be tuned using thermal or electrical input to change the output wavelength. With an electrically-tuned laser, a tuning current applied to the grating section changes the resonant wavelength it outputs. Commercially available electrically-tuned lasers can tune across ∼100 wavelengths in the optical C-band (with 50GHz spacing around 1550 nm) and are typically used in wide-area networks for flexible wavelength multiplexing [22].

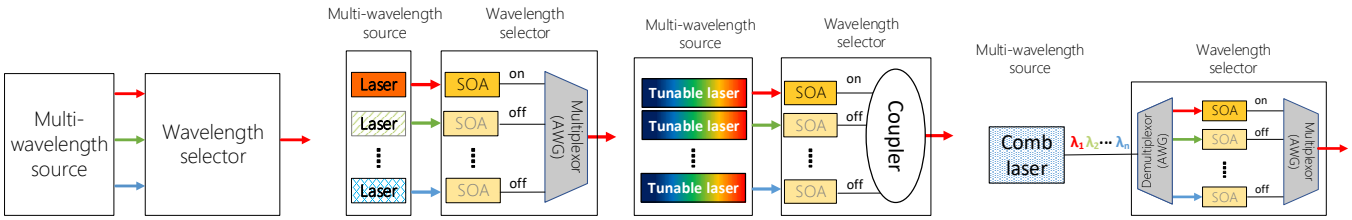
Tunable lasers, coupled with AWGR gratings, can operate as an all-optical switch that can route light from any input to any output. However, the latency of switch reconfiguration is dictated by the time to change the laser's wavelength. For off-the-shelf tunable lasers, the tuning latency can be a few milliseconds, mostly because their electrical drive circuitry is not designed for fast tuning. For e.g., our prototype uses DSDBR tunable lasers that can tune across 112 wavelengths with a tuning latency of 10 ms [51].

The key challenge with fast tuning is that the injection of current to the grating section for tuning the laser perturbs the gain section that is actually generating the light. Specifically, each output wavelength λ_i is associated with a tuning current I_i. However, when tuning from λ_i to λ_j, simply changing the current from I_i to



(a) Arrayed waveguide grating router (AWGR) (b) Standard tunable laser (c) Tunable laser on custom board (d) Fast tunable laser chip

Figure 3: (a) AWGR achieves wavelength-based routing, (b) Standard tunable laser, (c) Custom PCB for fast tuning of standard tunable lasers, (d) On-chip implementation of the disaggregated tunable laser (*fixed laser bank*).



(a) Disaggregated tunable laser (b) Fixed laser bank (c) Tunable laser bank (d) Comb laser

Figure 4: Disaggregated tunable laser and three design instantiations that we implemented.

I_j leads to a "ringing effect" whereby the laser's output oscillates across wavelengths adjacent to the destination wavelength before settling. To reduce these oscillations, we implement a dampening technique–instead of changing the tuning current in a single step, we apply it in a series of steps: intentionally overshooting and then undershooting the destination current before finally settling on it [26]. We implemented this technique atop a custom PCB board for laser driving (Fig. 3c) to achieve a median tuning latency of 14 ns and worst-case latency of 92 ns across all 12,432 pairs of wavelengths.

3.3 Disaggregated tunable laser

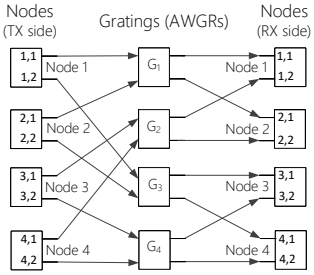
While the dampening technique above significantly reduces tuning latency, it still does not meet our target of reconfiguration within 10 ns. The reason is that the tight coupling between wavelength generation and wavelength selection in standard tunable lasers imposes a fundamental limit on how fast they can be reconfigured. In particular, the farther are the source and destination wavelength, the higher is the change in tuning current and longer is the settling time. To reduce the tuning latency and make it independent of the wavelength span, we propose a new tunable laser design whereby the light generation is disaggregated from selection of the wavelength. Fig. 4a shows the two components of the resulting laser: i). a *multi-wavelength source* that can generate multiple wavelengths simultaneously, and ii). a *wavelength selector* that can be tuned to emit only one of these wavelengths. Apart from alleviating the perturbation problem, such disaggregation also offers more fundamental gains since the technology and even the material (Silicon, Indium Phosphide, etc.) for implementing these two components can be chosen and optimized independently. We have implemented the following three instantiations of this disaggregated design [40, 68]:

1. Fixed laser bank. This design variant, shown in Fig. 4b, uses a bank of single wavelength lasers as the multi-wavelength light source [68]. The wavelength selection can be done using an array of semiconductor optical amplifiers (SOAs) that act as optical gates, i.e., they either let light through or block it based on the applied current. When the chip is tuned to wavelength λ_i, SOA_i is turned on while all other SOAs are turned off. The use of SOAs for wavelength selection (physically separated from the light generation) offers the advantage that they can be turned on and off in nanosecond timescales. The trade-off, however, is that they consume extra power and generate optical noise. However, we note that only one SOA is on at any instant. Furthermore, the SOA only amplifies unmodulated light which alleviates the impact of any optical noise. Other technologies that can serve as fast optical gate, such as Mach Zehnder Interferometers (MZI) [41] or ring resonators [16], could be used for wavelength selection too.

The advantage of this design is its simplicity, both in terms of the lasers and the drive electronics. The disadvantage is its scalability since the number of wavelengths is limited by the number of lasers, which, in turn, increase the power, complexity and cost of the source chip. In §4.5, we discuss how the Sirius architecture allows the disaggregated laser to be shared across optical channels in order to amortize the overhead of this design.

2. Tunable laser bank. This design variant uses a smaller bank of tunable lasers, as shown in Fig. 4c, that operate in a pipelined fashion [28, 66]. Hence, if the fact that the output wavelength needs to be tuned λ_i to λ_j is known in advance, the first tunable laser could be emitting λ_i while a second tunable laser is tuning to λ_j concurrently to hide its tuning latency. This bank of lasers is still connected to the SOA-based wavelength selector that switches from the first to the second laser once its tuning is complete. This concept can be generalized to more tunable lasers if the future sequence of wavelength transitions (not just the next one) is known in advance.

The advantage of this design is that it requires fewer lasers, albeit tunable, and is particularly amenable to redundancy using spare tunable lasers. In §4.5, we show that a bank of three tunable lasers, including a spare laser for fault-tolerance, is sufficient for Sirius'



(a) Topology.

Source	Timeslot 1		Timeslot 2	
(Node, Port)	λ	Dst	λ	Dst
(1, 1)	A	(1, 1)	B	(2, 1)
(1, 2)	A	(3, 1)	B	(4, 1)
(2, 1)	A	(2, 1)	B	(1, 1)
(2, 2)	A	(4, 1)	B	(3, 1)
(3, 1)	A	(1, 2)	B	(2, 2)
(3, 2)	A	(3, 2)	B	(4, 2)
(4, 1)	A	(2, 2)	B	(1, 2)
(4, 2)	A	(4, 2)	B	(3, 2)

(b) Network schedule.

Figure 5: Four-node Sirius topology with two uplinks each (equipped with lasers that tune across two wavelengths) and a layer of gratings. The network has a static schedule that repeats every epoch, i.e., two timeslots.

goals. The disadvantage is that the control and packaging of the tunable lasers is more complicated and the wavelength transitions need to be known in advance. Finally, since each tunable laser could be generating different wavelengths over time, the wavelength selection chip uses an optical coupler to combine the outputs of the lasers which introduces higher optical insertion loss than the wavelength multiplexor used in the previous fixed laser design.

3. Comb laser. Frequency combs are lasers that generate discrete, equally spaced wavelengths and, hence, are a natural fit for the multi-wavelength source in our design. Recent advances have demonstrated chip-based comb lasers that can generate more than a hundred wavelengths [46] and could be coupled with an SOA-based wavelength selector to form a fast tuning source (Fig. 4d) [40]. The advantage of such devices is that the light source is a scalable, single chip and equal spacing between the many wavelengths is always maintained without the need for temperature control. While the power consumption of this design variant with today's combs is higher than the other two, comb lasers are expected to improve significantly over the coming years and could be a promising alternative in future.

In this paper, we focus on the first design that we fabricated on a custom optical chip. The chip, made in Indium Phosphide and shown in Fig. 3d, is 6 mm x 8 mm and achieves a tuning latency of less than 912 ps (§6).

4 SIRIUS ARCHITECTURE

Sirius uses a combination of tunable lasers and gratings to provide the abstraction of a single, high-radix optical switch. It could be used to connect thousands of servers directly to form a non-CMOS-based, high-performance network cluster, or to scale to the entire datacenter by connecting all rack switches through a flat and bufferless core. We adopt the term *node* to generically refer to the endpoints attached to the optical network (either servers or racks) and use the term "server" or "rack" when discussing the specific deployment.

4.1 Physical topology

In Sirius, nodes are connected to the fully passive core network (comprising a single layer of AWGRs or gratings) through a number of uplinks. Each uplink port is equipped with a transceiver containing a tunable laser and connected to a grating through an optical fiber. This allows the node to send data to all other nodes connected to the outputs of that grating by changing the wavelength of its laser.

Fig. 5a shows a small-scale Sirius topology with four nodes (two uplinks each) and four gratings (two ports each). Each uplink port on a node is connected to a different grating and through it, can send traffic to a different set of destination nodes. So, node 1 can reach nodes 1 and 2 through grating $G1$ and nodes 3 and 4 through grating $G3$, by using the two wavelengths (A and B). Gratings with 100 ports and lasers that can tune across W=100 wavelengths are commercially available so each node uplink can reach W other nodes through the corresponding grating. Modern accelerator-driven servers can consume bandwidth in excess of 2.4 Tbps [54], internally comprising 48×50 Gbps channels. By connecting these uplink channels to different 100-port gratings, such a network could connect 4,800 servers (48×100), serving as a large cluster for high-performance scenarios like distributed DNN training or memory disaggregation. A rack-based deployment with the latest rack switches that have 512 SERDESes [8] (i.e., 256 uplinks) would instead allow upto 25,600 (100×256) racks to be connected. A large datacenter with 4,096 racks could thus be connected through just 16-port gratings.

We note that while Sirius' topology is flat with no opto-electronic conversion for data transmitted between a pair of nodes, by itself the topology provides direct connectivity between any pairs of nodes through only one of their uplink ports. So, with simple direct routing, the nodes would only be able to communicate directly with a fraction of their total uplink bandwidth. We address this through the Sirius' routing mechanism described next.

4.2 Routing and scheduling

A big advantage of Sirius' design is the lack of any queuing or packet processing inside the core network with all queuing pushed to the nodes themselves where it can be better managed. However, the lack of queuing also makes it necessary to schedule traffic so that no uplink port on a node receives traffic from multiple source nodes simultaneously. One approach is on-demand scheduling, i.e., sending the datacenter demand matrix, measured at the node, to a scheduler that calculates and assigns communication timeslots to the nodes. This requires measuring demands, calculating assignments and maintaining a robust control plane for communicating demands and assignments. While such an approach may be viable when optical switching is done at coarse timescales, it is not efficient and practical for Sirius' fast switching at scale.

Instead, Sirius adopts a scheduler-less design, proposed by Chang et al. [12] as an extension of Valiant load balancing [72]. Traffic from a node, irrespective of its destination, is routed uniformly on a packet-by-packet basis across all other nodes, which then forward the traffic to its destination node. Such detouring converts the datacenter's demand matrix into a uniform demand matrix, i.e., any given node has the same amount of traffic to send to all other nodes in the datacenter. This, in turn, results in the key property whereby Sirius only needs to provide equal-rate connectivity between all nodes—in a datacenter with N nodes with an aggregate uplink bandwidth B per node, each node pair only needs to be connected with $\frac{B}{N}$ bandwidth, which is a perfect match for Sirius' topology.

To achieve this, the nodes are time synchronized (§4.4) and follow a pre-determined, static schedule that specifies the connectivity at any given fixed-size timeslot. Each transceiver on a node is tuned across all wavelengths cyclically on a timeslot-by-timeslot basis and,

hence, can send to all the nodes at the output of the grating that the transceiver is physically connected to, in a round-robin fashion. Furthermore, all the transceivers on a node are connected to different gratings and, hence, to a different set of destination nodes. Taken together, a node is connected to all other nodes in the datacenter cyclically and with equal bandwidth. Fig. 5b shows the schedule for our example network: all nodes cyclically alternate between wavelengths A and B so that each node communicates to every other node once every two timeslots. Broadly, with G-port gratings, the network schedule repeats every "*epoch*" comprising G timeslots. Furthermore, the schedule is contention-free, i.e., no destination port has two sources sending traffic to it at the same timeslot.

Packet forwarding. Sirius uses fixed-sized packets or "cells" that are generated by the source server. In a server-based deployment the traffic generated locally on a server is directly forwarded to the intermediate servers. Cells received from other servers, instead, are consumed locally (if they reach the final destination) or forwarded to the final destination on the corresponding slot. Conversely, in a rack-based deployment, intra-rack traffic is forwarded directly through the rack switch. Traffic destined outside the rack is stored in a separate FIFO virtual queue and forwarded to the intermediate racks. Cells arriving at the rack switch from its uplinks are routed based on their destination, either to the server within the rack or directly to the destination rack. This ensures that traffic is detoured through at most one intermediate node. Further, unlike a traditional datacenter with a few paths between a pair of nodes, with Sirius, a node can reach any destination node through all other nodes.

Throughput and latency impact. While Sirius' cyclic schedule obviates the need for a datacenter-wide scheduler, it does mean that traffic between any pair of nodes needs to take an extra hop through an intermediate node. This can adversely impact both the throughput and the latency of the network. However, a big advantage of the load balancing technique is that it guarantees the worst-case throughput across any traffic pattern—it can be at most $2\times$ worse than that of an ideal, non-blocking network [12]. Practically, the throughput impact can be negated by doubling the number of uplink transceivers on the nodes. The fact that Sirius has a flat topology with no transceivers inside the network core means that the total number of transceivers in Sirius is still much lower than in a hierarchical, electrically-switched network, resulting in lower power and cost (§5).

On the latency front, there are two sources of extra latency due to the detouring. First, the extra physical distance traversed by traffic. In a large datacenter with a span of 500 m, the detouring of traffic through an intermediate node can add an extra physical distance of up to 500 m which translates to a maximum 2.5 μs of extra propagation latency. The second source of latency is queuing at the intermediate node. With 100 ns timeslots (576 B cells plus overhead) and 16 nodes connected to each grating, the network epoch is 1.6 μs ($16 \times$ 100 ns). So, a cell can incur a 1.6 μs of queuing delay for every other cell queued in front. Thus, to ensure low end-to-end network latency, it is crucial to minimize the amount of queuing at intermediate nodes.

Cell reordering.. Cells for a given flow take different paths through the network and, hence, can arrive out of order at the destination. However, as we show in §7, due to the low queuing ensured by the congestion control (§4.3), only a small reordering buffer is sufficient.

4.3 Congestion Control

Sirius' congestion control protocol aims to minimize the queuing at the nodes. Queuing at an intermediate node I occurs when, during the same epoch, two or more nodes send a cell to I for the same destination node D. Since I can only send one cell to D each epoch, subsequent cells have to wait one or more epochs before they can be transmitted. If this keeps occurring, queues can grow very large.

To address this, we designed a congestion-control protocol that requires a round of request/grants before cells can be transmitted to an intermediate node. The goal is to ensure that a source node is allowed to send a cell destined to D via the intermediate node I only if the latter has at most $Q-1$ cells already queued for D. The protocol works as follows. Each node is equipped with a buffer LOCAL containing the cells generated locally (in a server-based deployment) or received through one of the downlinks (in a rack-based deployment). At the start of each epoch, each node inspects LOCAL and for each queued cell, it randomly selects an intermediate node to which it will send a request asking for permission to forward the cell to. This process terminates when either one request has been generated for all cells in LOCAL or all intermediate nodes have been selected (to reduce overhead and protocol complexity, we only allow to send one request to the same intermediate node per epoch). In parallel, each node will also randomly select a request per each destination D among the ones received in the previous epoch and will issue a grant as long as the sum of the packets queued for D and the number of outstanding grants for D is lower than the queue threshold Q. Finally, when a grant for D from I is received, the node will move one cell for D from LOCAL into a virtual queue for I to be transmitted in the next epoch. The protocol is similar to a distributed version of DRRM [13], which has been shown to achieve 100% throughput for hot-spot traffic and is amenable to a simple and fast hardware implementation [78]. We report the complete pseudocode in Appendix (Fig. 15).

The value Q defines the upper bound on the maximum amount of queuing that cells will experience. The minimum value of Q is 2 because, depending on the network schedule, within the same epoch a node might receive a new cell for D before it has a chance to transmit the one received in the previous epoch. In practice, however, as we show in §7, we find that $Q=4$ provides higher throughput in case of bursty traffic without impacting latency for short flows. The protocol takes advantage of the fact that during each epoch, a node sends a cell to *all* other nodes. This enables piggybacking requests and grants on each cell, thus minimizing the overhead. On the downside, however, this will introduce an initial epoch-length worth of latency for each flow. We consider this an acceptable trade-off in exchange of bounded latency and low memory utilization. Further, by avoiding queuing overflow, we make the core of the network *lossless*, thus requiring re-transmission only in the rare case of packet corruption.

In a server-based deployment, the protocol discussed above could be implemented on the server NICs without requiring any additional mechanism. Conversely, in a rack-based deployment, a separate protocol is needed to rate-limit the traffic between servers and racks to avoid overloading the LOCAL buffer. However, since the protocol above eliminates congestion in the network core, a simple one-hop flow control mechanism between the source server and its rack

switch and between the destination server and its rack switch is required. This could be achieved using solutions prevalent in HPC networks, e.g., the InfiniBand credit-based link-layer protocol [47].

4.4 Time Synchronization

Apart from the challenge of switch reconfiguration and scheduling at nanosecond-timescales, another key barrier to fast optical switching is that it necessitates fine-grained time synchronization—for nanosecond switching, nodes need to be synchronized with an accuracy of less than 100 picoseconds. While we considered several existing synchronization protocols [27, 42, 63], they either do not offer sufficient accuracy or introduce additional complexity and require specialized hardware, impacting overall costs. In contrast, by leveraging the Sirius' passive core and cyclic schedule, we design a novel time synchronization protocol that incurs only minimal complexity. We provide a high-level sketch below.

The fact that the gratings in the core (§3.1) are completely passive provides two significant advantages. First, unlike other optical switches, they do not require any synchronization and, second, they do not perform any data retiming. This allows receivers to easily extract the clock of the sender from the incoming bit stream and adjust its local clock to match the frequency of the sender's clock, using a standard phase locked loop (PLL) [35] or a delay locked loop (DLL). Since each node pair is connected once every epoch, a naive implementation could be to simply designate a leader node and let all other nodes adjust their clock based on the one extracted from the cell received from the leader every epoch. For higher robustness, in Sirius we automatically switch the leader every few epochs in a round-robin fashion. This means that even if a node fails during its turn as a leader, it will be automatically replaced in few microseconds, which is sufficient to prevent any noticeable clock drift [27]. Notably, this solution does not pose any strict requirements on the quality of the clocks, i.e., no atomic clocks are necessary. Even if the clocks drift over time it does not matter as long as they remain synchronized among each other. Further, if a DLL is used, it is also possible to digitally filter too large frequency variations, thus partially addressing the case of byzantine clock failures.

The protocol described above ensures that all clocks tick at the same frequency. However, to achieve proper *time* synchronization, we also need to correctly estimate the propagation delay between any pairs of nodes and their relative time offset. We detail this technique in §A.2: the main idea is that the passive core makes it possible to accurately and efficiently measure a node's physical distance to the AWGR. This is used to derive the relative time offset between nodes. The knowledge of the physical distance of each node from the AWGR is also critical to ensure a correct behavior of the system when the fibers connecting nodes to the AWGR have different lengths. The intuition is that each node starts its first epoch at a different time depending on the distance from the AWGR: the longer this distance is, the sooner it will start so that the different distances are factored out and the packets belonging to the same slot will arrive at the AWGR at the same time.

4.5 Design discussion

We now discuss a few other aspects of Sirius' design.

CMOS dependency. A server-based Sirius deployment enables an all-optical, non-CMOS-based network interconnecting up to a few thousand servers. With rack-based Sirius, instead, there is still a dependency on CMOS-based electrical switches for intra-rack connectivity. By eliminating electrical switches and transceivers from the core network, however, Sirius is less impacted by a slowdown in Moore's law—if electrical switches stop scaling for free, the rate of increase of the overall network power and cost will be slower with Sirius as compared to the status quo. In such a post-Moore's law world, datacenter operators may even have to resort to increasing the levels of hierarchy for traditional networks which further increases Sirius' relative gains. A more efficient albeit invasive alternative for datacenter operators to continue scaling network bandwidth would be to build parallel networks [50]. Sirius' design is particularly amenable to such scaling through topology-level parallelism.

End-to-end reconfiguration latency. Timeslots are separated by a "*guardband*" when no application data is transmitted to allow the end-to-end path to be reconfigured. Beyond laser tuning and time synchronization inaccuracy, this also needs to account for the latency resulting from the ephemeral nature of the connection between nodes. Specifically, whenever two nodes are interconnected at a timeslot, the fact they are not perfectly synchronous means that their clock and data recovery (CDR) circuitry needs to be trained so that they can sample the incoming bitstream correctly. The CDR latency for standard transceivers is microseconds which has been a big impediment to fast optical switching at the system-level. We designed a novel technique called phase caching that reduces the CDR latency to less than a nanosecond [20, 21]. The main idea is to "cache" such system parameters instead of learning them from scratch, which, in turn, minimizes the latency for clock recovery (details in §A.1). Similarly, to equalize the varying optical power a node receives from different sources, we use "amplitude caching" instead of slower gain control circuitry. Sirius' cyclic schedule makes these techniques particularly appealing as every pair of endpoints is interconnected periodically, thus allowing the cached parameters to be updated without extra overhead. Overall, we show that our prototype achieves error-free operation with an end-to-end reconfiguration latency of 3.84 ns (§6), below our 10 ns target and allowing for a slot as low as 38 ns. We also note that the fact that SERDES rates are fast plateauing means that this reconfiguration latency target does not need to be reduced every network generation [4, 55].

Laser sharing. Most optical switching technologies introduce "insertion loss", i.e., reduction in light power as it traverses the switch. It is thus critical that the laser generates sufficient optical power to meet the link budget, i.e., despite the insertion loss along the lightpath, destinations receive sufficient signal for error-free operation. In our testbed, the receiver requires -8 dBm (0.16 mW) of incoming light to achieve error-free operation with standard FEC. 100-port gratings can be fabricated with a maximum 6 dB insertion loss. Accounting for other sources of light loss like fiber coupling and modulator losses (7 dB) and a 2 dB margin, a laser output power of 7 dBm (5 mW) is required.

Commercially-available tunable lasers, including our prototypes, can generate an output power of 16 dBm (40 mW) [51]. A single

laser can thus be shared across up to 8 transceivers. So a rack with 256 uplinks would only need 32 tunable laser chips plus any additional lasers for fault tolerance. Tunable lasers with higher output power and receivers with better sensitivity [73] have been demonstrated, which would allow an even higher degree of laser sharing.

Our disaggregated laser designs trade-off an increase in complexity and chip area (and, hence, power and cost) to achieve sub-nanosecond tuning latency. Such laser sharing amortizes this overhead and makes it a good trade-off. We note that laser sharing is made possible by Sirius' use of load balanced routing as it allows all transceivers on a node to use the same wavelength at any timeslot. Furthermore, the fact that the network schedule is known in advance also allows for the design variant, shown in Fig. 4c, comprising a bank of standard tunable lasers operating in a pipelined fashion. For a system with a 100 ns total slot duration and tunable lasers with a worst-case tuning time less than 100 ns, as achieved with our custom board, the tuning latency can be hidden by using a bank of two tunable lasers (plus an additional laser as back-up.)

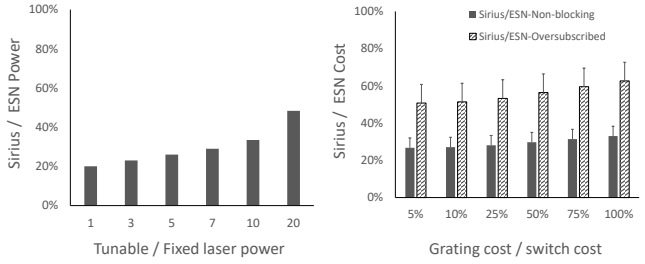
Fault tolerance. A key advantage of Sirius' passive core is its future-proofness and its robustness to failures due to the lack of any mechanical and electrical components. Our transceiver is also designed with robustness in mind: lasers are the prominent cause of transceiver failures, so separating them from the rest of the transceiver makes them field-replaceable and helps with shared backups, thus amortizing the cost of the backup lasers. Furthermore, accelerated-aging experiments on tunable lasers have been used to infer wear out times of tens of years and to show that tunability does not impose any reliability penalty as compared to fixed wavelength lasers [75].

However, nodes and transceivers can still fail, and load balanced routing does increase the blast radius of such failures, particularly with a rack-based deployment: the failure of a rack switch today only impacts the servers in that rack while with Sirius, it impacts all other nodes too as their traffic was routed through the failed rack. An increased blast radius is particularly problematic for "grey failures" that are sporadic or do not present themselves till a link is actually used. On the positive side, the interconnection of rack-pairs every few microseconds allows for low overhead yet fast failure detection, even for many grey failures. Thus, we can ensure quick datacenter-wide communication of any detected failures to prevent blackholing of traffic during failures. Further, any node and transceiver failures only result in a proportional loss of network bandwidth. For e.g., the failure of a node in a deployment with N nodes means that the effective uplink bandwidth of each node is reduced by $\frac{1}{N}$. For any failures that cannot be remedied immediately, the network schedule for all the nodes can be adjusted to omit the failed node and hence, regain any lost bandwidth, albeit at the expense of extra mechanisms for consistent updates of the nodes' schedules.

5 COST AND POWER ANALYSIS

The power and cost gains of Sirius over an electrically-switched network (ESN) are primarily due to its flat topology. As shown in the analysis below, by replacing the hierarchy of power-hungry switches and transceivers with a single-layer of passive gratings, it can reduce network power and cost by almost three quarters.

We analyze a large datacenter with 4,000 racks. For ESN, we consider a non-blocking network with 25.6 Tbps switches that are



(a) Power. **(b) Cost.**

Figure 6: (a). Sirius' power is 23-26% that of an electrically-switched Clos network (ESN) if tunable lasers consume 3-5× the power of fixed lasers. (b). Sirius costs only 28% as compared to a non-blocking ESN, with gratings costing 25% of electrical switches and tunable lasers 3× fixed lasers (5× for error bars).

estimated to consume 500 W [8] and optimistically assume a cost $5,000, and can be equipped with 400 Gbps transceivers that consume 10 W and cost $1/Gbps [38]. The network comprises four layers of switches with up to six transceivers across an end-to-end path. In contrast, for Sirius, the end-to-end path comprises a single layer of rack switches, a layer of gratings and two tunable transceivers. To account for the load balancing overhead in Sirius, we double the number of transceivers and rack uplink capacity.

Since tunable lasers consume more power, Fig. 6a shows the relative power of Sirius as compared to a non-blocking ESN with varying power overhead of the tunable laser. Laser manufacturers estimate that the fast tunability will introduce around 3-5× power overhead which is in line the 3.8 W power consumption of off-the-shelf tunable lasers as compared to ∼1 W for fixed laser. Much of the power consumption for the tunable laser is due to the need for a temperature controller to ensure wavelength stability and could be reduced significantly with more efficient cooling of the laser. Even assuming that the tunable laser consumes 3-5× the power of a fixed laser, the overall network power is only 23-26% that of ESN. We also analyzed efforts for network power reduction like the co-packaging of transceivers with the switch ASIC. Even with such optical copackaging, expected by 2023 with 51.2 Tbps switches, Sirius offers a similar power advantage.

A full cost analysis is harder to pin down. For e.g., tunable transceivers today are used in long-haul networks and cost around 10-15× short-reach transceivers. This is partly due to tunability but primarily due to the fact that they have much more complex circuitry to be able to transmit over thousands of kilometers and also due to significantly lower volumes. Instead we use packaged chip area for our laser and it power as a first-order yet conservative proxy for fundamental cost. Both indicate that Sirius' tunable laser will likely cost 3-5× a standard laser. Another factor is the cost of the gratings. They can be fabricated as etchings on Silicon, Silicon Nitride and PLC (Planar Lightwave Circuits), and at volume, are estimated to cost less than 25% the cost of electrical switches. Fig. 6b shows that Sirius cost is only 28% that of ESN when the grating cost is 25% of electrical switches, assuming a tunable laser is 3× the cost of a fixed laser. Even when comparing to an 3:1 oversubscribed ESN, Sirius only costs 53% while offering non-blocking connectivity which translates to significant performance gains (§7). Finally, we also considered

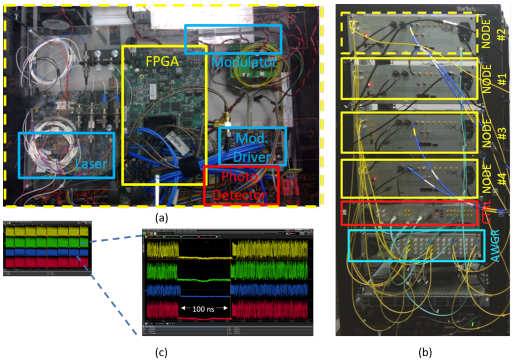


Figure 7: The Sirius prototype: (a) a Sirius node (inside), (b) the 4-node prototype, (c) the oscilloscope trace.

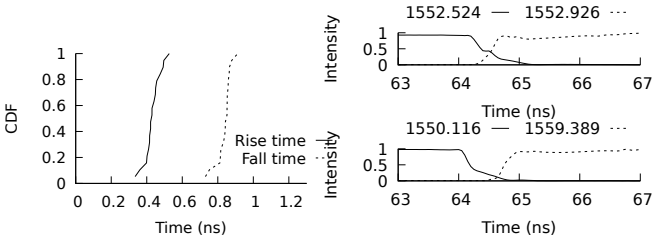
an electrically-switched variant of Sirius whereby the gratings are replaced with electrical switches plus transceivers, thus leveraging Sirius' topology and routing but without optical switching; we find that Sirius' cost is only 55% of this variant too.

6 PROTOTYPE IMPLEMENTATION

In the following, we describe the implementation of our four-node Sirius prototype. We start by describing our first generation testbed, which used our optimized off-the-shelf tunable laser with a worst-case tuning latency of 92 ns, and then discuss the second generation of our testbed that demonstrates 912 ps tuning using our custom optical chip leading to 3.84 ns for end-to-end reconfiguration. The goal of our prototype experiments is to evaluate the performance of our fast tunable laser (§3.2) combined with the traffic-scheduling and time-synchronization mechanisms (§4).

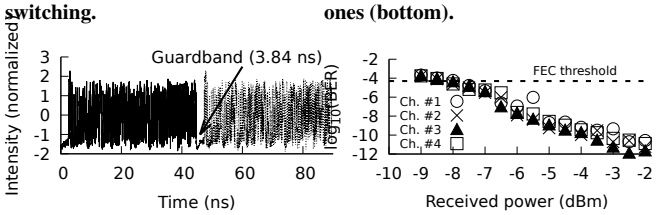
Sirius v1. We use a Xilinx UltraScale VCU108 FPGA to emulate a Sirius node. As depicted in Fig. 7a, we connected each FPGA to a custom PCB daughter board to enable fast tuning of the laser (§3). The FPGA was also connected to an external Mach-Zehnder modulator, operating at 25 Gbps using non-return-to-zero (NRZ) coding. The modulated light was coupled into fiber and connected to an external AWGR (bottom box in figure Fig. 7b). Depending on the wavelength, the AWGR redirects the signal to one of the four FPGAs in the testbed (see Fig. 7b). To measure the end-to-end performance, the FPGAs transmit a pseudo random binary sequence (PRBS) to each other following a cyclic schedule. On the receipt of the data, each FPGA compares the received stream with the expected PRBS to measure the bit error rate (BER). We use a guardband (§4.5) of 100 ns to account for the laser tuning time (92 ns) and cell preamble. In our experiments, we demonstrate post-FEC error-free transmission (BER $< 10^{-12}$) for a continuous period of more than 24 hours. For visualization purposes, we connected the electrical input channels of the FPGAs to the high sample rate oscilloscope (Fig. 7c), from which it is possible to visualize the interpacket gap of 100 ns.

Sirius v2. Next, we fabricated a packaged version of the optical chip shown in Fig. 3d to lower the tuning latency as compared to the standard tunable laser used in Sirius v1. The chip comprises an array of 19 SOAs and, hence, can tune across 19 wavelengths. We were limited by the chip area in the fabrication process but can use multiple chips to tune across a larger set of wavelengths. Tuning



(a) CDF of the SOA fall and rise time, illustrating sub-nanosecond switching.

(b) Switching between two adjacent wavelength (top) and distant ones (bottom).



(c) Captured burst waveforms of consecutive cell slots.

(d) BER measurement for four switching wavelengths.

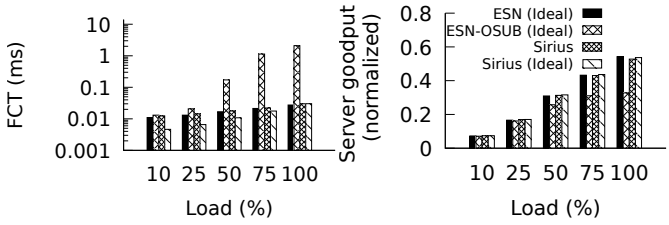
Figure 8: System demonstration of fast switching.

from λ_i to λ_j requires turning SOA$_i$ off and turning SOA$_j$ on. The tuning latency of the laser is thus determined by slower of the SOA turn-on and turn-off events. Fig. 8a shows a CDF of the on and off latencies for all SOAs on our chip; the worst-case values measured are 527 ps and 912 ps respectively. Further, this tuning latency is largely independent of the distance between the source and destination wavelength (as long as both are within the gain range of the SOAs). To illustrate this, in Fig. 8b we show switching events at the two extremes, i.e., between wavelengths that are adjacent to each other (resp. 1552.524 nm and 1552.926 nm) and wavelengths that are farther away across the C-band (resp. 1550.16 nm and 1559.389 nm). In both cases, the tuning latency is less than 900 ps. We also repeated the same experiments using modulated light, bumping the speed to 50 Gbps using four levels of pulse amplitude modulation (PAM-4) as used in state-of-the-art 400 Gbps transceivers with 8 lanes of 50 Gbps. Since the tuning time enabled by our custom chip was significantly lower than in our first-generation prototype, we were also able to reduce the guardband to 3.84 ns (including laser tuning time and cell preamble) as shown in Fig. 8c. To accommodate fast switching, we also implemented sub-nanosecond CDR (§A.1) and to cope with the multi-level signal encoding, we also developed a custom digital signal processing algorithm to guarantee fast equalization [68]. Both techniques leverage the cyclic schedule to "cache" the relevant parameters instead of having to learn them from scratch. We report the BER results for four different wavelengths in Fig. 8d. Across all of them, our prototype was able to achieve post-FEC error-free transmission at -8 dBm (0.16 mW) of received power.

Finally, we also performed an experiment to measure the accuracy of the time synchronization protocol by measuring the clock phase difference between two separate FPGAs. Over 24 hours, the maximum deviation was $\pm$ 5 ps, which is significantly below the symbol duration time (40 ps at 25 GBaud).

Hardware changes. Deploying Sirius in production would require some of the FPGA logic that we developed for our prototype to be moved onto the server NICs (for server-based deployments) or to

the rack switches (for rack-based deployments). In particular, nodes need to be able to use an external clock oscillator (or adjust the internal one) and maintain W FIFO output queues per port with W being the number of wavelengths used (see §4.1), which are drained in a round-robin fashion every slot. In addition, nodes need to implement the routing (§4.2) and congestion-control logic (§4.3), e.g., by leveraging programmable network hardware via P4 [58]. The logic for time and clock synchronization (§4.4), instead, can be implemented through a micro-controller as it does not need to execute on a packet-by-packet basis.



(a) 99th-perc. FCT. **(b) Average goodput.**

Figure 9: 99th-perc. flow completion time (FCT) for short flows and average goodput as we vary the network load.

7 SIMULATION RESULTS

In this section, we evaluate the performance of Sirius at scale, using a packet-level network simulator. Our analysis shows that Sirius can match the performance of an ideal non-blocking folded Clos network, achieving comparable throughput and flow completion time (FCT) for latency-sensitive flows with significantly lower power and cost (§5). This demonstrates the importance of a vertically-integrated design, combining novel optical devices (§3.2) and electronics (§4.5) with a bespoke network stack (§4).

Workload characteristics. We generate a synthetic workload, modeled after published datacenter traces [1, 31]. Flow sizes are heavy tailed, drawn from a Pareto distribution with shape parameter 1.05 and mean 100 KB [2, 3]. This distribution creates a heavy-tailed workload where the majority of flows are small, but the majority of traffic is from large flows, as is commonly observed in production networks. Flows arrive according to a Poisson process with uniformly randomly chosen sources and destinations. Each simulation generates approximately 200,000 flows.

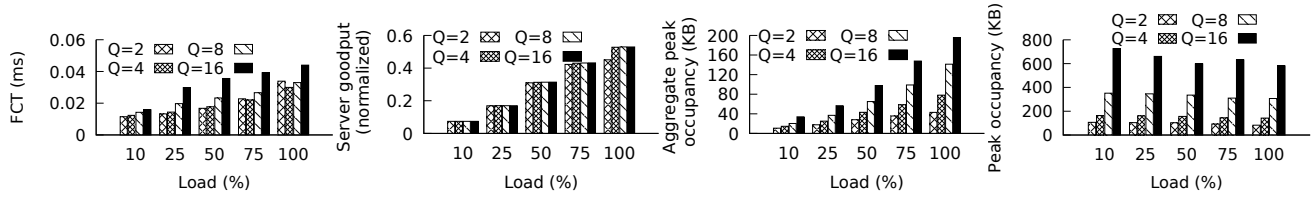
Network setup. We simulate a Sirius rack-based deployment datacenter with 128 racks, each comprising 24 servers (3,072 servers in total) and we compare the performance of an electrical network and Sirius. Each ToR switch is equipped with 8 uplinks, each operating at a native rate of 50 Gbps. We chose 8 uplinks for 128 racks as this yields the same ratio as 256 links for 4,096 racks (§4.1). For electrical networks, we adopt the three-tier folded Clos topology prevalent in production datacenters [31, 65, 70]. We evaluate both a non-oversubscribed setup (ESN (Ideal)) and one with a 3:1 oversubscription at the aggregation tier beyond the racks (ESN-OSUB (Ideal)). For routing and congestion control of these electrical baselines, we consider an idealized setup that assumes per-flow queues and back-pressure mechanisms at all switches and it uses packet spraying [23] to forward packets on all available paths. While this solution would be very costly to implement in practice (if possible at all) due to the large number of queues needed, it does provide an upper bound on the performance achievable by *any* rate control and routing protocol across an electrically switched network. This removes any bias due to the specific shortcomings of existing load-balancing and congestion-control protocols and enables us to focus on the fundamental trade-offs between Sirius and electrically-switched networks.

In Sirius, each uplink transceiver is driven by a fast tunable transceiver. By default, we assume 90-ns transmission slots (corresponding to a total cell size of 562 bytes) and we conservatively set the guardband between consecutive transmission slots to 10 ns, resulting into a total slot duration of 100 ns. Unless otherwise noted, we use 50% more uplinks on each ToR switch to compensate for the loss in throughput due to the cyclic scheduling (see §4.2). As shown in §5, even after doubling the uplinks per node, Sirius remains power- and cost-efficient compared to both a non-blocking and an oversubscribed electrical network. We use Sirius' congestion control with a per-destination queue size of 4 at each node (§4.3).

Network Load We start our analysis by comparing the behavior of the electrical and Sirius setups for different values of network load in Fig. 9. We define the load $L = \frac{F}{R \cdot N \cdot \tau}$ where F is the mean flow size, R is the per-server bandwidth, N is the number of servers, and τ is the mean inter-arrival flow time. For example, $L = 1$ means that, on average, there are N flows in the system. Fig. 9a shows the 99th-percentile FCT for short flows (flow size $<$ 100 KB) while Fig. 9b reports the average server received goodput, measured as the total number of bytes received during the simulation divided by the total simulation time and normalized by $N \cdot R$. The former metric captures the ability of providing low-latency communication for latency-sensitive flows while the latter measures the overall network goodput achieved.

SIRIUS significantly outperforms ESN-OSUB (Ideal) for both the FCT (reduced by up to 86%) and goodput (increased by up to a factor of 6.7). The reason is that, by introducing oversubscription, ESN-OSUB (Ideal) restricts the amount of available bandwidth for inter-rack communication, which leads to silo-ed performance and overall lower efficiency. In contrast, SIRIUS provides a flat network with uniform bandwidth among all racks, closely matching the performance achieved by ESN (Ideal). This is an important result because it shows that Sirius can achieve the performance of an (idealized) non-blocking electrical topology at only a fraction of power and cost. While with oversubscription it is possible to build a cheaper and less power-hungry network at the cost of reduced performance, Sirius shows that by taking advantage of optical switching it is feasible to reduce expenses without sacrificing performance.

To evaluate the behavior of Sirius's congestion-control protocol, in this experiment we also include an additional baseline (SIRIUS (IDEAL)), corresponding to an idealized version of Sirius that, instead of the Sirius' congestion-control protocol, uses per-flow queues and a back-pressure scheme, similar to the one used for ESN (Ideal). While this would not be a practical solution, it provides us with a performance bound and it enables us to quantify the penalty introduced by our congestion control. For $L < 50\%$, SIRIUS (IDEAL) exhibits a lower FCT than SIRIUS, resp. 63% for $L = 10\%$ and 55% for $L = 25\%$. This is expected because the additional latency required to issue requests and receive grants delays the transmission of new flows compared to SIRIUS (IDEAL) in which flows can be



(a) 99^{th}-perc. FCT. **(b) Average goodput.** **(c) Queue peak occupancy.** **(d) Out-of-order buffer.**

Figure 10: 99^{th}-perc. FCT for short flows, average goodput, and peak aggregate queue occupancy for different queue sizes Q.

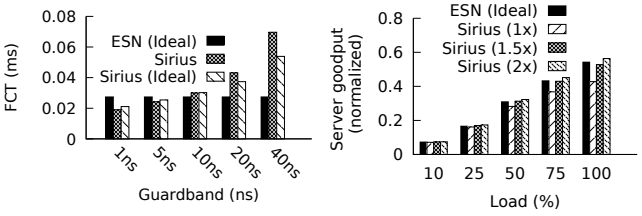
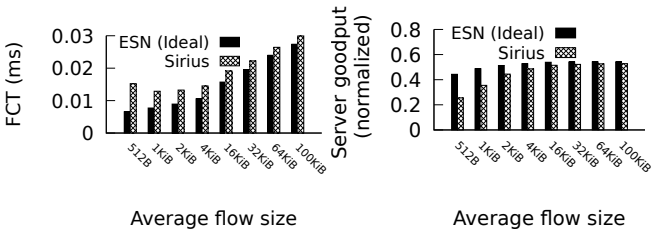


Figure 11: 99^{th}-perc. FCT for short flows when varying guardband ($L = 100\%$).

Figure 12: Average goodput as we vary the network load for different uplink bandwidth.



(a) 99^{th}-perc. FCT. **(b) Average goodput.**

Figure 13: 99^{th}-perc. FCT for short flows and average goodput for different average flow sizes.

transmitted immediately. However, as the load increases, the network becomes more congested and, hence, the additional overhead becomes less critical. This effect is not visible for long flows (Fig. 9b) because the additional latency only impacts the startup time. This demonstrates the ability of our protocol to mimic the behavior of an idealized congestion control while only consuming a small amount of memory resources as we see next.

Queue size and memory occupancy. To measure the impact of the queue size Q on the congestion-control performance and memory occupancy (§4.3), in Fig. 10 we repeat the same experiments of Fig. 9 varying the size of each individual queue from 2 cells to 16 and we measure the FCT and server goodput as well as the aggregate *maximum* queue occupancy per ToR switch. We also report on the peak size of the buffer needed at the server to reorder the packets arrived out of order before delivering them to the application layer. Intuitively, larger queue sizes lead to higher FCT and memory occupancy as well as a higher number of out-of-order packets. However, if the queue is too small ($Q = 2$), the server goodput at high load is somewhat reduced because the queues might not be able to absorb a burst of traffic. Therefore, based on this analysis, we select a value of 4 for our setup as this gives the best combination of FCT and server goodput while still retaining a very low peak aggregate memory utilization (78.2 KB in the worst case (Fig. 10c) and a peak size of the reorder buffer at the servers of 163 KB per flow (Fig. 10d).

Guardband. Next, we evaluate the impact of the size G of the guardband on Sirius performance by varying it from 1 ns to 40 ns (in the previous set of experiments it was set to 10 ns). To enable meaningful comparisons, as we vary the guardband, we also proportionally adjust the slot length to ensure that the guardband always accounts for 10% of the total slot length. In Fig. 11 we only report the results of FCT as the guardband is mostly relevant for latency while the impact on goodput is negligible. The chart shows that low values of G significantly improves the relative performance of SIRIUS compared to ESN (Ideal). Vice versa, as the guardband increases, the FCT of SIRIUS worsens as the length of the epoch grows proportionally and, hence, the queuing latency (i.e., the time a packet has to wait at an intermediate node) negatively affects the FCT. High values of G also magnify the difference between SIRIUS and SIRIUS (IDEAL) as the cost of waiting an additional epoch before start transmitting a new flow becomes more relevant. These results motivate the need for fast tunable laser and fast CDR to minimize the guardband and achieve lower end-to-end latency.

Uplink bandwidth. We assess the impact of the uplink bandwidth on Sirius performance. As explained in §4.2, the load-balancing routing adopted by Sirius removes the need for on-demand scheduling but on the negative side it extends the flow path length, resulting in 50% lower throughput in the worst case. In principle, this would imply that to achieve full throughput, each ToR switch should be equipped with twice the number of uplink transceivers. The bursty and stochastic nature of datacenter traffic, however, makes the occurrence of the worst-case scenario (all-to-all communication) unlikely, thus reducing the number of additional uplinks needed. We show this in Fig. 12 where we plot the server goodput for three different configurations of SIRIUS, resp. with 1x, 1.5x (the value used in prior experiments), and 2x the number of transceivers of ESN (Ideal) (we omit the FCT as the uplink bandwidth has no noticeable impact on short flows). The results indicate that at low load no additional transceivers are needed to match ESN (Ideal)'s goodput. As the network traffic increases, however, the increased path length of Sirius begins to impact overall goodput. For example, for $L = 100\%$, SIRIUS without any additional transceivers achieves only 79% of the goodput achieved by ESN (Ideal). However, just adding 1.5x more transceivers is sufficient to match ESN (Ideal) performance without having to resort to doubling the number of transceivers.

Flow size. We conclude our analysis by evaluating the impact of varying the average flow size F from 512 bytes to 100 KB, i.e., the value that we used in all other experiments. The goal is to understand

the overhead due to using fixed-size cell in Sirius. The results in Fig. 13 show that for the $F = 512$ byte, the use of fixed-size in Sirius leads to 2.3x increase in FCT and a reduction in average goodput of 1.7x compared to ESN (Ideal), which, instead, uses variable-sized packets. This is not surprising: given the long-tail nature of the Pareto distribution, $F = 512$ byte will result in a median size flow of just 46 byte, i.e., 12 times smaller than the cell size used in our experiments. However, as F grows, the mismatch between flow and cell sizes is reduced and Sirius approximates ESN (Ideal)'s performance. For example, for $F = 16$ KB, which corresponds to a median flow size of 1,506 bytes, the FCT gap is only 1.2x (resp. 1.05x for the average goodput).

8 RELATED WORK

Reconfigurable datacenter networks. The shortcomings of today's electrical networks have motivated reconfigurable networks realized using optical [14, 15, 25, 43, 57, 74, 76] and wireless technologies (RF-based [33, 81] and free-space optics [29, 34]). The switching speed of the underlying technologies, however, ranges from microseconds [29, 43, 57] to milliseconds [15, 25, 34, 74]. They thus rely on the fact that most of the bytes come from large flows, complementing the high-capacity reconfigurable network with an electrical or emulated packet-switched network. This, however, limits their appeal in terms of overall power and cost savings. Further, having two separate networks adds management overhead; for example, mapping traffic onto two networks is challenging [6, 65]. Finally, even the assumption regarding traffic stability may not hold for emerging cloud workloads (§2). Kassing et al. [37] show that coupling static expander topologies can offer significant cost benefits over non-blocking topologies. However, they still rely on electrical switches whose scaling appears gloomy. Instead, Sirius provides an all-optical network offering efficient non-blocking connectivity and ultra-fast reconfiguration with good future scaling potential.

Optical switches. Optical switches provide high bandwidth and low latency but vary in terms of switching time by almost six orders of magnitude and in terms of their maturity from off-the-shelf switches to proof-of-concept prototypes [17, 71].

At one end of the spectrum, *optical circuit switches* based on piezo-electric [56], MEMS [10, 67], and liquid-crystal [36] technologies can support a few hundred ports, switch in millisecond to microsecond timescales and are available commercially. At the other end, several technologies are being studied for optical *burst switches* with nanosecond switching times. This includes space-switching technologies such as Mach-Zehnder Interferometers (MZI) [41], SOAs [9], hybrid MZI-SOA [18], and ring resonators [16]. The crucial challenge with most space switching technologies is that the fundamental building block is a 2x2 switching element, which is then cascaded to create a larger switch at the cost of higher loss and noise [18]. Further, as the switches are active components, they still need to be synchronized with the nodes and their power savings compared to traditional networks are less pronounced. Wavelength-switching solutions based on tunable lasers [24] and AWGRs [59, 60, 77, 79] represent a promising solution due to the passive nature of the core switch element. Most existing solutions, however, are limited by relatively slow tunable lasers ($\geq$ 100 ns) or slow CDR, which negatively impacts the performance for short flows. Conversely, Sirius uses vertical integration and builds upon fast tunable lasers [40, 68] and burst-mode receivers [20, 21] to enable a flat network and a full system prototype that achieves fast *end-to-end* reconfiguration in a scalable fashion.

More generally, recent papers on fast optical switches focus on the performance of the actual device, ignoring the system-level challenges involved such as congestion control, scheduling, and time synchronization. In contrast, Sirius takes a cross-layer approach and show that by co-designing the optics with the hardware and network stack, it is possible to achieve a holistic and more efficient solution.

Network scheduling. On-demand scheduling at sub-microsecond granularity at datacenter scale is very hard. An alternative to dynamic scheduling [7, 44] is to statically assign wavelengths between all port pairs [39, 45] but this reduces scalability to a few tens of nodes [45]. In contrast, Sirius uses a static schedule in combination with load balanced routing, which eliminates the need for demand collection and estimation as well as the computational complexity of dynamic scheduling while being able to scale to the entire datacenter.

Load-balancing routing is also at the core of recently proposed datacenter network architectures [48, 49, 69]. RotorNet [49] uses an optical circuit switch with 20 us switching latency and, hence, relies on a separate electrical packet-network to handle short flows. Opera [48] extends RotorNet to support low-latency traffic allowing short flows to take more hops so that they do not have to wait for the switch to reconfigure. While effective in reducing the tail latency, it comes at the expense of increased average path length, thus reducing the overall available bandwidth and potentially leading to a throughput penalty higher than 50%. Shoal [69], instead, uses electrical circuit switches, which can be reconfigured in nanoseconds, and, hence, it does not suffer from these limitations. However, its design focuses on a rack-scale network, which reflects in its choice of switching and transmission technology. Sirius, instead, eliminates electrical switches in server-based deployments and hence, can offer a lower power and lower cost solution that is future-proof against the slowdown of CMOS scaling, while allowing for further scalability using a rack-based deployment.

9 CONCLUSION

We present Sirius, a flat, optically-switched datacenter network for the post-Moore's law era. It supports end-to-end nanosecond switching, thus illustrating the viability of building non-blocking topologies for emerging cloud applications in a power- and cost-efficient fashion. This required solving several challenges, including a custom design for fast laser tuning, building a scheduler-less network, fine-grained time synchronization and efficient congestion control, all while ensuring good fault tolerance. To tackle these challenges we leverage vertical integration across the entire network stack—from the choice of the switching technology to its flat topology and network protocols. For e.g., Sirius' congestion control and time synchronization leverage the fact that the optical network offers deterministic latency and that the nodes are connected to each other periodically. These properties, in turn, derive from Sirius' passive core and load-balanced routing. Sirius leverages a unique feature afforded by the cloud environment whereby providers can and are willing to deploy wholesale changes to the network stack, particularly as the mainstream network technologies start to taper off.

ACKNOWLEDGMENTS

We would like to thank all the Microsoft Research interns that contributed over the years to the Sirius project: Grant Brodnik, Ciro Ceissler, Kari Clark, Adam Funnell, Nadeen Gebara, Thomas Gerard, Cenk Ibrahim Ozdemir, Hussein Kassir, Qi Li, Alana Marzoev, Nadesh Ramanathan, Arslan Sajid Raja, Pablo Wilke Berenguer, Hilda Xue and Aaron Zhao. We also would like to express our gratitude to the Optics for the Cloud team [61] at Microsoft Research for all their help and support, in particular Miguel Castro, Christos Gkantsidis, Thomas Karagiannis, Francesca Parmigiani and Ant Rowstron, as well to Dave Bragg, Jeff Cox, Mark Filer, Dave Maltz, and Adrian Power from Azure Networking for their feedback. This work has also benefited from interactions and engagements with all our academic partners through the Optics for the Cloud Research Alliance [62] and, in particular, with Polina Bayvel, Daniel Blumenthal, Tobias Kippenberg, Zhixin Liu, Phil Watts, and George Zervas. We would also like to thank Neon Photonics [53] for their high-port-count AWGR. Finally, we would like to thank our shepherd Ankit Singla and the SIGCOMM reviewers for their feedback and help.

REFERENCES

[1] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). In *SIGCOMM*.

[2] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. 2012. Less Is More: Trading a Little Bandwidth for Ultra-Low Latency in the Data Center. In *NSDI*.

[3] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. pFabric: Minimal Near-optimal Datacenter Transport. In *SIGCOMM*.

[4] Architectural Consideration for 100 Gb/s/lane Systems. 2018. http://www.ieee802.org/3/100GEL/public/18_03/ghiasi_100GEL_01a_0318.pdf.

[5] Hitesh Ballani, Paolo Costa, Istvan Haller, Krzysztof Jozwik, Kai Shi, Benn Thomsen, and Hugh Williams. 2018. Bridging the Last Mile for Optical Switching in Data Centers. In *OFC*.

[6] Hamid Hajabdolali Bazzaz, Malveeka Tewari, Guohui Wang, George Porter, T. S. Eugene Ng, David G. Andersen, Michael Kaminsky, Michael A. Kozuch, and Amin Vahdat. 2011. Switching the Optical Divide: Fundamental Challenges for Hybrid Electrical/Optical Datacenter Networks. In *SoCC*.

[7] Joshua L. Benjamin, Thomas Gerard, Domaniç Lavery, Polina Bayvel, and Georgios Zervas. 2020. PULSE: Optical Circuit Switched Data Center Architecture Operating at Nanosecond Timescales. *Journal of Lightwave Technology* 38, 18 (2020).

[8] Broadcom. 2020. 25.6 Tb/s StrataXGS Tomahawk 4 Ethernet Switch Series. https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56990-series.

[9] Nicola Calabretta, Wang Miao, Ketemaw Mekonnen, Kristif Prifti, and Kevin Williams. 2017. Monolithically Integrated WDM Cross-Connect Switch for High-performance Optical Data Center Networks. In *OFC*.

[10] Calient. 2020. 3D MEMS Optical Circuit Switch. http://www.calient.net/.

[11] A. Cevrero, I. Ozkaya, T. Morf, T. Toifl, M. Seifried, F. Ellinger, M. Khafaji, J. Pliva, R. Henker, N. Ledentsov, J.-R. Kropp, V. Shchukin, M. Zoldak, L. Halmo, I. Eddie, and J. Turkiewicz. 2018. 4×40 Gb/s 2 pJ/bit Optical RX with 8ns Power-on and CDR-Lock Time in 14nm CMOS. In *OFC*.

[12] Cheng-Shang Chang, Duan-Shin Lee, and Yi-Shean Jou. 2002. Load Balanced Birkhoff-von Neumann Switches: Part I. *Computer Communications* (2002).

[13] H. Jonathan Chao and Jin Soo Park. 1998. Centralized contention resolution schemes for a large-capacity optical ATM switch. In *IEEE ATM Workshop*.

[14] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. 2012. OSA: An Optical Switching Architecture for Data Center Networks with. Unprecedented Flexibility. In *NSDI*.

[15] Li Chen, Kai Chen, Zhonghua Zhu, George Porter, and Chunming Qiao. 2017. Enabling Wide-Spread Communications on Optical Fabric with MegaSwitch. In *NSDI*.

[16] Qixiang Cheng, Meisam Bahadori, Sébastien Rumley, and Keren Bergman. 2017. Highly-Scalable, Low-Crosstalk Architecture for Ring-Based Optical Space Switch Fabrics. In *IEEE Optical Interconnects Conference*.

[17] Qixiang Cheng, Sébastien Rumley, Meisam Bahadori, and Keren Bergman. 2018. Photonic switching in high performance datacenters. *Optics Express* 26, 12 (2018).

[18] Q. Cheng, A. Wonfor, J. L. Wei, R. V. Penty, and I. H. White. 2014. Demonstration of the feasibility of large-port-count optical switching using a hybrid Mach-Zehnder Interferometer-Semiconductor Optical Amplifier switch module in a recirculating loop. *Optics Letters* 39, 18 (2014).

[19] Stanley Cheung, Tiehui Su, Katsunari Okamoto, and S. J. B. Yoo. 2014. Ultra-Compact Silicon Photonic 512 × 512 25 GHz Arrayed Waveguide Grating Router. *IEEE Journal on Selected Topics in Quantum Electronics* 20, 4 (2014).

[20] Kari Clark, Hitesh Ballani, Polina Bayvel, Daniel Cletheroe, Thomas Gerard, Istvan Haller, Krzysztof Jozwik, Kai Shi, Benn Thomsen, Philip Watts, Hugh Williams, Georgios Zervas, Paolo Costa, and Zhixin Liu. 2018. Sub-Nanosecond Clock and Data Recovery in an Optically-Switched Data Centre Network. In *ECOC*.

[21] Kari Clark, Daniel Cletheroe, Thomas Gerard, Istvan Haller, Krzysztof Jozwik, Kai Shi, Benn Thomsen, Hugh Williams, Georgios Zervas, Hitesh Ballani, Polina Bayvel, Paolo Costa, and Zhixin Liu. 2020. Synchronous subnanosecond clock and data recovery for optically switched data centres using clock phase caching. *Nature Electronics* (July 2020).

[22] L. Coldren, G. Fish, Y. Akulova, J. Barton, L. Johansson, and C. Coldren. 2004. Tunable Semiconductor Lasers: A Tutorial. *IEEE Journal on Lightwave Technology* 22, 1 (2004).

[23] Advait Abhay Dixit, Pawan Prakash, Y. Charlie Hu, and Ramana Rao Kompella. 2013. On the Impact of Packet Spraying in Data Center Networks. In *INFOCOM*.

[24] Michael Düser and Polina Bayvel. 2002. Analysis of a Dynamically Wavelength-Routed Optical Burst Switched Network Architecture. *Journal of Lightwave Technology* 20, 4 (2002).

[25] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2010. Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers. In *SIGCOMM*.

[26] Adam Funnell, Kai Shi, Paolo Costa, Philip Watts, Hitesh Ballani, and Benn Thomsen. 2017. Hybrid Wavelength Switched-TDMA High Port Count All-Optical Data Centre Switch. *Journal of Lightwave Technology* 30, 25 (2017).

[27] Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosunblum, and Amin Vahdat. 2018. Exploiting a Natural Network Effect for Scalable, Fine-Grained Clock Synchronization. In *NSDI*.

[28] Thomas Gerard, Christopher Parsonson, Zacharaya Shabka, Polina Bayvel, Domaniç Lavery, and Georgios Zervas. 2020. SWIFT: Scalable Ultra-Wideband Sub-Nanosecond Wavelength Switching for Data Centre Networks. Technical Report. https://arxiv.org/pdf/2003.05489.pdf.

[29] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil R. Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel C Kilper. 2016. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *SIGCOMM*.

[30] Google. 2020. Cloud TPU. https://cloud.google.com/tpu/.

[31] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: A Scalable and Flexible Data Center Network. In *SIGCOMM*.

[32] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, Zhi-Wei Lin, and Varugis Kurien. 2015. Pingmesh: A Large-Scale System for Data CenterNetwork Latency Measurement and Analysis. In *SIGCOMM*.

[33] Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. 2011. Augmenting Data Center Networks with Multi-gigabit Wireless Links. In *SIGCOMM*.

[34] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. 2014. FireFly: A Reconfigurable Wireless Data Center Fabric Using Free-space Optics. In *SIGCOMM*.

[35] Intel Altera. 2017. Phase Locked Loop (PLL) IP Core User Guide. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_altpll.pdf.

[36] Masaki Iwama, Masanori Takahashi, Masayoshi Kimura, Yasuyoshi Uchida, Junichi Hasegawa, Ryo Kawahara, and Nobuyuki Kagi. 2015. LCOS-based Flexible Grid 1x40 Wavelength Selective Switch Using Planar Lightwave Circuit as Spot Size Converter. In *OFC*.

[37] Simon Kassing, Asaf Valadarsky, Gal Shahaf, Michael Schapira, and Ankit Singhla. 2017. Beyond fat-trees without antennae, mirrors, and disco-balls. In *SIGCOMM*.

[38] Vladimir Kozlov. 2018. Datacenter Optics – Market Forecast. https://arpa-e.energy.gov/sites/default/files/Kozlov_ENLITENED2018.pdf.

[39] Ashok V. Krishnamoorthy, Ron Ho, Xuezhe Zheng, Herb Schwetman, Jon Lexau, Pranay Koka, GuoLiang Li, Ivan Shubin, and John E. Cunningham. 2009. Computer Systems Based on Silicon Photonic Interconnects. *IEEE* 97, 7 (2009).

[40] Sophie Lange, A.S. Raja, Kai Shi, M. Karpov, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Fotini Karinou, X. Fu, J. Liu, A. Lukashchuk, Benn Thomsen, Krzysztof Jozwik, Paolo Costa, T. J. Kippenberg, and Hitesh Ballani. 2020. Subnanosecond Optical Switching Using Chip-Based Soliton Microcombs. In *OFC*.

[41] Benjamin G. Lee, Alexander V. Rylyakov, William M. J. Green, Solomon Assefa, Christian W. Baks, Renato Rimolo-Donadio, Daniel M. Kuchta, Marwan H. Khater, Tymon Barwicz, Carol Reinholm, Edward Kiewra, Steven M. Shank, Clint L. Schow, and Yurii A. Vlasov. 2014. Monolithic Silicon Integration of Scaled Photonic Switch Fabrics, CMOS Logic, and Device Driver Circuits. *Journal of Lightwave Technology* 32, 4 (2014).

[42] Ki Suh Lee, Han Wang, Vishal Shrivastav, and Hakim Weatherspoon. 2016. Globally Synchronized Time via Datacenter Networks. In *SIGCOMM*.

[43] He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter. 2014. Circuit Switching Under the Radar with REACToR. In *NSDI*.

[44] He Liu, Matthew K. Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M. Voelker, David G. Andersen, Michael Kaminsky, George Porter, and Alex C. Snoeren. 2015. Scheduling Techniques for Hybrid Circuit/Packet Networks. In *CoNEXT*.

[45] Yunpeng James Liu, Peter Xiang Gao, Bernard Wong, and Srinivasan Keshav. 2014. Quartz: A New Design Element for Low-latency DCNs. In *SIGCOMM*.

[46] Pablo Marin-Palomo, Juned N. Kemal, Maxim Karpov, Arne Kordts, Joerg Pfeifle, Martin H. P. Pfeiffer, Philipp Trocha, Stefan Wolf, Victor Brasch, Miles H. Anderson, Ralf Rosenberger, Kovendhan Vijayan, Wolfgang Freude, Tobias J. Kippenberg, and Christian Koos. 2017. Microresonator-based solitons for massively parallel coherent optical communications. *Nature* 546, 7657 (2017).

[47] Mellanox. 2014. InfiniBand Credit-Based Link-Layer Flow-Control. http://www.ieee802.org/1/files/public/docs2014/new-dcb-crupnicoff-ibcreditstutorial-0314.pdf.

[48] William M. Mellette, Rajdeep Das, Yibo Guo, Rob McGuiness, Alex C. Snoeren, and George M. Porter. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *NSDI*.

[49] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George C. Papen, Alex C. Snoeren, and George M. Porter. 2017. RotorNet: A Scalable, Low-complexity, Optical Datacenter Network. In *SIGCOMM*.

[50] William M. Mellette, Alex C. Snoeren, and George Porter. 2016. P-FatTree: A Multi-channel Datacenter Network Topology. In *HotNets*.

[51] Micro-Integrable Tunable Laser Assembly (ITLA), LambdaFLEX. 2020. https://www.lumentum.com/en/products/micro-itla-tunable-laser-300-khz.

[52] Samuel K. Moore. 2019. Another Step Toward the End of Moore's Law. *IEEE Spectrum* 56, 6 (2019).

[53] Neon Photonics. 2020. http://neonphotonics.com/.

[54] NVIDIA A100 Tensor Core GPU Architecture. 2020. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf.

[55] OIF to double data rate with a 224G electrical interface. 2020. http://www.gazettabyte.com/home/2020/6/17/oif-to-double-data-rate-with-a-224g-electrical-interface.html.

[56] Polatis. 2020. All-optical Circuit Switching. https://www.polatis.com/.

[57] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2013. Integrating Microsecond Circuit Switching into the Data Center. In *SIGCOMM*.

[58] Programmable Data Plane at Terabit Speeds. 2017. https://p4.org/assets/p4_d2_2017_programmable_data_plane_at_terabit_speeds.pdf.

[59] Roberto Proietti, Zheng Cao, Christopher J. Nitta, Yuliang Li, and S. J. Ben Yoo. 2015. A Scalable, Low-Latency, High-Throughput, Optical Interconnect Architecture Based on Arrayed Waveguide Grating Routers. *Journal of Lightwave Technology* 33, 4 (2015).

[60] R. Proietti, X. Xiao, K. Zhang, G. Liu, H. Lu, P. Fotouhiand J. Messig, and S. J. B. Yoo. 2018. Experimental Demonstration of a 64-Port Wavelength Routing Thin-CLOS System for Data Center Switching Architectures. *IEEE/OSA Journal of Optical Communications and Networking* 10, 7 (2018).

[61] Microsoft Research. 2020. Optics for the Cloud Group. http://opticsforthecloud.com/.

[62] Microsoft Research. 2020. Optics for the Cloud Research Alliance. https://www.microsoft.com/en-us/research/group/optics-for-the-cloud/#!optics-for-the-cloud-research-alliance.

[63] Mattia Rizzi, Maciej Lipiński, Tomasz Wlostowski, Javier Serrano, Grzegorz Daniluk, Paolo Ferrari, and Stefano Rinaldi. 2016. White Rabbit Clock Characteristics. In *ISPCS*.

[64] Rockley Photonics. 2019. Sailing through the Data Deluge. https://rockleyphotonics.com/wp-content/uploads/2019/02/Rockley-Photonics-Sailing-through-the-Data-Deluge.pdf.

[65] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George M. Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. In *SIGCOMM*.

[66] Neil Ryan, Michael Todd, Tom Farrell, Adrian Lavin, Pierre-Jean Rigole, Brian Corbett, Brendan Roycroft, and Jan-Peter Engelstaedter. 2008. A 10Gbps optical burst switching network incorporating ultra-fast (5ns) wavelength switched tunable laser sources. In *ICSO*.

[67] Tae Joon Seok, Niels Quack, Sangyoon Han, Richard S. Muller, and Ming C. Wu. 2016. Large-scale broadband digital silicon photonic switches with vertical adiabatic couplers. *Optica* 3, 1 (2016).

[68] Kai Shi, Sophie Lange, Istvan Haller, Daniel Cletheroe, Raphael Behrendt, Benn Thomsen, Fotini Karinou, Krzysztof Jozwik, Paolo Costa, and Hitesh Ballani. 2019. System Demonstration of Nanosecond Wavelength Switching with Burst-mode PAM4 Transceiver. In *ECOC*.

[69] Vishal Shrivastav, Asaf Valadarsky, Hitesh Ballani, Paolo Costa, Ki Suh Lee, Waltz Networks, Han Wang, Rachit Agarwal, and Hakim Weatherspoon. 2019. Shoal: A Network Architecture for Disaggregated Racks. In *NSDI*.

[70] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hoelzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *SIGCOMM*.

[71] Francesco Testa and Lorenzo Pavesi (Eds.). 2017. *Optical Switching in Next Generation Data Centers*. Springer.

[72] L. G. Valiant and G. J. Brebner. 1981. Universal schemes for parallel communication. In *STOC*.

[73] B. Wang, Z. Huang, X. Zeng, D. Liang, M. Fiorentino, W. V. Sorin, and R. G. Beausoleil. 2019. 50 Gb/s PAM4 Low-Voltage Si-Ge Avalanche Photodiode. In *CLEO*.

[74] Guohui Wang, David G. Andersen, Michael Kaminsky, Michael Kozuch, T. S. Eugene Ng, Konstantina Papagiannaki, and Michael Ryan. 2010. c-Through: Part-time Optics in Data Centers. In *SIGCOMM*.

[75] T. Wipiejewski, Y.A. Akulova, Gregory Fish, P.C. Koh, Clint Schow, Peter Kozodoy, A. Dahl, M. Larson, M. Mack, T. Strand, C. Coldren, E. Hegbiom, S. Penniman, T. LiIjeberg, and L.A. Coldren. 2003. Performance and Reliability of Widely Tunable Laser Diodes. In *ECTC*.

[76] Yiting Xia, Mike Schlansker, T. S. Eugene Ng, and Jean Tourrihes. 2015. Enabling Topological Flexibility for Data Centers Using OmniSwitch. In *HotCloud*.

[77] Xian Xiao, Roberto Proietti, Kaiqi Zhang, and S. J. Ben Yoo. 2018. Experimental Demonstration of Flex-LIONS for Reconfigurable All-to-All Optical Interconnects. In *ECOC*.

[78] Yihan Li, S. Panwar, and H. J. Chao. 2001. On the performance of a dual round-robin switch. In *INFOCOM*.

[79] Yawei Yin, Roberto Proietti, Christopher J. Nitta, Venkatesh Akella, Christopher Mineo, S. J. Ben Yoo, and Ke Wen. 2013. AWGR-based All-to-all Optical Interconnects using LimitedNumber of Wavelengths. In *Optical Interconnects Conference*.

[80] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. 2007. High-Resolution Measurement of Data Center Microbursts. In *IMC*.

[81] Xia Zhou, Zengbin Zhang, Yibo Zhu, Yubo Li, Saipriya Kumar, Amin Vahdat, Ben Y. Zhao, and Haitao Zheng. 2012. Mirror Mirror on the Ceiling: Flexible Wireless Links for Data Centers. In *SIGCOMM*.

A APPENDIX

Appendices are supporting material that has not been peer-reviewed.

A.1 Clock and Data Recovery (CDR)

Existing transceivers use the Clock and Data Recovery (CDR) circuitry to appropriately sample the incoming bits to achieve error-free reception. Learning the correct sampling rate typically takes hundreds of microseconds [20] (*locking time*). This is not an issue for electrical network because for any given link the sender and receiver remain the same (e.g., the server NIC and the rack port). On the other hand, this represents a major hurdle for optical-switch architectures like Sirius in which in every slot the receiver will get data from a different sender. While burst-mode receivers have been demonstrated, they still exhibit locking times of 8 ns or longer [11], thus partly offsetting the benefits of Sirius's sub-nanosecond switching times.

To address this shortcoming, we developed *phase caching* [20, 21], a new CDR mechanism that ensures practically instantaneous locking time. We observe that the correct sampling rate for a given transmitter T and receiver D is determined by the frequency and phase difference between their clocks. Since in Sirius all clocks are synchronized, we only need to estimate the correct phase difference. We thus measure the phase difference at the receiver and apply it at the sender by means of a delay line so that the transmitted bits arrive perfectly in phase at the receiver. The phase delay has to be applied at the sender because in a traditional network architecture, the receiver cannot predict from which sender it will receive the next packet. This, however, requires frequent updates from the receiver to the sender to compensate for periodic phase drifts. In contrast, since Sirius adopts a cyclic schedule, every receiver knows the identity of the sender for the next slot and, hence, can apply the delay locally. This greatly reduces the overhead as no explicit updates are needed.

A.2 Time Synchronization

Achieving proper time synchronization requires estimating the delay between any pairs of racks and their relative time offset. Again, the key idea here is to leverage the passive core and to take advantage of Sirius' cyclic network schedule. By selecting the appropriate wavelength, a rack can transmit a cell to itself. For instance, in the example in Fig. 5b, by using wavelength A, the packet generated by rack 1 will be forwarded by the AWGR back to rack 1. This enables accurately measuring the delay $d_{i,AWGR}$ between a rack i and the AWGR by comparing the transmit and receive timestamp. Since the latter is completely passive and there is no re-timing or buffering, this delay is completely deterministic and it is only a function of the fiber length and wavelength used (which can be accounted for). Therefore, the time offset $o_{i,j}$ between rack i and j can simply be obtained by measuring the difference between the transmit timestamp T_i at the source and the receive timestamp T_j at the destination and subtracting the delays to and from the AWGR ($d_{i,AWGR}$ and $d_{j,AWGR}$), i.e.,

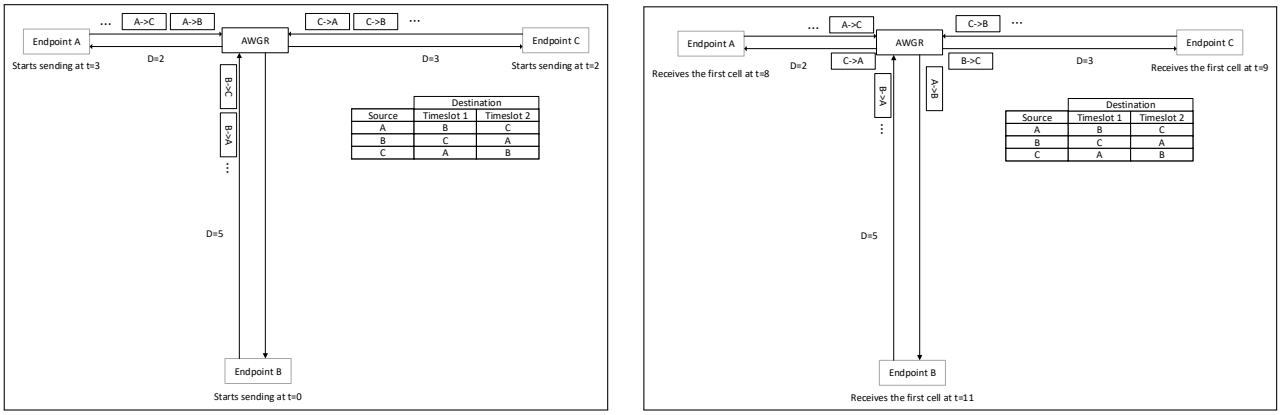
$$o_{i,j} = T_j - T_i - (d_{i,AWGR} + d_{j,AWGR})$$

In case of clock skew, the above formula would be incorrect. However, since clocks are frequency-synchronized, this does not apply. Also, while temperature variations do affect the fiber delay, their impact is relatively small (typically 20 ps per Kilometer of fiber for every Kelvin degree of change) and it can be factored out by periodically repeating the measures.

The information about the delay from each endpoint to the AWGR can be used to ensure the correct behavior even in presence of different fiber lengths. The goal is to ensure that the first cell of the first epoch from each rack arrive at the AWGR at the same time. To achieve this, once the bootstrap phase and time synchronization have been completed, each rack starts its first epoch at a different time depending on the distance from the AWGR: the longer this distance is, the sooner it will start so that the different distances are factored out. Given the cyclic nature of the schedule and the fact that all clocks are frequency-synchronized, if all the cells transmitted by all racks in the first slot of the first epoch arrive at the AWGR at the same time, it means that also the cells of the second slot will be synchronized and so on. We illustrate this by means of an example in Fig. 14. We consider a network with three endpoints (A, B, and C) which are connected to the AWGR with a fiber with a delay of 2, 5, and 3 time units respectively. As explained in §4.4, to ensure that cells arrive at the AWGR at the same time, endpoints with longer delays from the AWGR should start their epoch sooner. In this example, since B is the one with the longest delay to the AWGR, it would start sending the first cell of the first epoch at $t = 0$ while A will start at $t = 3$ and C at $t = 2$. At $t = 5$ (Fig. 14a), the first cells from all endpoints will enter the AWGR and, assuming one time unite delay to go through the AWGR, at $t = 6$ (Fig. 14b) the cells will leave the AWGR and will be delivered at the destination at $t = 8$, $t = 11$, and $t = 9$ respectively. While in the example, the differences between individual fibers are significant, in a real system, we would expect only minor differences among fibers (few centimeters due to fiber manufacturing tolerance) so the epoch starting times will be similar.

A.3 Congestion Control

Fig. 15 shows the pseudocode of the congestion-control protocol described in §4.3.



(a) t=5. **(b) t=6.**

Figure 14: Example of how synchronization works with different time delays. D is the fiber delay expressed in time units and we assume that each packet takes exactly one time unit to transmit.

Input: $N = \{n_i\}$: the set of all nodes in the system
Input: $F = \{(f_i, d_i, p)\}$: the set of flow tuples containing the flow ID f_i, the flow destination $d_j \in N$, and the number $p > 0$ of cells in LOCAL belonging to flow f_i
Output: $R = \{(d_i, n_j)\}$: the set containing all requests to send a cell to $d_i \in N$ via intermediate node $n_j \in N$

Function `compute_requests()`**:**
- $R \longleftarrow \emptyset$
- $C \longleftarrow N$ `// Initialize the set of candidate intermediate nodes`
- **while** $F \neq \emptyset$ **and** $C \neq \emptyset$ **do**
  - $(f, d, p) \longleftarrow$ `random_select(`F`)` `// randomly select and remove a tuple from the flow set`
  - $c \longleftarrow$ `random_select(`C`)` `// randomly select and remove a node from the candidate set`
  - $R \longleftarrow R \cup \{(d, c)\}$ `// Generate a request to forward a cell for` d `via intermediate node` c
  - **if** $p > 1$ **then**
    - $F \longleftarrow F \cup \{(f, d, p-1)\}$ `// reinsert the tuple in` F `after decrementing its value` p
  - **end**
- **end**
- **return** R

Input: $N = \{n_i\}$: the set of all nodes in the system
Input: $R = \{(s_i, d_j)\}$: the set of requests received in the previous epoch from $s_i \in N$ to forward a cell for destination $d_j \in N$
Input: $M = \{(d_i, m_i)\}$ the set containing the sum m_i of the packets queued and outstanding grants for each destination $d_i \in N$
Input: Q: the maximum queue size
Output: $G = \{(s_i, d_j)\}$: the set containing all grants to source node $s_i \in N$ for destination $d_j \in N$

Function `compute_grants()`**:**
- $G \longleftarrow \emptyset$
- **for** $(s, d) \in R$ **do**
  - **if** $\exists (d, m) \in M$ **such that** $m < Q$ **then** `// The request is accepted only if` m `is lower than` Q
    - $G \longleftarrow G \cup \{s, d\}$
    - $M \longleftarrow (M \setminus \{(d, m)\}) \cup \{(d, m+1)\}$ `// Increment the value` m `associated to` d
  - **end**
- **end**
- **return** G

Figure 15: The pseudocode of the congestion-control protocol described in §4.3.