Towards Sub-microsecond Optical Circuit Switched Networks for Future Data Centers

Joshua L Benjamin

A dissertation submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Electronic & Electrical Engineering
University College London

April 1, 2020

I, Joshua L Benjamin, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The combination of ever-growing data demand and the slowing down of Moore's law are creating substantial pressure to today's Data Center Networks, forcing them to scale. On one note, traditional scaling methods of increasing electronic packet switches (EPS) in the network have a significant impact on energy consumption and costs. On another, computing application performance suffers from long tail latencies when a network employs packet switching technology. Moreover, a holistic migration to optically switched networks further increases control plane complexity when duly replicating all functionalities available in current EPSs.

Hence, I propose PULSE, a broadcast-and-select optically circuit switched network, that has the potential to transform data center networks by reducing circuit establishment time to sub-microseconds, whilst consequently reducing latency, energy consumption and cost. We show how the development and re-arrangement of this all parallel disjointed optical data plane architecture can provide flexibility, modularity and scalability. The primary focus and novelty of this work is the development of the control hardware scheduler that enables OCS timeslot and wavelength computation in nanosecond speeds. Each rack contains a PULSE Network scheduler Processing Unit (NsPU), which is highly parallel and pipelined multi-core processor that functions at a 435 MHz clock speed for a 64-server rack when synthesized on 45nm CMOS library. The scheduler exploits parallelism and pipelining to compute the best possible resource matching configuration to an NP-hard problem within tens to hundreds of. My simulations show how PULSE's SDM/WDM/TDM based network can be configured to achieve above 90% sustained throughput, tolerant to scaling N-server racks and diverse traffic distributions, achieving median latency of 120ns and tail latency of 6.6μs. PULSE is a synchronized network that uses fast wavelength selection transceivers based on widely tunable DS-DBR lasers, coherent receivers and SOAs to achieve fast tuning capabilities that consume only 100-200 pJ/bit.

Impact Statement

The magnitude of data growth is forcing data center architectures to scale, resulting in increasing cost and energy. As we are heading towards the saturation of Moore's law, current electronically switched networks cannot continue to support the ever-growing data as chip designs are reaching their limits. However, the next decade forecasts highly dense heterogeneous Cloud data center networks, which will host 95% of the world's data traffic with extremely powerful processing units. PULSE, the novel ultra-fast optically circuit switched network and scheduler design, arising out of this thesis work and research has the potential to minimize latency in networks by 4-5 orders of magnitude (relative to electronic networks). Further, the proposed network solution is found to be cost-effective with reduced energy consumption. Moreover, this maiden ASIC-based hardware design reduces the reconfiguration cycle to sub-microseconds in an optical circuit switched network. In contrast to the conventional software-based switch configuration computation, this innovative idea of implementing algorithmic designs on ASICs for control can become a foundational concept for network engineers.

In optical networks engineering, intelligent networks with adaptive and reconfigurable topology with high flexibility and control is essential. PULSE addresses these requirements by re-arranging the topology at packet granular timescales while maximizing throughput and increasing tolerance to the demand. In terms of impact of the research at UCL, PULSE has opened a few areas of research: sub-nanosecond wavelength-timeslot selection, sub-microsecond scheduling and passive scalable broadcast-and-select network architectures for 10,000s of blades. A hardware equivalent software simulation environment built as part of this work could be used by research students to emulate the performance of network scheduling hardware behavioral models on MATLAB. The proposed novel scheduler unit (currently synthesized on 45 nm CMOS) in this research is implementable and can be enhanced substantially by adapting faster CMOS technology (7 nm) with more capability and features.

PULSE is now the world's leading pure all optical OCS network in research that establishes circuit configuration in sub-nanoseconds while scheduling in sub-

microseconds. In terms of academic research impact, the research in this work has been published and presented in leading optical communication conferences like IEEE/ACM HotI, PSC, ECOC and OFC (invited 2020). The work is currently under review for publishing in Journal of Lightwave Technology's Special Issue on Optical Interconnects 2020.

Acknowledgements

I would like to thank the LORD JESUS CHRIST for being the anchor and the only constant in my life. I thank Him for His grace, love and mercy on my life and for enabling me to complete this PhD. Without the grace of God, I would not be where I am today.

I would like to thank my supervisor, DR. GEORGIOS ZERVAS, for the valuable advice, support, guidance and patience with me through my PhD. Thank you for transforming the direction of my research, improving the technical quality of my work, arranging funding for my project, encouraging me, giving me ideas and feedback, paying high attention to detail, patiently reviewing my presentations, journals and conference papers in detail. Thank you for investing time into this research. Thank you for the productive meetings I had with you and for being patient even if they were quite long because of me.

I would like to thank PROF. POLINA BAYVEL for her support, words of encouragement and for providing feedback on my presentations, writing and research. Thank you for your valuable feedback and encouragements. I strongly believe that they have helped me to become a better researcher.

A huge thank you to all the fellow Optical Network Group (ONG) members for the many wonderful lunches, laughter, fun, motivation and for teaching me new skills and giving feedback that developed me as a person. I would like to especially thank THOMAS GERARD, DR. DOMANIÇ LAVERY and DR. ADAM FUNNELL for productive discussions, making time to discuss creative ideas and review my presentations, journal and conference papers. I would like to thank PARIS ANDREADES, VAIBHAWA MISHRA and HUI YUAN for supporting me with technical or casual discussions every day. Thank you all ONGers for making me feel at home and part of the family.

I would like to thank my PhD supervisors during the initial phase of my PhD, DR. PHILIP WATTS and DR. BENN THOMSEN, for patiently guiding me through the architectural principles and hardware design fundamentals. Thank you for investing time into this field of research during the many meetings I had with you.

I would like to thank LEE HEAGNEY and SCOTT LANDERS for putting up

with me when I constantly bothered them for file revisions and data back-up. I appreciate your valuable help and patience.

I would also like to thank my parents for their constant support, encouragement and love through my PhD. Thank you for believing in me and for reassuring your support. I could not have done this without you.

I would like to thank my sister for providing me emotional support, motivation, guidance and love during my meltdowns. Thank you for listening to my laments and constantly reminding me that I am not alone in this.

Finally, I would like to thank UCL, EPSRC-DTP for funding my research and for supporting me financially.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

1.1 The need for Data Center growth

1.1.1 The ever-growing data

There was a time when man's information needs were simpler [6]. Less than half a century ago, the information generated and shared were significantly lower compared to where we are today. The boom of the internet has led to the rise of many high-bandwidth technological breakthroughs that have advanced media, modern medicine, industries, gaming, information, social platforms and activities. Consequently, the amount of data generators, sharers and users have also increased enormously, making man a data hungry species. Within a short span of three decades, many milestones have been achieved on the development of internet and there has been a vast increase in the volume of data flow/exchange across the world.

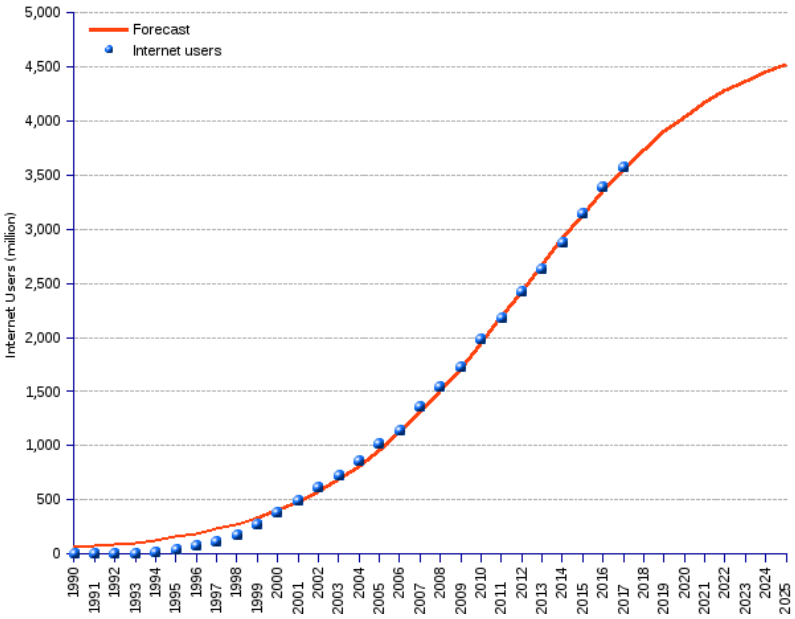


Figure 1.1: IP users since boom of internet and forecast [1]

As we step into the zettabyte era (1 ZB = 10^{21} bytes), Cisco's Visual Networking Index (VNI) predicts that annual global IP traffic will increase threefold to 4.8 ZB per year by 2022 [2]. The evolution of technology from TVs and computers to mobile devices like laptops, smart phones and tablets has contributed to a continual surge in the number of internet users. At the birth of the Internet of Things (IoT) between 2008-2009, there were more devices connected to the internet than there were people [7]. According to the same source, a group of Chinese researches confirmed that the Moore's law equivalent of the internet doubles in size every 5.32 years. Fig. 1.1 shows a seventy-fold increase in the number of internet users from 50 million in 1990 to 3500 million in 2017. An extrapolation suggests that this will continue to increase to another billion internet users by 2025. The number of connected devices is also predicted to increase as shown in Fig. 1.2 to almost 30 billion by 2022. Therefore, the usage of internet will also increase to 400 Exabytes per month by 2022, as shown by Fig. 1.3.

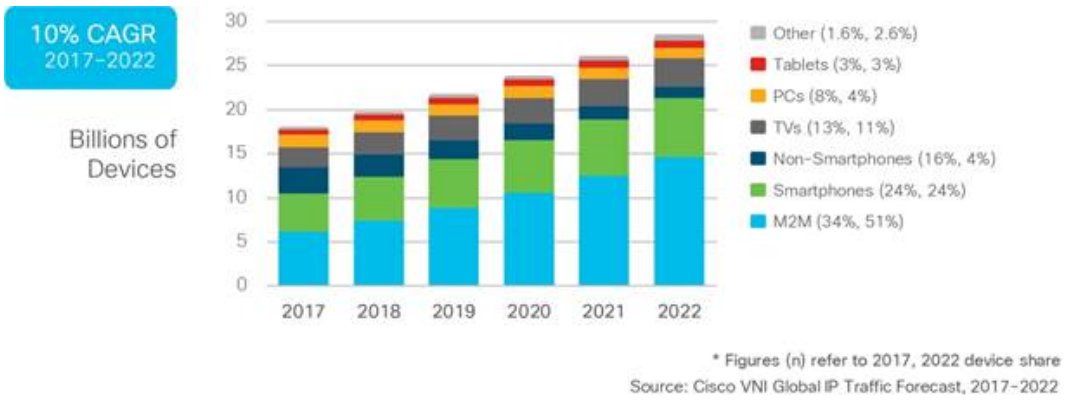


Figure 1.2: Cisco VNI Forecast: Devices Connected to the Internet 2017-2022 [2]

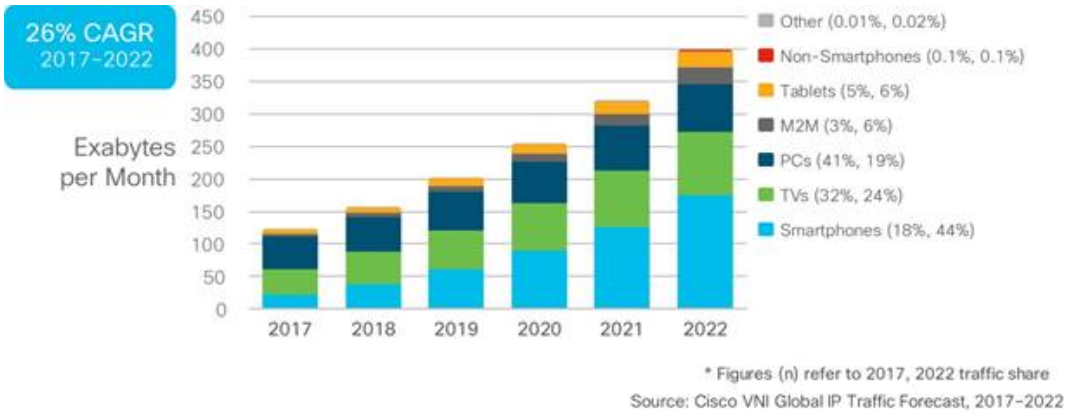


Figure 1.3: Cisco VNI Forecast: Global Internet Traffic Flow 2017-2022 [2]

1.1.2 Concerns of DCNs

An article in Data Economy Magazine [8] labels data centers as the key pillars of a new world that is built on digital and data. Hence, 'Big Data' and 'Cloud'-based business organizations tend to establish their own data center farm, which then behave as resource banks. They provide centralized storage, backups, management, networking and dissemination of data [9] by allocating their resources to customers. Data centers have functioned as the foundation of a majority if not all the sectors of the 21st century including economy, aviation, media, medicine etc. Most importantly, they have served as the backbone of the internet, which has grown drastically in the last 30 years.

1.1.2.1 Growing Data, Cloud, Power, Costs

As discussed, recent years have experienced a rapid increase in the rate of global data being generated and shared. Behind the scenes, corporate data centres (DCs) have also scaled their overall capacity in order to support the sheer magnitude of growth in data. According to Google, the demand for bandwidth in data centres doubles every 12-15 months [10]. Intel also anticipates that 70-80 % of their computing and storage systems will be deployed into data centres by 2025 [11]. While data centre researchers are battling against time to scale data centres to meet demand, they are also under tremendous pressure to control their cost and power consumption. In 2015, total data centre power consumption worldwide was 416.2 terawatt hours, while the national power consumption of UK for all purposes was approximately 300 terawatt hours [12]. In some cases, energy use is a substantial cost relative to the IT hardware itself [13]. Hence, restructuring data centre network architectures is required to enhance scalability and meet the growing demand, while mitigating energy consumption and operating costs.

1.1.2.2 If not Moore's law, then what?

Although Moore's law is slowing down, [14] suggests that the end of Moore's law does not indicate an end to faster computer applications. However, the implication is that the 'top' of the stack must drive performance gains (software, algorithms and hardware), rather than the traditional 'bottom' of the stack (semiconductor physics and silicon technology). The post-Moore era requires performance engineering, i.e. restructuring and optimization of software applications, network architecture, hardware electronics and algorithms, in order to boost computer application speeds. On this note, it is critically important to pay attention to the performance engineering of network architecture and hardware, specifically targeting the switching electronics that define the efficient operation of a network.

1.1.2.3 Scalability bottlenecks

The scalability bottlenecks of a data center interconnect lies in the network architecture, the switching devices and the protocol used. Arista defines scalability of DCNs as the ability to construct and expand the network with simple, repeatable designs that can accommodate increased traffic or new devices without affecting applications, workflows or cost per port [15]. Such a network can create a linear scalability model that enables performance/cost linearity. Hence, one of the critical requirements of DCN is linear scalability to a large number of high performance interfaces without the use of over-subscription. However, traditional network architectures limit the scaling linearity of a network to a specific number of end-points. After reaching this point, a super-linear regime is entered where an increase in multiple switches or levels of switches is required to support more nodes; this has a direct impact on the scalability of the network.

1.1.2.4 Modern Cloud Workloads

The network traces from a large-scale production cloud service were analysed and found to show the following packet size distribution in fig. 1.4(a) [3]. Over 34% of the packets comprise less than 128 bytes (11 ns at 100 Gbps). While 97.8% of the packets are shown to consume 576 bytes or less here, 91% of the packets generated by Facebook's in-memory cache are also shown to be 576 bytes or less [16] (43 ns at 100 Gbps). The short duration of these packet sizes emphasise the requirement of rapid switch reconfiguration in modern DC workloads.

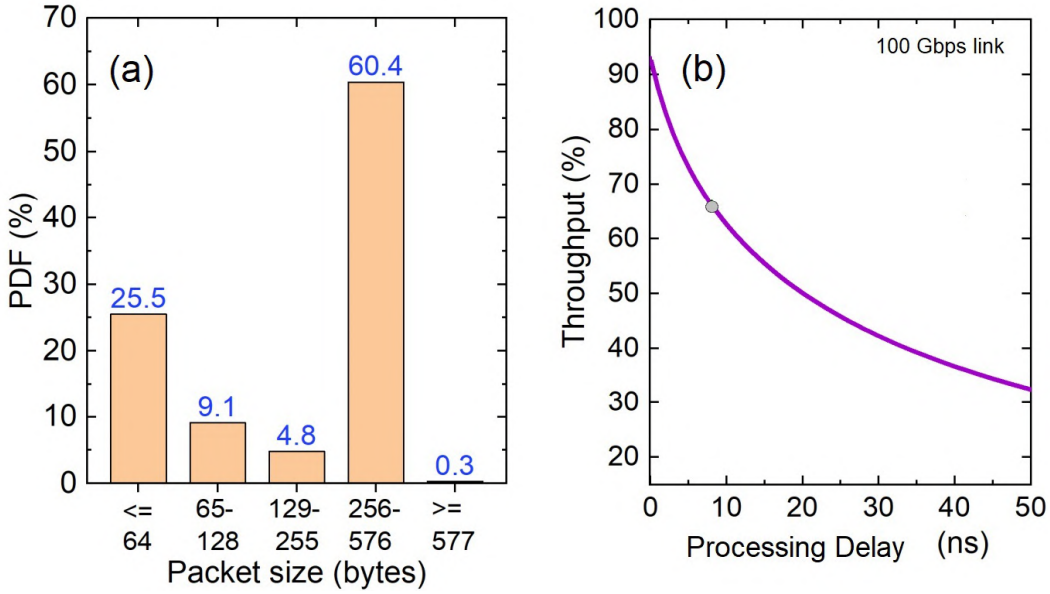


Figure 1.4: Cloud application (a) Packet Size Distribution (b) Throughput against Processing Delay (ns) [3]

Figure 1.4(b) shows that, at 100 Gbps link rate, a 20 ns processing overhead CDR locking, synchronization delay has a 50% degradation in throughput. Hence, such applications leave no margin for packet processing, requiring rapid connections to be established. In this thesis, the aim is to create an ultra-fast OCS network and scheduling algorithm that can cater to networks with such needs.

1.1.3 Limitations of electronic packet switches (EPSs)

1.1.3.1 High-speed switch ASIC investment

Another DCN scalability bottleneck lies in the switching elements used in the interconnect. Although high capacity switches like Broadcom's Tomahawk and Jericho were introduced, the number of switch ports remained as low as 64 ports and the bandwidth was locked per link, meaning that if a link is unused, the bandwidth cannot be used elsewhere in the network. Apart from the link bandwidth, Ethernet ASICs have remained the same for decades in terms of the features they offer [17]. The introduction of Barefoot Tofino and the programmability it offers increases network functionality enabling the connection of heterogeneous processors to the network [17]. However, the power consumption of such ASICs, when deployed in large scale could prove to be high for large networks.

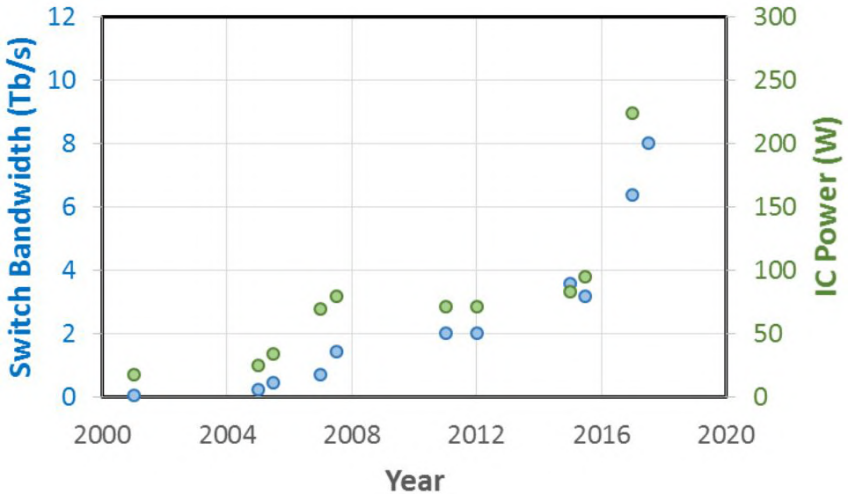


Figure 1.5: ASIC Bandwidth and Power consumption vs Year [4]

Switch ASIC bandwidth has been approximately doubling every two years and off-chip bandwidth is fueled by increasing per channel rates [4]. However, as shown by Fig 1.5, the power consumption of switches increases with switch bandwidth. The power consumption of 6.4 Tbps switch in 2017 is shown to have an overall IC Power of 230 W in Fig 1.5. This trend is expected to increase with more bandwidth as the bandwidth density within the IC will also increase.

1.1.3.2 Hyper-bursty era vs EPS

On another note, Cisco VNI also forecasts that 94% of workload and compute instances will be processed by cloud data centers [2]. Moreover, in bursty, cloud based applications, 90% of packets have a size of less than 576 bytes [3]; smaller packets require faster switching. Current Ethernet switches employ layer 2/3 switching methods like cut-through, store-and-forward or fragment free methods in order to forward 'frames' or 'packets'in a multi-hop fashion that eventually direct the flow to correct destinations. Each hop in a 10 GE takes in the order of tens of μs end-to-end latency for cut-through switching and hundreds of μs for store-and-forward switching [18]. The switch speed is dependent on the processing and queuing speed, frame size and the switch line rate, 10 GE in this case. Although fast cut-through and fragment-free are being designed with advanced switch ASICs, switching in layer 2 increases latency and degrades overall network and computing application performance. In addition to this, the increase of smaller packets in future data center networks implies the increase in queuing if the packets are not processed in high speeds. Hence, in order to operate in the nanosecond domain, switching has to be done in layer 1 and simpler protocols for switching must be devised to significantly reduce the latency.

1.2 Migration to optically switched networks

1.2.1 Motivation and Challenges

1.2.1.1 Attractive Optical Features

The migration to optical switches has been under research and discussion for the past two decades but it has never really deployed. The key motivation that drives this front is the inherent capacity enhancement offered by optical technology with WDM (frequency), modulation (phase), amplitude and polarization schemes. The transmission and reception at high bandwidth enables the exchange of heavy amount of data within nano-seconds is undeniably a covetous feature that today's optical technology offer. However, there are a few challenges which need to be addressed before migration to optical solutions.

1.2.1.2 Can Custom CMOS switch ASICs help optical switching?

Firstly, the drive to meet the ever growing traffic has been assisted by the CMOS technology growth, which was predicted by the co-founder of Intel, Gordon Moore, to diminish half in size every two years. However, the trend is approaching an end as we are bordering at the one of the lowest CMOS transistor sizes, 5nm. The development of electronic switch-based networks, server processors and accelerators are all slowing down as a result. How can migrating to optical switching

solutions help this situation? Well, data center networks have always benefited developments in the physical layer. I argue that the physical layer has not completely developed and tapped to be fully advantageous for optical solutions. A parallel electronic hardware-based scheduler design complementing an optical switch can provide great advantages with nano-speed circuit re-configurations or packet forwarding. Such scheduler designs can minimize switch latency by three to four orders of magnitude.

1.2.1.3 The big challenge

Another factor that has affected the adoption of optical switches in data center interconnects is complexity and the number of devices required in performing unique functions. Optical packet switching, especially, require additional components and controller mechanisms for wavelength conversion, optical buffering, re-transmission, optical/opto-electronic header processing. In addition to this, the existing layer 2-7 electronic routing algorithms and protocols that have been developed are compatible with existing traffic types and technologies for reliable end-to-end communication and functionality. Hence, the development of a novel physical architecture demands the transforming at all layers and levels of hierarchy.

1.2.1.4 Optical Network Motivation

The key drive behind moving to optical network and optical switching technology is to minimize latency that is being pertained in current EPS switches. As shown by Amazon's EC2 DCN, median latency lasts for up to 600 μs and tail latency can last up to 100 ms [19]. While the PULSE network proposed in the thesis creates a high bandwidth single-hop network and scalability and high-capacity systems are catered for, minimizing latency is the one main goal that PULSE aims to achieve.

1.2.2 Packet or Circuit?

If optical switching is the proposed way forward, an immediate question follows. What technology is going take up this big burden of driving data center networks forward - optical packet switches (OPS) or optical circuit switches (OCS)? However, it is important to answer this question in terms of the interest of the entire network in terms of efficiency, latency, scalability and performance. It is important to bear in mind that an ideal DCN architecture would consume low power and cost, be scalable to support tens of thousands of end-points, support heterogeneous IT computing resources, pertain low latency and achieve high throughput, which is tolerant to diverse types of traffic workloads. Listed below are the key differences between packet switching and circuit switching technology:

- Packet switched technology work on link or transport layer reliably in contrast

to circuit switching technology, which works on the physical layer.

- Packet switches usually work in-band requiring label processing, while circuit switches work out-of-band and process requests instead of packets.
- Packet switched technology require packet in-switch buffering while circuit switching techniques eliminate this need.
- Packet switching technology have the advantage of being compatible with current technology and protocols in contrast to circuit switching.
- Traditional circuit switches are slow and consume lower power compared to packet switching technology.
- Compared to circuit switching, packet switching technology requires complex addressing and packet management techniques.
- Packet switching technology also must deal with packet loss and retransmission, while circuit switching technology establish circuits and work with protocols that promise zero packet loss.

1.2.2.1 OPS

The issues of OPS technology are listed in the following hierarchical manner: lack of optical memory/buffer, architecture complexity, controller/scheduler complexity, label processing, addressing and packet management techniques (loss/retransmission), power hungry O/E and E/O conversions, complex replication of equivalent electronic technology. As already discussed, packet switches have inherent advantages of using existing communication and routing protocols with standardised layer 2/3 frame or packet structures. However, they suffer from long tail latency that degrade application performance, require large buffers or reliable packet loss management for the network and packet ordering to sustain heavy traffic. In addition to these features, they also to additional uni-, multi- and broadcast capabilities by replication, although not inherently available. Many of these features when translated to the optical domain require complex components, as discussed, and can prove hard to implement and scale. Hence, resolving to opt for optical packet switching technology result in a contribution of complexity both in the data and control plane. The optical packet switching technology can potentially pose a scalability, capacity and energy consumption disadvantage or challenge, if many components are required. Another disadvantage is the requirement of buffers in optical packet switching. Since optical buffers do not exist, researchers have

proposed the use of delay lines; however, the delay always remains static (propagation through the length of fibre) and there is no control of this delay. If not, optical packet switching require E/O conversion and O/E conversion before and after the electronic buffer employed. The power hungry conversion of optical to electronic and vice- versa could also prove disadvantageous in a optically packet switched network.

1.2.2.2 OCS

Meanwhile, there has also been development in the field of optical circuit switching. Circuit switching provide stable optical link establishments that can last from milliseconds to hours. They have been proposed to handle uniform non-bursty traffic in hybrid electro-optic interconnect (HELIOS [20], c-Through [21]) in order to reduce the number of core switches in networks and thereby, the cost. The limitation of optical circuit switches (OCSs) in such scenarios is the reconfiguration time. The technology used, MEMS or beam steering, is limited by the mechanical movement of mirrors or alignment of collimators to a few milliseconds. Hence, the assisted software-based switch configuration computation time, not being dominant, was not considered to be a limitation when lasting a few hundred microseconds or milliseconds. Although OCS technology opens up the possibility of exploiting capacity gains offered by optical technology, the long reconfiguration times, in both data and control fronts, creates poor performance to specific types of traffic. In order to resolve this problem with OCS technology, researchers have explored the possibility of speeding up data plane reconfiguration by resolving to free-space optical switching. RotorNet tackles this problem by proposing the Rotor switch, which has a faster reconfiguration time (data-plane) in the order of tens of microseconds and resolves to use a scheduler-less control-plane to reduce the configuration compute time. However, again, OCS methods are only shown to support a limited range of traffics or specific traffic types. The quality of service offered by RotorNet [22] and the tolerance it has to a dynamic and diverse traffic range in today's data centers that host diverse applications is questionable. In addition to this, the lengthy configuration time, tens of microseconds, of the Rotor switch is considerably slow in a Gbps network. So, there is a need for nanosecond speed circuit switching technology and a second need for nanosecond speed control, if the effective throughput must be very high and the latency must be ultra-low and deterministic. Hence, the requirements are identified.

1.3 Towards ultra-fast OCS Networks

1.3.1 OCS Switching Technology

First, as the requirement specifies, is the need for fast circuit switching technology. The channels provided by WDM can be potentially used for routing and establishing circuits. Although this implies that WDM capacity enhancement is removed, there are other capacity enhancing techniques still to exploit in the optical regime. Moreover, optical wavelength switching has been developed and it has come a long way with a potential to enable wavelength selection at both the transmitter and the receiver at nanosecond timescales. Furthermore, wavelength switching and routing was explored decades ago for metro networks and are still being proposed in order to provide reliable and fast data exchange. Tunable SG-DBR laser or widely tunable DS-DBR lasers have been proved capable of selecting wavelengths within nanoseconds. Exploiting wavelength switching technology to aid and favour circuit switching can improve and challenge novel OCS technology by providing ultra-fast network reconfiguration. Another component that could prove advantageous is SOA, an optical gate proven to have control in sub-nanoseconds. In addition, SOAs also provide gain advantages but at the expense of higher energy consumption. Hence, a power-balanced optical transceiver technology with a fast switching capability is required. In this thesis, I propose PULSE, a fast OCS network architecture that employs one of the listed transceiver architecture options for switching and I analyse the network energy they consume.

1.3.2 OCS Control technology

1.3.2.1 The NP-hard resouce allocation problem

Firstly, it is important to notice that the task of resource allocation is a complex one. Each node (server or ToR) in the network can send multiple requests requesting a diverse range of network resources in a given time. A demand matrix is created from these requests, a large one, requires an optimal matching of wavelengths and timeslots in order to provide maximum or high throughput. If fast allocation is done, but matching is poor, then the algorithm can perform data center application degradation.

Secondly, the computation of resource allocation is required to be done within nanoseconds. Serial techniques or software techniques of binning or a sorting algorithm based on size of each request in correspondence with the resource map available provides significant advantages. In such algorithms, a clear idea of resource allocation evolution allows the software to examine and introspect the next allocation. However, these approaches take long computation times, even if they do

provide optimal matching. Hence, this mathematical problem is NP-hard and it is impossible to perform in the speed required using software methods.

1.3.2.2 Why software algorithms fail?

A major challenge in optical switching in general, packet and circuit, is the design of the controller or the scheduler. Traditionally, circuit switching control has been software-based. However, these techniques take in the order of milliseconds to provide an optimal solution or configuration. As highlighted, the aim is to contain control and data plane of the OCS technology within a few nanoseconds. Software technology simply can never deliver such a reconfiguration task within the proposed timescales. Going scheduler-less, like in the RotorNet scenario, cannot cater for diverse workloads in DCNs. Hence, there is a requirement for a scheduler, but faster than a software-based method. Thus, the design, synthesis and implementation of a custom switch ASIC that can analyse a request demand and compute a wavelength-timeslot map for all transceivers in a network within nanoseconds would be ideal.

1.3.2.3 From Software to Hardware

Custom hardware switch ASIC design can provide highly advantageous solutions but as hinted, the requirement is three-fold. Firstly, the matching performance must be high along with being tolerant to diverse traffic workloads. Secondly, resource allocation has to be performed within nanoseconds and thirdly, the algorithm must also be scalable. Although many of the advantages offered by software algorithms are not evident in custom hardware electronics, the parallelism offered by hardware digital designs can be exploited to our advantage in order to provide high throughput. However, it is important to be careful not to make contradicting assignments in parallel. Next, the scheduler is on a clock; it has to perform multiple tasks within a given time. But since the task is parallelised, pipelined and distributed, the scheduler behaves as a multi-core network scheduler processor that performing multiple threads in parallel, enabling computation to be done in a few nanoseconds. Thirdly, the scalability of the controller is also crucial. The faster the scheduler, the more iterations it can perform in order to provide an improved matching with high throughput. Although I have used the open-source 45nm NanGate library and Synopsys synthesis tools to synthesize the hardware design on 45nm ASIC, the proposed solution can be translated to better CMOS technology with improved results and at a fast clock, extending the scalability of the network.

1.3.3 CDR and Synchronization

The fast OCS network also requires timeslot-level clock and data recovery (CDR) and synchronization across all nodes in the network for correct operation. This

thesis does not cover the scope of CDR but this work is carried out by research team in UCL [3] and successful experimental phase caching techniques have been demonstrated. I identify research work already performed by researchers to showcase synchronization or the building of a scalable clock network. However, they are not explored in detail in this thesis. Hence, I assume that this synchronization is available and is present across all end-nodes in the network.

1.3.4 Solving Key Problems

In this sub-section, we aim to list the main problems that our proposed OCS network and control algorithm aims to solve. Current OCS switching technology are shown to establish paths that lasts from milliseconds to hours [23]. Due to their slow reconfiguration and circuit computation, they are usually employed for operating uniform long flows, while bursty traffic is handled by an electronic packet switch (EPS) [20, 21]. In other words, hybrid networks are employed simply because OCS technology is not suitable to handle all types of traffic. Hence, the vision of PULSE, our proposed ultra-fast OCS technology and control, is to remove the need for EPS in future DCNs and make an all optical pure OCS network. While this vision is the motivation behind PULSE, we also aim to achieve other points of the ideal DCN architecture described in section 1.2.2. In addition, I would like to make it clear that this thesis does not aim to solve all the problems pertaining to PULSE but aims to create the ultra-fast OCS network and mainly design the control aspects in hardware that is one of the key challenges in enabling fast OCS reconfiguration computation.

1.4 Thesis Structure

1.4.1 Overview of Chapters

The thesis is structured in the following manner: the second chapter contains a detailed literature review. Starting with the need for migration from electronic to optical switching, the complexity involved with (electronic and optical) packet switching are highlighted. Following this, current state-of-the-art optical circuit switching technology and their limitations are explained. As I delve into architectures as well, the relevant network architectures that have been proposed previously are brought to attention. Finally, previous work on scheduling algorithms and their implementations are studied to show the novelty of this work, by comparison. In this chapter, I aim to identify the research gap that exists and find the place PULSE correctly amidst other fast growing technology.

In the third chapter, I begin with the original architectural idea that started with our SIGCOMM paper [24]. The requirement of a hardware designed scheduler and the digital architecture of the scheduler and its components are highlighted. An im-

proved and developed stable hardware scheduling algorithm and its corresponding hardware architecture are analysed to identify the bottlenecks at each stage. Following a brief performance analysis, upgrades that are required in all fronts i.e. transceivers, architecture, scheduler are identified as areas to improve. After understanding the cons, the pros of the scheduler are highlighted, as I show it to be capable of achieving a high throughput due to its performance gain in just a few iterations; it can support multiple networks.

In chapter 4, the proposed hardware based wavelength scheduling technique that adapts parallelism is compared with software based wavelength assignment heuristics. As equitable performance results are found, a detailed analysis of resource usage, iteration effects, request count per node, wavelength usage are analysed. Here, I aim to show the network concerns that majorly limit our proposed OCS and how they affect throughput, scalability and latency.

In chapter 5, I address the network concerns and propose a novel network architecture called PULSE with nanosecond speed switching potential. The modifications to the scheduler are highlighted, the scalability of the hardware is maintained and data plane scalability is reviewed. The proposed transceiver architecture that enables ultra-fast switching is discussed and the requirements of the network are explained. The PULSE OCS network is characterized in terms of latency overhead, scalability, power consumption and cost.

In chapter 6, I evaluate the overall performance of scheduling in PULSE and show how PULSE is able to achieve low, determistic latency with high sustained throughput under diverse traffic workloads. Highlighting the ability to maximize throughput and wavelength usage, PULSE can also support scalable solutions while consuming low median and tail latency. Some key questions that answer how scalability in terms of rack size and multi-plane architecture affect scheduling are also highlighted. A summary of the benefits of ultra-speed OCS switching and its potential in future optical interconnects are highlighted.

Finally, I conclude in chapter 7 with a detailed summary of the list of things achieved during the course of this PhD and how this has contributed to research outcomes. I show the room that has been created for further research - future of PULSE and show what improvements can be made in order to boost the network. I discuss how PULSE's hardware solution can be employed for variant OCS switch architectures and improved. I also propose scalability enhancements, routing techniques for scaling to more end-points or capacity. As concluding remarks, I attempt to link back to the motivation of the work and show a platform is created for more practical and theoretical research.

1.4.2 Original Contributions

The concept of building a passive star-coupler with tunable transceivers was initiated by UCL in collaboration with Microsoft Research in [24], initially to scale to large port count flat network. However, this concept was very rudimental and had a low network efficiency of 8% with enormous wastage of resources. But to demonstrate proof of concept, the data plane physical layer (optical) with the aid of tunable DS-DBR and coherent receiver combination was investigated to support a larger split [25]. However, the network concepts in terms of efficiency, performance and scalability were not elaborated. Concepts about transceivers were vague, as tunability was shown to be achieved under 200 ns, but epochs were made to last for 2μs; the power, cost, resource utilization efficiency were also left unexplored. Crucially, the scheduling techniques and the nature of the controller, that defines and reconfigures the entire network, was missing. Without performance efficient ultra-fast nanosecond speed scheduling, there is no actual use to the network no matter how fast or efficient the network or the transceiver switching is. In this PhD research and thesis, I address these unexplored territories in the following manner.

First, the scheduler design and hardware architecture is the most crucial work in this work. The NP-hard problem of finding maximal matching within a few iterations requires both spatial and temporal parallelism. Hence, as parallelism introduces contention, I explored the scalability of the round-robin arbiters to 1000-ports and explored there possibility for use in multi-core processing scheduler. Once identified to be scalable, I identified other bottlenecks that existed in the digital hardware design that created dependencies. By critically evaluating the scalability of each module and removing dependencies, I created a design, in which, the critical path length lies in the arbiter. I synthesized the hardware design on 45 nm CMOS ASIC to show that a parallel hardware can be designed to solve the NP-hard problem in a matter of a few nanoseconds. To ensure performance, I created a simulation testbed that evaluated the performance and verified the design of the synthesized hardware with MATLAB and Modelsim. Once a 100% match was achieved when compared to an equivalent software model, I compared it with other software based serial wavelength assignment techniques. In addition to this, I identified other network bottlenecks that cause scheduling limitations and degrade overall efficiency. I addressed these limitations to unlock and maximize the potential of fast optical circuit scheduling. In the process, I have engineered a hardware synthesizable solution, a hardware-software simulation environment that with equivalent performance shows the potential of hardware implementation for servicing a wavelength-timeslot switched network with maximal performance in terms of throughput, resource utilization and minimal latency, almost 3 orders of magnitude lower than state-of-the-

art electronic switching technologies.

In addition to this, I have introduced a novel scalable architecture that can scale to 10,000-100,000 blades by using distributive scheduling and minimizing the pressure on scalability. The novel OCS architecture, called PULSE, uses independent modular sub-networks and it is also capable of supporting large capacities and potentially aid future heterogeneous data center networks. In contrast to the original architecture, I have proposed faster SOA assisted transceivers in order to enable switching at the time-slot level rather than at the epoch-level. In addition to adjusting and re-designing the scheduler to equip this feature, I have proposed different transceiver architectures that can potential aid ultra-fast OCS systems. I have investigated the power and cost of such transceiver-based networks per path and their impact on the overall network relative to that of state-of-the-art electronic switching solutions. As scalability was dependent on the number of transceivers, I also investigated scaling with novel techniques that scale and at the same time, reduce dependency on transceivers.

1.4.3 Research Outcomes

This PhD research has, first, created a novel simulation environment and testbed that verifies hardware functionality and performance. The simulation environment can be extended and the hardware modified accordingly while making sure parallelism is unaffected to create smart hardware-based techniques that have the potential to reconfigure optical circuit switches in the sub-microsecond time domain. The simulation platform has more room for research, development and optimization before finalizing on efficient hardware design. Also, going for faster hardware CMOS technology can help to reach even higher speed and lower cost and power with respect to ASIC designs. As a ground level proof of concept, the initial synthesis gives encouraging indication that careful parallel hardware designs can solve NP-hard problems and achieve nanosecond speed computation.

Another exciting research area to explore is for DS-DBR tunable lasers to potentially achieve faster tuning time and support more wavelengths by extending to other bands. On the other hand, fast switching SOAs with low power consumption in coordination with these transceivers can be explored for fast wavelength switched networks. Extending wavelength resources or employing the use of modulation formats or space division multiplexing can also be explored for fast OCS systems, as this do not affect the concept of the network or the scheduling. Small scale synchronization and source synchronous techniques can be tested to support the proposed network architecture. Other control and data plane improvements can be practically invested on to further advance the research scope of PULSE.

In terms of publications, the research has been presented to a wide variety of top technical research audiences from computer communication conferences like ACM SIGCOMM, IEEE/ACM HotInterconnects to top optical conferences like PSC, ECOC and OFC (Invited talk, 2020). The work is under review for publication in Journal of Lightwave Technology's Special Issue on Optical Interconnect 2020. Being still in its initial stages, the research has the potential to advance further in both theoretical and practical aspects.

1.4.4 Publications arising from thesis work

The following are the publications that arise from this PhD work and they are listed in reverse chronological order:

[1] **J. L. Benjamin** and G. Zervas, "Scaling PULSE Data Center Network Architecture and Scheduling Optical Circuits in Sub-microseconds," 2020 Optical Fiber Communications Conference and Exhibition (OFC), San Diego [Invited Talk]

[2] **J. L. Benjamin** T. Gerard, D. Lavery, P. Bayvel and G. Zervas, "PULSE: Optical Circuit Switched Data Center Architecture Operating at Nanosecond Timescales," Journal of Lightwave Technology Special Issue on Optical Interconnects 2020 [Under Review]

[3] **J. L. Benjamin**, T. Gerard, P. Bayvel and G. Zervas, "PULSE: Scalability of a sub-μs wavelength-timeslot based circuit switched Data Center Network," European Conference on Optical Communications (ECOC), Dublin, Ireland, 2019

[4] G. Zervas and **J. L. Benjamin**, "PULSE: Sub-microsecond Optical Circuit Switched Data Center Network," 2019 24th OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC), Fukuoka, Japan, 2019, pp. 1-3

[5] **J. L. Benjamin** and G. Zervas, "Parallel Star-coupler OCS Architectures using Distributed Hardware Schedulers," 2018 Photonics in Switching and Computing (PSC), Limassol, Cyprus, 2018, pp. 1-3.

[6] **J. L. Benjamin**, A. Funnell, P. M. Watts and B. Thomsen, "A High Speed Hardware Scheduler for 1000-Port Optical Packet Switches to Enable Scalable Data Centers," 2017 IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI), Santa Clara, CA, 2017, pp. 41-48.

[7] A. Funnell, **J. Benjamin**, H. Ballani, P. Costa, P. Watts and B. C. Thomsen,

”High port count hybrid wavelength switched TDMA (WS-TDMA) optical switch for data centers,” 2016 Optical Fiber Communications Conference and Exhibition (OFC), Anaheim, CA, 2016, pp. 1-3

[8] D. Alistarh, H. Ballani, P. Costa, A. Funnell, **J. Benjamin**, P. Watts, and B. Thomsen. 2015. “A High-Radix, Low-Latency Optical Switch for Data Centers”. SIGCOMM Comput. Commun. Rev. 45, 4 (August 2015), 367-368.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, I will attempt to expose and highlight some of the fundamental limitations of electronic packet switched interconnects that pose a threat to scalability and power consumption, as we approach the 'capacity crunch'. The challenges and bottlenecks of the both the switching elements and the network are indicated to propose the need for architectural change. I identify and justify the need for the adoption of optical technology in order to support the sheer magnitude of data that is expected flow in future data center networks.

Packet switching has dominated data centers and have sustained the transport of traffic for the past two decades. Whichever optical switching technology is to be adopted, circuit or packet, both have their share of challenges to implement. They must also compete with standardized electronic packet switches and networks that have been established and developed over the last three decades. Hence, in this chapter, I aim to recognize and weigh the various challenges present within optical packet and circuit switches and interconnects to make an educated decision for the adoption of optical technology and the transformation of present day data center networks. I present evidence to argue that when transitioning and migrating to optical technology, ultra-fast circuit switches are simpler to adopt, scalable, compatible and have a higher degree of performance compared to optical packet switches. Although future heterogeneous data center networks may potentially adapt hybrid packet-circuit switching technology, hybrid networks increase operational complexity and they also need to be accurately dimensioned as two networks are managed. Hence, the focus of this study and thesis is purely on the side of circuit switched networks.

The motivation for advancing towards circuit switched technology are scalability, complexity, power consumption, throughput and latency. However, having its share of challenges, present day OCSs are also limited by reconfiguration time

and hence, latency; this emphasises the need for ultra-fast OCS technology. The stance of PULSE is shown, the ultra-fast OCS technology proposed in this thesis, in relation to the state-of-the-art OCS.

2.2 Electronic Packet Switches

For the sake of cloud service and information growth sustenance, the last two decades have witnessed the assisting growth of data center interconnects along with the growth of traffic. The size of data centers have grown to tens to hundreds of thousands of servers. Along with size, network and switch complexity have also grown undesirably. In order to continue scaling data center interconnects, switch elements have to be significantly simplified [26].

Hence, researchers have proposed the simplification of the network fabric with simple *cell* switch elements to open up scaling opportunities. Traditionally, regardless of switch type (store-and-forward, cut-through or fragment free) and packet size (64 or 1500 bytes), all packets experience ingress, egress processing, scheduling, switching and queuing. In Stardust, a relatively passive network fabric that eliminates in-switch ingress, egress processing, scheduling and queuing is proposed in order to minimize switch resources for better performance [26]. Also, research has shown that, apart from complexity, there are concerns with power consumption and costs and limitations in terms of scalability with electronic packet switches.

In this section, a case is proposed that electronic switching technology are limited in various aspects that degrade power, cost, latency and throughput efficient scalability. Moreover, as suggested by researches for more than three decades now, deploying optical technology can potentially open up a new realm of possibilities (in terms of capacity, power consumption, throughput and latency).

2.2.1 Switch Radix

Traditionally, Data Center networks have employed low-radix (24 to 64-port) network electronic switches in order to interconnect an entirety of tens of thousands of servers. However, as the server count is much larger than the switch port count, a hierarchy of switches with multiple levels have to be deployed to provide server access to all end-points. Moreover, these hierarchical multi-layer architectures create bandwidth bottlenecks, high end-to-end latency and poor cost efficiency [27]. Hence, high radix switches reduce switch and hop count, which significantly decreases component cost, latency and power consumption [28]. In [29], a clear comparison of the use of 4-port switch and a 256-port radix switches in a folded Clos topology are shown. This architecture requires about 1,000,000 4-port switches but only 1,000 256-port switches in order to interconnect 100,000 servers or end-points.

The research also highlights that the latency through 256-port switch is more than 2-3x lower compared to the 4-port switch. Although the advantages of large radix switches are obvious, ASIC chip I/O bandwidth, more on this later, and power budget create port-count scaling limits in current electronic network switches [28].

This problem continues to exist even when migrating to optical switching technology. Optical circuit switches with slow reconfiguration time, using simpler software-based controller, scale to large port counts (100 ports) [30] and HUBER-SUHNER Polatis 384-port OCS is commercially available [23]. Large switch reconfiguration times lead to long tail latency and degrade computational performance. However, scaling low-latency optical switches to high radii is also a challenge chiefly because of controller complexity and the time it requires to compute switch configuration [31]. Hence, aiming to reduce the switching time while maintaining scalability, semiconductor optical amplifier (SOA) based network switches have been proposed and feasibility of a 128-port using 2×2 hybrid SOA-MZI switch module has been demonstrated [32]. This technology requires multiple SOA elements and it is shown to introduce a power penalty of 2.4 dB at 10Gb/s. Requiring N^2 components, this technology consumes large amount of power and introduces complexity in both the data control plane computation, which now needs to compute routing as well as port allocation. The complementary work on a SOA crossbar architecture is shown to consume an end-to-end latency of 82.4 ns, while scaling to 32-ports [33]. Another high-radix optical switch is showcased in the research on 512×512-port Arrayed Waveguide Router (AWGR) with 25 GHz channel spacing, which has a -4 dB crosstalk [34]. The AWGR switching technology requires transceivers or tunable wavelength converters that selecton of 512 wavelength channels with the ability to coarse and fine tuning. This wavelength selection process also contributes to the additional switching time. Moreover, the AWGR switch matrix also requires a control plane to dictate configuration in the order of nanoseconds for ultra-low latency.

2.2.2 ASIC limits

Electronic switch ASIC port-count and bandwidth scaling is limited by the Ball Grid Array (BGA) package and the I/O pins. Devices packages are big in size, $55mm \times 55mm$ or more, bigger than standard high-end CPUs [35]. High-end Ethernet switch ASICs can support 16 to 128 ports with a flexible bandwidth of 400 GBE to 50 GBE, depending on network port link bundle [26]. The large switch IC packages raise concerns about flexibility, cost, power consumption and reliability. Hence, any high-end Ethernet switch ASIC is limited by the overall bandwidth operable by the switch; in the case of [35], this is equal to ($16 \times 400G$ or $128 \times 50G$ =)

6.4 Tbps. Less number of high-bandwidth switches are often implemented at the higher layer (core or aggregate) of a fat-tree or multi-layer topology due to their expensive nature. On the other hand, lower layers (ToR and Edge) in the network architecture are often low bandwidth switches with 10 GBE (or 25 GBE).

Similar high-bandwidth ethernet network switches are Barefoot Tofino 2 [36], Broadcom Tomahawk 3 [37], Trident 4 [38] and Jericho 2 [39], which suffer from bandwidth limits at 12.8 Tbps or less. These type of Ethernet switch ASICs exploit a 7nm CMOS technology and a dense PAM4 SERDES. As switch ASICs are border lining the saturation limits of Moore's law in terms of power consumption, the scaling of I/O pins beyond 12.8 Tbps will not continue to be at the same rate [26] while maintaining flexibility, cost, power consumption and reliability.

In optical switching technology, more often, the switching is done out-of-band to the I/O transmission or reception. Although keeping communication out-of-band demands synchronization and/or a control-data handshake, the amount of resources required in switching is significantly reduced. In such cases, if an electronic switch ASIC is designed for an optical switch, a room for improvement and scalability exists in addition to the capacity scaling offered by optical transceivers. In other words, the de-coupling of TX/RX I/O bandwidth to the control engine creates a new realm of scalability by keeping resource utilization to a minimum. Hence, ASIC implemented hardware-based control for fast optical switching are a great way forward for implementing optical data center switching technology. The resources can be maximized for computing better and improved routing or switching and traffic prediction, estimation and analysis.

2.2.3 Power Consumption

Chip designers are always aware of limits posed by the die size and power consumption when engineering a high-power network switch ASIC. The study of power consumption in [40] estimates that the power consumption of 24-port 10GE Ethernet switch [41] is about 37 nJ/bit or 238 W. In [35], the I/O ports (32-port $\times$ 200 GBE) are identified to be the dominant energy consumers in the high-bandwidth switch (6.4 Tbps) and is found to be 224 W, which is relatively low for the bandwidth it provides. Novel switch chip architectures and the benefits of advancing CMOS technology (Moore's law) previously aided maintaining or lowering Ethernet switch ASIC power consumption. However, the extremely high bandwidth I/Os and the coupling of complex functional hardware for buffering, queuing, processing, scheduling fundamentally consume a minimum power for a given CMOS technology. Moreover, as Moore's law is reaching it's limit, CMOS technology cannot continue to aid in minimizing power consumption. From the overall data center

interconnect perspective, the requirement of switches in order to interconnect more and more servers (end-points) is continually growing and this means that the overall network power consumption will also continuously grow within a DCN, as servers (and hence, switches) are added.

As power consumption is a major concern for current data center networks, optical solutions have been sought for the significantly lower power they consume. Most optical switching technology deal with out-of-band communication, where the data plane communication that includes I/O, transceiver power consumption is independent of control plane communication. In particular, the study in [42] (2011) has been particular interest in the low power consuming optical packet switching fabrics to employ them replacing electronic switch fabrics. Although there have been considerable improvements in both technologies over time, the research shows SOAs, micro-rings and AWGs switch fabrics as increasing in order of power consumption. Note that this does not include the control plane or the transceivers. In [43], the transceiver and SERDES for an optically switched network is shown to consume lower power than NRZ signals by using PAM-16 modulation formats and WDM signal at 25 Gbps is shown to consume under 30 mW. The research in [42] shows that as the data rate increases to tens of Gbps, optical switching technology tend to become more energy efficient. However, as concluding remarks, this study of power consumption states that ultimately, the major power consumers of optical packet switching technology are optic-to-electronic and electronic-to-optic converters, which are essential for header processing or header replacement.

Ultra-low MEMS based optical circuit switching fabrics have also been proposed and shown to consume several orders of magnitude lower power than conventional switching technology [44]. HELIOS is an hybrid electro-optic architecture that utilizes both electronic and optical circuit switching technology to get the benefits of both worlds. The key motivation of HELIOS [20] is to reduce a number of electronic packet switches in a 'sweet-spot' layer (preferably core) by replacing them with single low-power optical circuit switch. The use of such hybrid solutions reduce power consumption significantly and hence, commercially available large-port count circuit switches have been employed to form hybrid networks [23]. However, the use of such slow-speed OCSs only perform efficiently for specific types of uniform, non-bursty traffic. Hence, in order to maximize performance benefits, a traffic predictor or estimator must schedule the load efficiently dictating the use of EPS or OCS switch. Other non-hybrid circuit switching technologies have been under research; these include some architectures that have reconfiguration timelines in microsecond (RotorNet [22], Mordia [45] and ReacToR [46]) or milliseconds (Firefly [47] and OSA [48]). Although the low power consumption of these switching

scheme prove attractive, the long reconfiguration time makes them suitable only for particular types of networks or traffic. In other words, in present-day heterogenous DCN environment with diverse workloads, processes, tasks and applications, employing such circuit switching architectures will degrade performance and quality of service.

2.2.4 Performance: Latency and Throughput

Data centers host many types of applications and each application causes a different unit to be intensely used: CPU, memory or accelerator. In fact, the order of variance between CPU and memory intense applications (or the CPU-MEM utilization ratio) varies in 4 orders of magnitude [49]. Naturally, this creates a diverse range of traffic patterns. In addition to this, the type of data center network also plays a role in the application diversity: Cloud, Enterprise, Education or other types [50]. Hence, network engineers are keen to create a network architecture that can maintain high-performance and is relatively tolerant to traffic diversity. While performance is primarily defined by latency and throughput, the tolerance of a network to diverse workloads and traffic loads also matter.

Applications in Ethernet-based networks suffer from tail latency in several order of tens to hundreds of milliseconds. Application to Amazon's EC2 data center consumes a median and tail latency of 600 μs and 100 ms respectively [19].

2.2.5 Network Construction

Each Ethernet network switch supports 48 or 64 ports, which is two to three orders of magnitude lower than the number of servers in a cloud DCN. Hence, multiple switches are required to connect an entire network. In addition to this, in order to ensure speed and reduced bandwidth, each level in the network (edge, aggregate or core) has a different grade switch. Traditionally, electronic network switches (edge, aggregate and core) are interconnected to form a folded-clos or tree-based topology, which provide a full bisection bandwidth; however, such interconnects would face complexity and cost issues when forced to scale as the number of servers, N, increases. Even though this type of network gives an all-to-all communication access, not all servers communicate to each other in the same time window.

A data center network traffic characteristics analysis in [50] shows that the link utilization leading to the expensive core switches (core links) never exceeds 25% usage in any data center. So, when a data center is built, some degree of over-subscription is introduced to the folded-clos to form a multi-rooted tree topology. What this implies is that network switches are over-designed yet under-utilized, especially when under the influence of bursty, small-packet traffic.

Over-subscription allows data centers to ease (higher layer) links that have high

bandwidth tension without compromising the quality of service significantly. This enables data centers to scale more effectively and support more servers. However, such topologies result in bandwidth fragmentation, with more bandwidth available within lower level of the tree than higher levels. This results in network performance variation, where the quality of service varies a lot depending the locality and skew of the data center traffic matrix. This skew exists only in oversubscribed multi-tiered electronic networks. In contrast to this, an oversubscription in an optically switched network can create a sharable pool of bandwidth fairly available to all nodes in each sub-network and hence, the skew does not exist.

2.2.6 Benefits of Optical Switching

To summarize, the hyper growth of data demand implies that data center network growth is inevitable. Ethernet switches cannot continue to grow at the same rate as the we approach the end of Moore's law. As discussed in this section, Ethernet switches are limited by I/O pins that an ASIC ball grid array (BGA) and the bandwidth each pin can support. Moreover, high bandwidth switch ASICs consume high amounts of power and deployment/running costs. The applications within a data center are also limited by the network built by the Ethernet switches with latency lasting in the order of milliseconds.

Optical switches can de-couple the data plane I/O communication from the control and scheduling aspects of a switch to unlock the dependency of transmission capacities on switch ASICs limits. Optical technology can, hence, reduce the power consumption and the cost of the network significantly. Optical switching technology can prove to be game-changers in future novel data center interconnects, creating traffic agnostic topology with ultra-low latency. However, there are some basic requirements to create an optically switched network with ultra-low latency, cost and power consumption. Moreover, optical technology possess inherent capability to enhance transmission and reception capacity.

2.3 Optical Packet Switching (OPS)

Although optical switching technology can offer many advantages as described above, they still have to meet particular requirements to achieve the proposed benefits. There are some challenges and concerns that exist with migrating immediately to optical switching. Optical buffering/memory is still in its early stages of research and hence, electronic-to-optic and optic-to-electronic converters are used to include electronic buffers into the switching fabric. These converter units are power hungry and expensive. Another concern is that the advanced features available in current electronic switching technology could prove difficult to implement in optical

switching.

A major question to be answered in terms of migrating to an optically switched network. It is important to identify if optical packet switch or circuit switch the best way to moving forward. The advantages and concerns of each technology must be weighed to understand which will give the maximum benefits with minimum implications on cost and power. The aim, here, is to evaluate the advantages and the disadvantages of both the optical packet switching and circuit switching worlds. Once anaylsed, the best technology for minimizing latency, power and cost is exploited for design optimization, research and development.

When the concerns of each switching technology is highlighted, circuit switching with fast dynamic path establishment can prove to create relatively lower latency, power and cost consumption with scalable hardware controllers. The main reason behind this is the complexities that come into play when employing optical packet switching technology, which are highlighted in this section.

2.3.1 Concern 1: Controller Complexity

Several optical packet switching technology have been under study for replacing current electronic packet switches by focusing the boost of performance in particular aspects. Nevertheless, extensively, there have been fundamental challenges and concerns that impede optical switches from being adopted. Firstly, a majority of optical switch architectures, packet or circuit, employ a centralized controller or scheduler. In fact, the research in [31] identifies controller complexity as the most critical factor that limits switch scalability. In this part of the chapter, I intend to highlight how the hardware controller complexity have affect optical packet switching technology.

For example, a high performance computing optical network architecture called OSMOSIS exploits a broadcast plane of star couplers followed by a select plane of high-speed fiber and wavelength selection (using SOAs) for creating a crossbar fast optical packet switch [51, 52]. Buffering is managed at the ingress and egress adapters, avoiding packet loss across the switch fabric. The power, cost and scalability of OSMOSIS is also a major concern due to the lossy nature of the star-coupler and the ample number of SOAs and EDFAs the scheme employs. However, the major challenge in this network lies in the design of the central scheduler [53]. In [54], the feasibility of OSMOSIS is questioned due to the hardware complexity of the required bipartite matching algorithm implementation. The complexity of a computing problem defines the time it takes to find an ideal or a high-throughput solution; however, OSMOSIS requires this computation to be done but within a few nanoseconds. Bipartite matching algorithms are known for their complex nature

and they become an NP-hard problem as OSMOSIS scales to large port numbers. Hence, this low-latency, low-power solution is limited by the scalability of its control plane.

The Data Center Optical Switch (DOS) architecture uses a simple cyclic AWGR with label extractors, fiber delay lines and tunable wavelength converters in the data path. The label extractors are used by a centralized controller to read the header to identify the destination of interest and use groups of arbiters to control the relevant tunable wavelength converters that have access to the corresponding destination. DOS also hosts a shared buffer in parallel to the AWGR, where the failed requests (the requests without a path to the destination) are stored. The photo-diodes in the DOS architecture limit the scalability of DOS architecture, which is handled by grouping wavelengths. However, a major concern with DOS is the scalability of the control plane. The DOS architecture assumes a 2 GHZ clock speed for the controller regardless of the size of round-robin arbiters. However, as highlighted by chapter 3 and [53], arbiter clock period increases as it scales. Hence, the assumption of a 2 GHz clock period for the controller is unreasonably high, creating a very slow non-scalable optical switch system.

Petabit is an optical switch that employs a multi-stage cyclic AWG routers, tunable lasers and tunable wavelength converters to create an optical packet switching technology [54]. Moreover, a multi-chip distributed scheduling architecture (probem is broken down) is proposed in order to manage the bandwidth and produce a highly scalable control plane. Each scheduler chip employs request filters and sorters and multiple iterations (up to 3) are performed to offer speedup, efficient throughput and low latency. The three stage architectures wavelength selection at three places (IM, CM and OM) in order for the packets or frames to reach the correct destination. However, the scalability of Petabit depends on the scalability of the 128-port sorting hardware. The implementation of sorting algorithm is an NP-hard problem; the implementation on custom hardwares results in large clock cycles and requiring many iterations, the computation of a full-sort takes several milliseconds to seconds.

Optical circuit switches suffer from complex controllers too. However, the magnitude of functional requirement of an OPS is much higher compared to that of an OCS. Optical switch controllers are required to perform more than just scheduling. They have to deal with complex functions like congestion control, buffer management, packet parsing for header field extraction, routing, packet replication for multi- and broad-casting capabilities. In contrast, OCSs are required to compute and establish paths while provide maximal matching.

2.3.2 Concern 2: Latency implications

Some optical switching technology cannot be employed due to their poor performance in terms of latency. Traditionally, circuit switching technology have slow circuit reconfiguration times, which make them only suitable for limited traffic types. Some optical packet switches (OPSs), competing with current electronic packet switches, also suffer from long tail latencies. Here, switch architectures that suffer from high latency values are identified. In order to reduce this latency, optical packet switches either employ large buffers/virtual output queues or introduce complex network components in order to decrease contention and hence, latency. If optical switches employ a buffering scheme, as optical buffers do not exist, electronic buffers and E/O or O/E converters have to be used. It is important to identify that the latency of the optical switching schemes are very much dependent on the efficiency of the controllers they use and it has a direct impact on latency (median and tail) and buffer sizes, which degrade computational application performance.

The DOS architecture uses arbiters in the hardware controller. As discussed, the assumption of a 2 GHz clock is unreasonable as the arbiter hardware cannot support such high speeds. Using the round-robin arbiters proposed in Chapter 3 and using accurate scaling equations in [55], a clock period of 1.1 ns can be assumed using a 7nm CMOS ASIC (a best case). Assuming a pipelined architecture, the best and worst-case average end-to-end latency at 90% input load for a 256 byte to arrive at the destination is 1.98 μs to 40.48 μs respectively. The same measurement for best and worst-case 4096 byte is 14.85 μs to 475.2 μs respectively. The tail latency for the DOS system will be several milliseconds, as the average packet delay itself is in the order of tens to hundreds of microseconds. DOS also uses an electronic buffer, requiring power hungry O/E and E/O converters. Long tail latencies could results in the requirement of a larger buffer at the switch.

In the 64-port POTORI (coupler based) switch, a centralized and tailored MAC protocol uses Largest First (LF) and iSLIP scheduling heuristics, which were shown to incur a latency of 10ms above 80% workload [56]. Although data buffering happens in a distributed fashion at the source server, long tail latencies can require large buffers, making this scheme expensive or impractical in terms of implementation. In Petabit (AWG based) architecture, the c-MAC scheme is used to control a 64-port AWGR and the estimated average latency was 5μs for offered network loads above 70% [54].

In Petastar [57], latency is reduced with the use of ingress and egresss adapters containing a distributed queues VOQs and VIQs for reducing overall latency. Moreover, the Central Module (CM) in the Clos-based photonic switching fabric (PSF) is increased 1.5$\times$ in order to provide speed up and reduce contention. Hence, this

increases the implementation complexity of the system, requiring more components in order to reduce latency. The Petastar architecture in [57] uses a concurrent matching algorithm for the clos (c-MAC) based photonic switch fabric, which is based on a exhaustive deal round-robin matching scheme (EDRRM) and finding routing paths for specific I/Os. In this packet switched scheme, an overall average latency at 100% load is 10-35 μs is experienced with the scheduling latency of 2.6 μs. However, a high amount of resources are used ot achieve such latencies and increased network throughput.

2.3.3 Concern 3: Complexity and Scalability

Hipoλaos [58] is an optical packet switch targeted for disaggregated data centre network applications where attaining high throughputs at low-latency is critically important. The 3-stage, N-plane switch architecture is controlled by an FPGA controller per plane. The first two stages involve the use of a combination of multi-wavelength light sources, AWGs, SOA-MZI based fast wavelength converters (WCs) and delay lines to manage packet ordering and buffering. The third stage uses a wavelength light source, WCs and AWG to choose the output port for unicast operation. For a multi-cast operation, an additional set of wavelength sources (for multiple AWG output port access) and an amplification stage is required. This architecture is shown to scale up to 256-ports [58] or 1024-ports in [59]. However, it requires a high number of components.

The optical TONAK-LION switch uses two AWGRs, one for control and one for data plane, to create a packet switch [60]. In order to reduce control complexity in the optical packet switched fabric (AWGR), the architecture distributes the control plane to the server level. At any given time, contention is resolved using reflective semiconductor optical amplifiers (RSOAs) and demultiplexers at the output of the control AWGR. Each distributed controller only configures a 1 by 2 TONAK switch, reducing switching time to less than 1 ns. Although TONAK simplifies controller complexity, it requires an excessive number of components per AWGR switching, which creates implementation complexity. In addition to this, as the number of wavelength channels available in an AWG devices is much less than the number of ports ($W << N$), an additional complexity is introduced to the 1:k optical multiplexers.

Data vortex is a distributed deflection routing optical architecture that uses fiber delay lines to circulate packets and create optical buffers and uses a simple SOA based control to navigate the packets to the relevant destination with ultra-low latency [61, 62]. Data Vortex control plane, relying on fast arbitration and scheduling, reads specific wavelengths to find the destination and reconfigure the SOA of

the relevant route (SOA) so that the packet has minimum latency. Although the research on Data Vortex suggests scalability, the cabling complexity and implementation complexity for large scale systems makes this architecture a big challenge. The SPINet architecture employs similar routing and SOA switching techniques to create a multi-level switch but also suffers from the same challenge.

OPSquare [63, 64] has a fast SOA-based modular switch architecture and scales by stacking modules in a 2-stage Spanke architecture and using wavelength routing (AWGs). The controller are also modular, which extract label information to configure a direct-modulated laser and an array of SOA, avoiding arbitration for the output modules using wavelength conversion. The arbitration time for the input modules depends on the number of wavelenghs routed, bringing the total scheduling delay down to an impressive 25 ns. However, wavelength conversion and the high number of optical components required to implement OPSquare increase the implementation complexity and cost.

2.3.4 Concern 4: Feature replication

Including Layer 2 and Layer 3 Ethernet control, electronic switches have a few other key features which when migrating to optical switching are lost. Some of the key features which increase the complexity of the optical architecture when introduced include multicasting and broadcasting capability, header processing in the control plane and buffer management schemes. All such operations are traditionally performed on the switch ASIC in Ethernet switches. However, while functioning in the optical domain, such operations require more resources either in the control or data plane, which has a direct impact on the implementation, cost or power consumption.

In the Hipoλaos [59] architecture, multi-casting uses some of the output port resources of the AWGR to multi-cast and broad-cast as they employ packet replication. Moreover, multi-casting introduces additional losses and enforces the requirement of an amplification stage. Also, more tunable wavelength sources are required to enable packet replication on each of these wavelengths. Hipoλaos, a wavelength-time-wavelength, already requires a number of components as discussed in [58]. A simple feature replication of an EPS, therefore, has an implication on complexity, power and cost for an optical packet switch

Architecture		Max servers (N)	No. of switch stages	Scale Limits	Concerns	Cons of using more switches
AWG	**DOS [65]**	400	1	AWGR	Power hungry buffers, scalability of scheduler	Expensive TWCs
	TONAK [60]	1000	156 x 80-port	Transc.	Number of transceivers	Quadratic increase in transceivers
	OPSquare [63, 64]	40,960	1024 x 32-port	AWGR	Cost, Power	Many 1xF selectors
	Petabit [54]	10,000	384 x 128-port	Transc.	High number of expensive TWCs	Exponential increase in transceivers
	IRIS [66]	80	4 x 80-port	AWGR, Transc.	TWCs, Complexity	Complexity, number of transceivers
	Hipoλaos [58, 59]	1024	32 x 32-port	AWGR, Transc.	Multi-casting, Complexity	Complexity, number of components
SOA	**SPINet [67]**	1024	5120 x 2-port	SOA	Number of SOAs required	# SOAs
	OSMOSIS [51, 52]	2048	96 x 64-port	SOA	Power consumed by 8000 SOAs	# SOAs
InTune's OPST [66]		16	1	Transc.	Very low radix	Number of switches
Star Coupler	**Avionics[68]**	120	6 x 32-port	Transc.	Receiver complexity: 5000 diodes	Reduced resources for nodes
	PULSE	16,000	256 x 64-port	Transc./server	Synchronization, communication	Requires more Transc./server

Table 2.1: Comparison of PULSE with other OPS networking solution, their scaling limits and concerns

2.3.5 Summary

To summarise, although there are many benefits with migrating to optical switching, it is important to consider cost, power, latency, throughput, complexity in terms of control and data plane and scalability. Although many optical packet switching technology have produced novel solutions, some parameters are traded off for the benefit of others. Although this compromise is unavoidable when achieving any novel solution, packet switching has several parameters to consider that are interdependent. An alternative is to employ optical circuit switching solutions for data center networks. Table 2.1 shows the complexity involved in designing an OPS and compares PULSE (the proposed ultra-fast OCS) to other OPS research to show its standing. In addition to the complexity involved in designing and deploying OPS technology, they also require the replication of complex electronic packet switching techniques and management algorithms.

2.4 Optical Circuit Switching (OCS)

Traditionally, OCS have been used in many data center network solutions as they offer stable non-blocking circuit configurations with high-capacity and scalability [23]. In contrast to OPS, they are simpler to implement and they eliminate the need for in-switch buffering or queuing and addressing. They establish single-hop connections with a wide range of circuit establishment time, lasting from milliseconds to hours. Leveraging on stable circuit establishments, they can employ multi-wavelength (WDM) and modulation formats to reach high capacity. OCS switches are readily available [23] and are being used as part of many existing networks. They are mainly employed as part of a hybrid network like in [20] in order to cater to specific types of traffic. However, they cannot be used on their own because they suffer from two key limitations: the long reconfiguration time and the long circuit computation time, as shown by Fig. 2.1.

Figure 2.1 shows the circuit computation time and the reconfiguration time of each OCS technology. In summary, slow beam steering and light guiding technology (millisecond OCS) were assisted with slow software-based circuit computation to provide reconfiguration, also in milliseconds (HELIOS, Firefly and OSA) [20, 47, 48]. More recent work has shown microsecond speed WSS-based OCS reliant on FPGA-based control (REACToR, Mordia) [46, 45]. Rotor switches with schedule-less control were also explored for fast OCS [22].

However, with transceivers growing at a staggering rate, already reaching 100 Gbps [69] (trending towards 400G and 800G) and switches achieving 6.4 Tbps [37], the increased data-rate makes OCS 5-6 orders of magnitude too slow. This ever increasing gap between OCS switching/control speed and data rate makes OCS un-

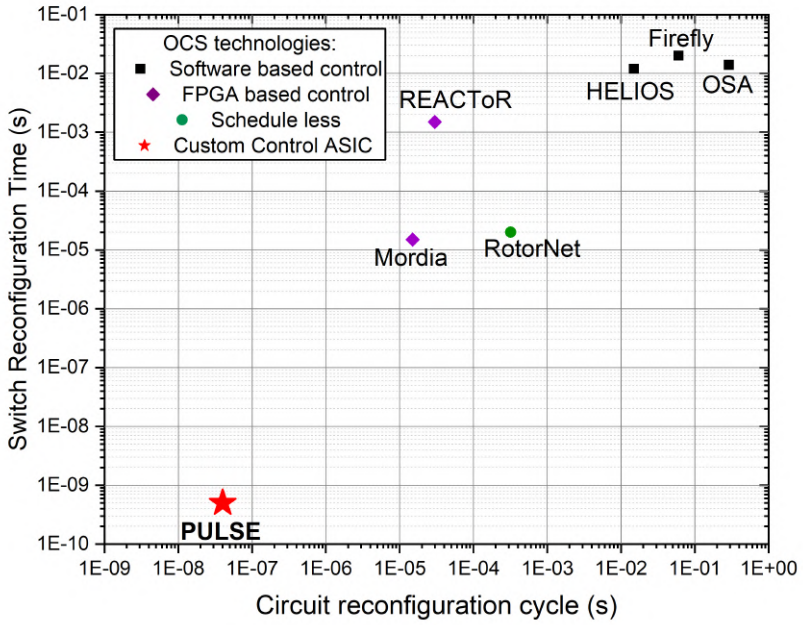


Figure 2.1: OCS circuit reconfiguration and computation time

suitable as standalone solutions. Hence, PULSE (indicated by a star in Fig. 2.1) proposes a two-fold solution: The first is the use of SOA-aided widely tunable-switching methods to minimize the reconfiguration time to sub-nanoseconds. The second is a custom-made ASIC controller or scheduler that reduces reconfiguration computation time to sub-microseconds. PULSE attempts to match OCS switching times to packet-level granularity, making them suitable and adaptable to modern high capacity, bandwidth and speed switching data center networks.

2.4.1 Milli-second speed OCS

In this sub-section, the focus is on the milli-second speed optical circuit switch solutions. The key limitations of both data and control reconfigurations are highlighted for HELIOS, c-Through, OSA and firefly OCS technology that rely on mechanical movement of mirrors for beam guiding are highlighted. OCS solutions, switching schemes/techniques, their reconfiguration, computation time and circuit duration and the components they employ are shown in Fig. 2.2, at the end of this section.

2.4.1.1 HELIOS

The aim of HELIOS, a hybrid EPS and OCS data center network solution, is to reduce cost, power and EPS switch count in a 'sweet-spot' tier and hence, give performance benefits. Micro-electro-mechanical system (MEMS) based high capacity optical switches can perform high bandwidth workload off-loading (Glimmer-glass in HELIOS architecture [20]) and reduce overall electronic switch count, cost and

power consumption. Glimmerglass, an OCS with $N \times N$ crossbar of tiny mirrors is used, each of which is attached to a motor with overall power consumption of 240 mW and a cost of $500/port; however, the reconfiguration time is about 12 ms long. The FPGA-based control counterpart uses a processing system (PS MAC - hence, software) to read the traffic and when a uniform traffic is expected it uses the PHY (with disabled EDC) in the AEL2005 NetFPGA to compute reconfiguration in 15 ms. Hence, the slow configuration time of the MEMS-based optical circuit switching, about 27ms limits their application to long-lived stable traffic; they need to work in co-ordination with electronic packet switches to cater for diverse types of bursty traffic. A faster single comb driven 2048-port MEMS switch has been proposed that can achieve a switching speed of 20 μs [70]. However, the slow software-based approach still implies an overall configuration time of 15.02 ms.

2.4.1.2 c-Through

In c-Through [21], a hybrid packet-circuit network solution like HELIOS, physically rotating mirrors that re-direct laser beams configure the switch. The MEMS switches are reported to consume 10-20ms latency (overlapping with HELIOS on fig. 2.1) and consumes $4200 for switching 48 servers. c-Through employs user manageable daemon interfaces to configure the switch, which for a 1000-rack data center takes about 640 ms on a Xeon 3.2 GHz processor. The proposal expected switching technology to speed up and the costs to go down; however, the interfaces, being software-based controls, still take in the order of 100s milliseconds to compute configuration.

2.4.1.3 OSA

The goal of OSA is to achieve high bisection bandwidth. The OSA optical switch architecture employs WSS and Optical Switching Matrix (OSM - an array of tiny mirrors mechanically adjusted, also MEMs) modules to reconfigure OCS with a 50% non-blocking architecture that works in coordination with ToRs [48]. A low power consumption of 4.74 W/port, a cost of $5600 per switching module is achieved while the switching speed 10 ms. OSA follows a 4-step model that studies traffic, computes topology, computes routes and then assigns wavelengths. The overall time taken for all these computations to finish is about 290 ms. The reason for this high a value is attributed to the traffic study control generator, which is (a) software based and (b) scales with respect to servers rather than racks, causing a high latency.

2.4.1.4 Firefly

The goal of Firefly is to attain maximal matching in order to boost OCS throughput. Firefly propose a practical free space optics (FSO) technique using ceiling mir-

rors above ToRs, requiring mirror switching or guiding [47]. Here, they showcase switchable mirrors (SMs - based on liquid crystal technology) can achieve reduction of switching time to 10-20 ms, power to 40 mW and a low cost of <5% per volume. All the aforementioned techniques have slow circuit reconfiguration times (the right-top corner of Fig. 2.1). The Firefly controller also examines the traffic and adapts topology and forwarding rules in corresponds by using efficient topology selection and traffic engineering algorithms; C++ based OpenFlow topology configuration algorithm is employed [47]. An overall configuration time of 20ms is assumed in this work and the configuration time is not reported. However, being software-based, it can be safely assumed that this computation time would exist in the milli-second domain.

2.4.1.5 Commercial switches

Polatis OCS also rely on beam steering using piezo-electric to align collimators and create stable connections [23]. These commercially available switches also take in the order of milliseconds to configure (25 ms for a 192 port 6000n switch with a low insertion loss of 2-3 dB and power consumption of about 200 mW per port). These are good solutions for hybrid networks that rely on EPS and OCS for different types of traffic; once, a circuit is established the latency through the switch is kept to a minimum. Although they serve the application they are designed for, MEMS OCS technology still incur substantial latency overhead and poor efficiency if employed for switching small size data (e.g. 20 ns data packets); hence, such technology is only suitable for long-lived data flows.

2.4.2 Micro-second speed OCS

In this sub-section, faster OCS technology that operate in the micro-second speed regime is explored. The approach towards speeding up switching times in terms of both the data and the control plane are detailed with the pros and cons. A discussion is carried out on (i) RotorNet that relies on rotor switches and scheduleless solutions, (ii) REACToR and Mordia that rely on wavelength selective switches (WSSs) and FPGA based solutions for faster control. The conclusion, however, is that packet granularity circuit switching and efficient scheduling are required for achieving ultra-low latency.

2.4.2.1 RotorNet

RotorNet architecture has a primary aim to eliminate the complex controller unit that is required for most modern optical switches but at the same time maximize throughput between node pairs. Rotor switches, which are also MEMS-based ocs (micro-mirrors) but instead of switching for every port (large radix), the mirrors are

made to switch for every matching, which is kept to a minimum (4, 8 or 16) [22]. Hence, challenging Polatis or other large radix OCS technology, Rotor switches are able to achieve a switching speed of 20 μs. Many of these Rotor switches are then used to form an entire network to support 512, 1024 or 2048 ports. Forming a multi-stage OCS network requires (i) multiple switching elements to create a large network and (ii) a network wide cycle time in the order of 1 ms. Moreover, the quality of service is uniformly distributed as the switching is cyclic. However, the diversity in high-performance applications, their workloads, tasks and processes are ever increasing making this approach unfair to the required demand and load within the network. The number of rotor switches, N/M, were shown to reduce by employing two layers of the square-root of the number before $\sqrt{(}N/M)$ in [71] and support bi-directional traffic. An advanced architecture that increases permutations is employed to significantly reduce switch count and hence, increase scalability.

In terms of control, the need for a centralized controller is completely eliminated as round-robin based rotor switching does not require demand estimation or schedule computation [22]. However, as shown by Fig. 2.1, the reconfiguration cycle is still limited in the microsecond region and the unfair distribution of resources in a diverse environment could result in power throughput and extremely high latency in an environment with a diverse traffic distribution like a realistic data center traffic pattern.

2.4.2.2 Mordia

The goal in Mordia is to develop an ultra-fast optical circuit switch in order to increase throughput and minimize switching overheads. Mordia relies on 2D-based MEMS, instead of the conventional 3D MEMS, for wavelength selective switching (WSS). The initial work in [45] proposed a 24×24 OCS switch, where each wavelength is connected to a given port. A uni-directional ring topology is adopted with add/drop wavelength units with switching time of 11.5 μs (including ringing). However, such a network topology is not scalable. Hence, it is proposed as a hybid EPS/OCS integration solution in [72] in order to boost scalability.

The controller that was first proposed for use in Mordia was based on an FPGA-based SPI protocol developed to create a high-speed digital trigger signal. When working as an integrated solution in a hybrid network, a traffic demand estimation algorithm is required. A TDMA-based a minimum cycle time of approximately 120 μs (equivalent to one slot) is used as an end-to-end reconfiguration time. Secondly, a large buffer per port is required to account for the 11.5 μs switching overhead created. Operating in the micro-second domain is still very slow when transmitting/receiving packets in the order of tens of gigabits per second.

2.4.2.3 REACToR

The architectural goal in REACToR is for OCS to co-ordinate with EPS to maximize the performance and get the benefits of both worlds. REACToR also integrates Mordia OCS to an EPS based network as an hybrid solution. Employing fairly shared TDMA, REACToR takes in the order of 1.5 ms (multiple configurations) to compute each schedule. The data packets use the OCS and all TCP acknowledgements are communicated via the EPS and every timeslot (or circuit) is set to last 214.5 μs. However, REACToR has a minimal flow latency of 3 ms, which significantly increases the median and tail latency.

2.4.3 Summary and Proposal

Although the switching time in microsecond speed OCS switches is reduced by 3 orders of magnitude by simplifying MEMS technology, the methods of data/control employment make them severly suffer in performance. In the case of RotorNets, switching time is reduced to tens of microseconds and computation time to zero (cyclic switching); however, an even distribution of resources with no control or traffic estimation would not be suitable to cater a diverse environment. Mordia OCS, in itself, has a faster switching time also in the order of tens of microseconds. However, employing a a ring topology affects scalability and they are forced to be integrated with EPS solutions. REACToR has a long switching time circuit reconfiguration time resulting in high latency. Hence, these solutions are still quite slow considering that packet transmission speeds are ever increasing. A sub-nanosecond speed circuit switching with effective sub-microsecond speed scheduling can substantially reduce both the communication overhead and latency.

Topology	Switching Scheme	TDMA	Switch time	TDM Slot Resolution	Min. Circuit Duration	Compute time	Device(s)
Helios	Hybrid EPS/OCS	No	12ms	-	O(s)	15ms	MEMS
OSA	Circuit	No	14ms	-	O(s)	290ms	WSS, OSM
Rotornet	Circuit, EPS ToRs	Yes	20μs	O(100μs)	O(1ms)	-	Rotor switch
Firefly	Circuit, EPS ToRs	No	20ms	-	O(s)	60ms	LC, Galvo mirrors
REACToR	Hybrid EPS/OCS	Yes	30μs	185μs	1.5ms	O(10μs)	100G OCS, 10G EPS
Mordia	Circuit	Yes	15μs	95μs	O(100μs)	O(10μs)	WSS
PULSE	Circuit	Yes	500ps	20ns	40ns	40ns	SOA-based transceivers

Figure 2.2: Table of comparison: The relevance of PULSE with respect to current leading OCS Network Research

2.4.3.1 PULSE

As shown in Fig. 2.1, I propose PULSE, an ultra-fast transceiver-based wavelength selective (WS) and timeslot (TDMA) switching architecture that has all active switching elements at the end nodes. As shown by the table in Fig. 2.2, the proposed OCS architecture of PULSE has a short reconfiguration cycle of 40 ns and a transceiver switching speed in the range of 500 ps. The table has been included for comparison purposes showing the stand of PULSE with respect to OCS technology; the architecture goals and their prime device technology are also highlighted. To my knowledge and from my research, this is the fastest switching and reconfiguration cycle achieved among current leading OCS research. The details of the architecture and switching technology and their characteristics can be found in chapter 5.

Chapter 3

Network Architecture and Scheduler

3.1 Architecture

PULSE achieves 4-5 orders of magnitude faster switching by adopting switching at the transceivers (or end-nodes). Pushing the switching to the end-nodes removes the need for reception, re-transmission, buffering or O/E and E/O conversions for buffering and the mechanical or optical guiding movements in the core of the switch. Hence, switching is greatly simplified by distributing it to the local transceiver of a blade. Scheduling control network is also simplified as it can be co-hosted within the same rack as the source servers. However, like any OCS technology, PULSE has a need for a network. However, as actively and dynamically switching optical elements are already present at the nodes, the network architecture needs to be passive and optical; this would help to keep the network and the scheduling simple.

3.1.1 Ring network

A ring topology can be used to connect PULSE's active nodes. The use of a ring network requires optical add/drop multiplexer (OADM) stations equipped with fiber Bragg gratings and optical circulators. Intune's OPST architecture proposed to have a WDM-based ring architecture for data center networks [66]. In addition to this, multi-layer ring topologies have been explored to employ as a scalable all optical solution that scales and reaches more racks [73].

However, employing ring topology limits the network in terms of scalability. The filters cause a percentage of signal to be dropped at each node causing the last node in the network to suffer from the highest loss. Hence, the loss until the last node in the circle defines the scalability of the network. Hence, even if a passive solution, the limited scalability affects this network from being a chosen as a design choice for PULSE.

3.1.2 AWGR

Cyclic Arrayed Waveguide Routers (AWGRs) also provide an optical backplane, where choosing the wavelength corresponds to a unique port. They have been employed as switching fabrics network architectures like Petabit [57], Petastar [57], DOS [65] and TONAK [60]. They have also been included as an architectural element in [59]. Very low insertion losses vertically tapered waveguides have been under research that scale to 64 ports (like in PULSE) [74, 75]. AWGRs have also been shown to scale 512 ports [34]; however, (i) a high cross talk of -4 dB is experienced and (ii) the tunability range of DS-DBR lasers limits the wavelength range to $<$89 wavelengths, as shown in [5]. A disadvantage to resolving to AWGR based network is the reduction in flexibility of multi-casting or broadcasting. Any need for multicasting or broadcasting would require PULSE to employ packet replication and wavelength wastage depending on the size of multi-cast. Hence, as this would severely degrade the network performance, it is not adopted in PULSE.

Compared to the star-coupler approach employed, this network solution avoids high losses in the core. The scheduling performance in chapter 6 is still valid for a network that employs AWGR (independent of switch fabric). However, as wavelength assignment corresponds to a port-assignment in AWGRs, the PULSE scheduler does not need a look up table for generating parallel pseudo-random wavelength numbers at the wavelength decision stage. Hence, the scheduling can be simplified; this comes at the cost of supporting only uni-cast traffic in today's HPC network environment.

3.1.3 Star-coupler

A star coupler creates a broadcast and select network, where the signal power of all inputs is equally distributed across all output nodes. The receiver wavelength selection selects which input port to the receiver must listen to. Star-coupler networks have been proposed for 1000-port optical switches like POTORI [56] and our proposal in [24]. Star coupler networks inherently support uni-, multi- and broad-cast and eliminates the need for packet replication. The key limitation in a star-coupler is the high optical loss it has ($3log_2N$, where N is the size of the coupler), which requires compensation by amplification and/or high launch input power in order for the receiver to have a high SNR optical signal. As discussed, AWGRs can be alternatives for the coupler-based solution. However the research in [76] shows that although AWGs at the ToR in current data center networks may achieve good scalability, it suffers in flexibility. However, star-coupler networks are shown to meet host requirements due to the high flexibility it offer, although at the cost of high insertion loss. In PULSE (the version proposed with star-couplers), multiple levels of

flexibility are introduced; (i) casting (uni-, multi- or broad-), (ii) dynamic tunability at slot level (iii) wavelength selection at the transmitter and, receiver. At the same time, the losses are accommodated for by employing coherent reception and the use of SOAs at the transceivers.

Although the initial proposals of the proposed OCS start off with achieving 1000-ports, I show that these severely degrades network efficiency and hence, resolve to 64 to 256-port networks (primarily) so that wavelength utilization can be maximized. On another note, in epoch-level tuning/scheduling, if multiple transceivers are tuned to the same wavelength (for slotted communication), then the transceivers tuned to the same wavelength affect the extinction ratio of the optical signal as shown by [25, 5], as the DS-DBR lasers are never truly turned 'off'. All scheduling algorithms take these physical limitations into account.

3.2 Ultra-fast OCS Scheduler

In this section, the control plane requirements of the ultra-fast PULSE OCS technology are presented in terms of processing capacity and time. The traditional software-based dynamic wavelength allocation techniques are revisited and problems with serial allocations are highlighted. Howbeit, the software-based serial resource allocation techniques are adapted for a star-coupler architecture and redeveloped for performance comparison in Chapter 4. Finally, along with research, other suitable scalable hardware schedulers or scheduling elements are critically analysed for use in PULSE.

3.2.1 Bi-partite matching: Complex NP-hard problem

Using graph theory, each request from a source to a destination can be modelled as a Bipartite graph as shown in Fig. 3.1. Consider the graph on the left with a set of source nodes (indicated by the blue vertices on the left) and a set of destination nodes (indicated by the red vertices on the right). Each edge (arrow) indicates a request from a source to a particular destination. In the case of the figure, an average of 2 requests (R = 2) and 5-port sub-network (N=5) is assumed. In order to compute maximal matching, path augmenting can be employed. However, to arrive to one set of maximal matching for one timeslot it takes several cycles and the computational complexity of this increases with N and R. For an entire data center network with 10s to 100s of thousands of blades, this problem grows vigorously. A maximal matching operation step would be equivalent to allocating one timeslot and this process must be repeated for several timeslots in an epoch (batch).

Using parallel greedy and Karp-Sipser algorithms for maximal matching takes in the order of hundreds of milliseconds to seconds even when employing multi-

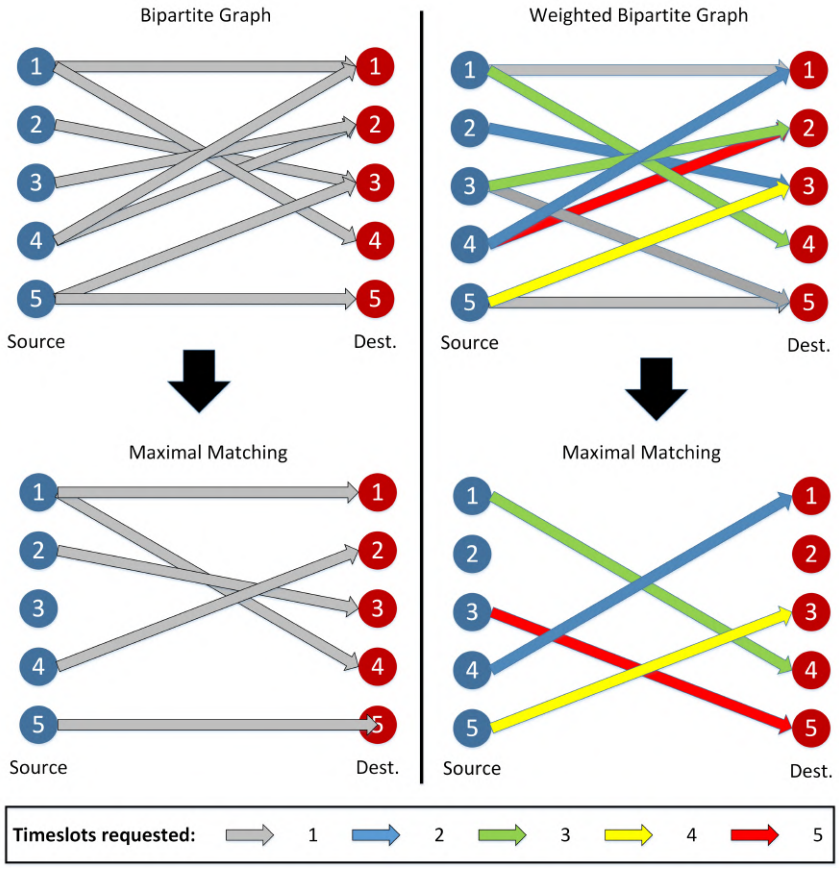


Figure 3.1: Bipartite and weighted bipartite graphs: 5x5 Request from Source to Destination

threaded multi-core processor-based solutions to compute as shown in [77]. Employing hardware implementable solutions of a Pure Logic Scheduler and Matrix Scheduler algorithms showcased in [78] provides high clock speed of 10 to 12 ns clock period on an FPGA. However, in Matrix Scheduler takes N cycles to perform one optimization step (N = 64 in PULSE, as it employs modularity) and a Pure Logic Scheduler takes O(1) step. However, the complexity of Pure Logic Scheduler implies that 100,000 LUTs are required for a 64-port system. Although this logic can be implemented on ASIC for simplification, the maximal matching achieved here does not consider the size of the request and hence, sustained reliable performance cannot be achieved.

Consider a second graph shown on the right in Fig. 3.1, but this time, the edges are weighted, shown by the colour on the edge, forming a weighted Bipartite graph. Note that the maximal matching achieved is completely different now. In fact, it has less edges that match but more weights. If maximal matching in itself is a NP problem that requires high computational (non-deterministic polynomial time), weighted maximal matching increases the computational requirement. Moreover,

as the ultra-fast scheduling is required to do this in sub-microsecond timescales, the problem becomes NP-hard.

3.2.2 Dynamic Wavelength Allocation: Software Algorithms

The inherent advantage of employing hardware (especially) over software is the speed up it gives in computation. However, dynamic wavelength assignment is not a new area domain. They are prominently employed in WDM networks or wavelength routed optical networks (WRONs) or optical burst switching (OBS) and they employ software-based wavelength allocation techniques [79, 80]. These software-based techniques can be implemented on hardware using simple gates to read, check registers and make a decision on wavelength in a serial timeslot-by-timeslot fashion: Least Used, Least Loaded and Random, for example. Software model equivalents (not hardware) of these techniques are adopted for comparison standards and include in the Appendix A section. A maximal matching software model is also made to compare parallel hardware allocation algorithms with serial allocation techniques in Chapter 4.

Consider a 64-port, 2-request system as showcased later in PULSE. If the average request size is one timeslot, a 40 ns epoch is created, and the timer is set for the scheduler to compute reconfiguration in 40 ns. Each clock cycle considers one timeslot; hence, as a total of 128 timeslots are to be served, a clock period of 3.2 GHz is required. However, the size a 6-stage all combinational 2:1 multiplexer (MUX) is required both at the transmitter and receiver register just to check the wavelengths. In addition to this, the wavelength computation based on availability will require more logical resources. Hence, a sequential allocation cannot achieve such a clock speed even if the fastest CMOS technology is employed; and all this effort would be invested just to complete one iteration through all timeslots. Moreover, larger epoch sizes will require more clock cycles. Hence, as parallelism is essential for computing wavelength/timeslot resources, a highly parallel hardware is employed in PULSE and it is shown to achieve a high throughput of above 92% in a 1-plane network at 100% input load. Another alternative to this approach is to have controlled parallelism and employ simpler allocation elements. However, employing such techniques would give priority iterations to one parallel set over the other and reduce fairness in the quality of service.

3.2.3 Specific OCS Scheduler Requirements

The scheduler is required to process a large data set that is dependent on the size of N, W, R and S (N=# nodes/rack, W=# wavelengths, R=Average # requests/server and S=Average #slots/request) within a given timespan defined by the epoch (batch) size. In the initial 1024-port 4-star 80-wavelength proposal, with every request (10-

bits) corresponding to one destination with an average slot size of 4, the scheduler was designed to collect 80 kB of requests and compute multiple grants (up to 80 kB) in a TDMA system within 1 μs. Parallel collection and distribution is a much smaller concern to the scheduler compared to the computation of matching with a high, consistent throughput above 70% that is tolerant to load, traffic and scalability that is required to be done [81]. The 80 kB of request would refresh every epoch.

Although PULSE Network schedule Processing Unit (NsPU) computes for smaller sub-networks (N=64 – decreased from 1024), the problem is not only limited to N, W, R and S but now also dependent on iteration/buffer management. The increase in wavelength channels includes more resources to map meaning that the scheduler needs to work harder (W=64 – decreased from 80). Note the reduction in resources is significantly smaller than the reduction in port count. In addition, the PULSE NsPU has to smartly manage buffer and iterations to compute a schedule within 40 ns with consistent throughput of above 90% that is tolerant to load, traffic and scalability.

It is important to understand that this is an extremely difficult problem to solve and the PULSE NsPU does it within the time given. The hardware design is able to achieve this performance by employing spatial and temporal parallelism. To solve the NP-hard matching problem, scalable parallel hardware elements, like the round-robin arbiter, are employed [82]; the details of which follow in chapter 3. Sorting hardware that re-arranges and checks after every sort are not considered due to the high number of clock cycles they require as N grows.

The previous chapter justified the pressing need of nanosecond speed optical circuit switched network architecture in future data centers to reduce long tail latency, power consumption and thereby, enhance network performance and scalability. The focus of this chapter is on (a) the development of the network architecture, the data/control plane, (b) the transceiver architecture and (c) the central scheduler. This chapter showcases, in a progressive manner, the development of each section highlighting the improvements and alterations required to achieve a better performance. The first network proposal serves as a starting point by discussing the steps undertaken to develop a simple coupler based network, identifying the fundamental optical and scheduling limitations. Highlighting the concerns of each prototype, each stage is analyzed in an effort to improve these architectures (network/transceiver/scheduler) and thereby, enhance performance. For more details on the architectural principles of the parallel hardware scheduler and the proposed optical network, the reader can refer to the papers published in IEEE HotInterconnect 2017 [81].

3.3 The first proposal: An overview

As discussed in the previous chapter, optical interconnects and switches reduce overall power consumption. One of the major contributors to the cost involved in setting up a datacentre network is the transceiver cost. Reflecting on the fact that low-radix switch based networks require a high number of transceivers, using high-radix optical switches would demand relatively less number of transceivers, reducing the cost significantly; provided that the signal is kept completely in the optical domain. Hence, Ethernet-based datacentre networks face power consumption, bandwidth and cost limitations when forced to scale. These factors limit the electronic network from meeting the increasing bandwidth requirement and emphasise the need for an optical network.

High-radix switches can be used to build a flatter network. This means that the optical network is power and cost effective and also sends data in a single hop. However, the major challenge faced is that the complexity of the central scheduler increases when scaled to support a high-port network. There is a limitation to the number of servers such a network can support. The work done has been done to prove that an all optical 1000-port switch can be implemented using a passive star-coupler core, tuneable transmitters and coherent receivers.

3.3.1 Star Coupler Network

In 2015, a high-radix optical switch was proposed to use WDM and TDM with high sensitivity coherent receivers, enabling the star coupler to support more splits [24]. The research work proposes to multiplex 80 (or 160) wavelengths in the C- (and L-) band with 50 GHz spacing (WDM) and interleave data from multiple nodes in a time-slotted system (TDM). Fig. 3.2 shows the hardware components used in the data plane of the optical network.

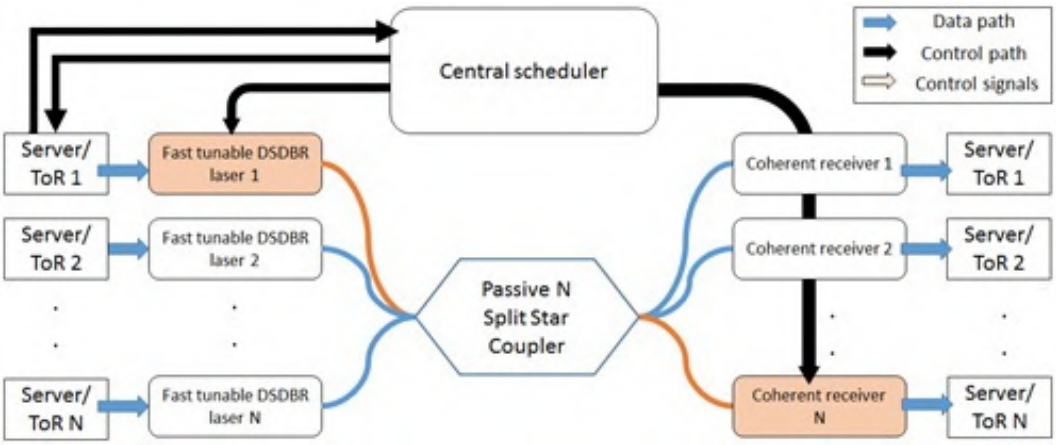


Figure 3.2: Optical switching with Configuring wavelength and timeslot of smart end nodes

Each server is equipped with an optical transmitter per star, comprising a Mach-Zehnder Modulator (MZM) and tunable DS-DBR laser, and a coherent receiver with an independently tunable DS-DBR local oscillator laser. This enables fast wavelength selectivity at both the transmitter and receiver, and the high sensitivity of the coherent receiver also enables scalability in the data plane since it allows for a larger system loss budget and thus a higher port count star-coupler. For data transmission across the switch, transmitter and receiver must both tune to the same wavelength. As there are 1000 nodes to serve but only up to 80 (or 160) wavelengths, time division multiplexing (TDM) is also adopted to better share the total available bandwidth between the nodes and maximize throughput.

The fast tunable DS-DBR laser in every transceiver synchronously tune (for 200ns) to a specific wavelength and timeslot decided by a central scheduler. The useful communication time is made to be 2 μs, so that the tuning overhead is reduced to remain within 10%. The MZM modulates the laser with the signal from the servers or the ToRs.

3.3.2 Data plane parameters

1. Network size: The aim of the network is to support a total of N nodes, whose scalability is optically limited by the loss that the coherent receiver can handle. As already discussed, the splitting loss in a star coupler network is $3 \times log_2 N$ dB. By using a coherent receiver with a sensitivity of -27 dBm and a transmit power of 6 dBm, after allocating 3 dBm for system and coupling losses, the system has an optical power budget of 30 dB 3.2. Hence, the switch can support 1000 ports owing to the highly sensitive coherent receiver. Hence, the first limitation is defined by equation 3.1:

$$3log_2 N \leq 30dB \implies N \leq 1000 \tag{3.1}$$

2. Wavelength resources: Secondly, the proposed network has a limitation, on the number of wavelengths, W. With a 50GHz frequency spacing in the C band, 80 wavelengths can be supported; if both the C and the L band is used, 160 wavelengths can be supported. Hence, the second limitation is defined by equation 3.2:

$$W \leq 80 \tag{3.2}$$

3. Timeslot Size: Thirdly, the epoch is divided into fixed length timeslots; the total number of slots is T. The fixation of the length of this timeslot requires the payload to be broken down to fit into equisized timeslots. The duration of one time-slot must be able to at least support the lowest Ethernet packet or payload

size. In other words, the timeslots should not be shorter in time than the time taken to communicate the smallest packet or payload, t_{min}. Hence, there is a limit on increasing the number of slots, T. On the other hand, if the whole epoch (of size E) is just one timeslot, the throughput suffers significantly. Hence, the time-slot has an upper limit, t_{max}, exceeding which, the throughput of the network is too low. This limitation is defined by equation 3.3.

$$t_{min} \leq E/T \leq t_{max} \tag{3.3}$$

The timeslot size is also dependent on the transceiver line rate used. In this work, a DP-QPSK modulation is proposed for transmission to achieve a line rate of 100 Gbps (assuming a symbol rate of 25 GBaud). At 100 Gbps, a 2 μs epoch allows each wavelength channel to carry 25 kB (kilobytes). By dividing the duration of an epoch into 100 time-slots, this system targets the transfer of 250 bytes per time-slot, which corresponds to the overall median packet size across various data centre workflows [83].

4. Same wavelength transmitters: Fourthly, there are N nodes but only W wavelength channels; multiple nodes are assigned the same wavelength. There is a limit on the number of nodes which can tune to the same wavelength in the same epoch (M). Tuneable lasers take considerable amount of time to turn on and off; some DBR lasers even have initialization times lasting up to 60 seconds [84]. Hence, the N tunable lasers are not turned completely off; but when one node is communicating in its given timeslot, the modulator power output of all the other nodes tuned to the same wavelength is set to minimum. The lasers in the other transceivers with the same wavelength are not truly off; there is a constructive interference of the 'off or zero' states of these transceivers. This has a direct effect on the optical eye i.e., the extinction ratio. Hence, the number of nodes using the same wavelengths, M, must be kept under a certain value for each of the optical signals to have the minimum extinction ratio required. This limit, as shown by equation 4, was identified to be 10 in [25], then shown to be 25 in [5].

$$M \leq 25 \tag{3.4}$$

3.3.3 Control Network

The control plane of the proposed network also has a star coupler. The lasers in the control star do not have a dedicated tuning time as in the data plane (as shown in fig. 3.2) but work on fixed wavelengths. The configuration of the switch required for the next epoch is computed by the scheduler in the current epoch. Ideally, all

nodes would send a burst of request on a WDM and TDM-basis using their assigned wavelength. Inside the scheduler, a CMOS chip is connected to receive signals from every node, which computes the reconfiguration required for the next epoch in the current cycle. As already indicated, a synchronized tuning time of 200 ns is dedicated for all transceivers and the epoch size is defined to be 2 μs in order to reduce tuning overhead.

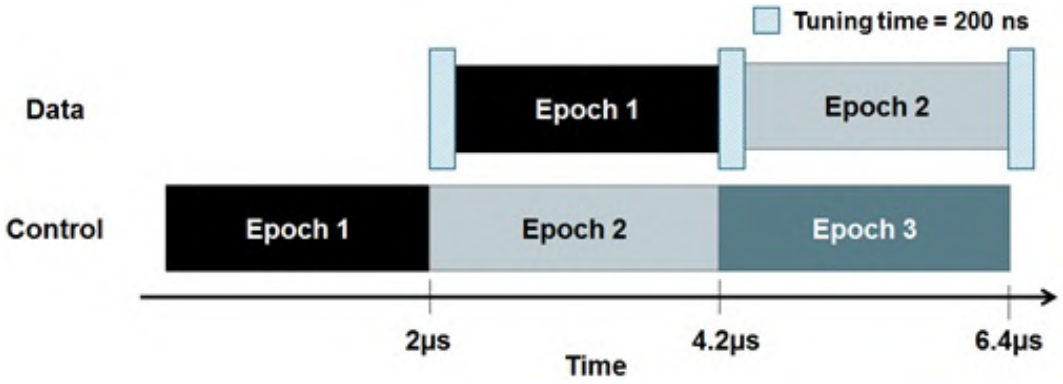


Figure 3.3: Proposed Network Timing: Data and Control

Figure 3.3 above shows the working timeline of the data plane and the control plane. The proposed architecture can be shown to follow three steps:

1. Control plane: the scheduler processes requests sent by multiple nodes and generates grants in the current epoch (2μs).

2. Tuning time: As dictated by the scheduler, the switch is configured by tuning the transceivers during the tuning time (200 ns).

3. Data plane: Once configured, active communication takes place between nodes and in a time-slotted epoch (2μs).

Focusing mainly on the control network, it is important to understand the process of wavelength and timeslot allocation from the scheduler's perspective. The scheduler has to collect requests, compute the configuration and send the grants to the transceivers; and all of this should be done within 2 μs (before the start of the next epoch). This puts a limit on the number of scheduler iterations, I, the clock period, c (ns) and size of an epoch, E (μs) (Equation 3.5).

$$I \times c \leq E \tag{3.5}$$

To maximize throughput and establish multiple paths, the number of iterations, I, on the scheduler must be increased. This means that the scheduler ASIC designed must meet timing within a few nanoseconds. The complexity of a hardware logic implemented is proportional to the critical path length of the design; the more complex the design, the more the time taken and higher the clock period,

c, for the design to process one iteration. By designing a simple logic that meets timing quickly, more number of iterations can be performed. The scheduler must employ parallelism for computing the requests from multiple nodes. By using parallel hardware logic, the scheduler can compute multiple grants in a given clock cycle.

3.4 Centralized Scheduler

3.4.1 Scheduler perspective

As shown by fig. 3.4, there are three stages or states in the scheduler algorithm:

1. **Collect state:** Collect requests and create a demand matrix.
2. **Schedule state:** Resolve contentions and assign resources (I iterations).
3. **Send state:** Send out all the collected grants.

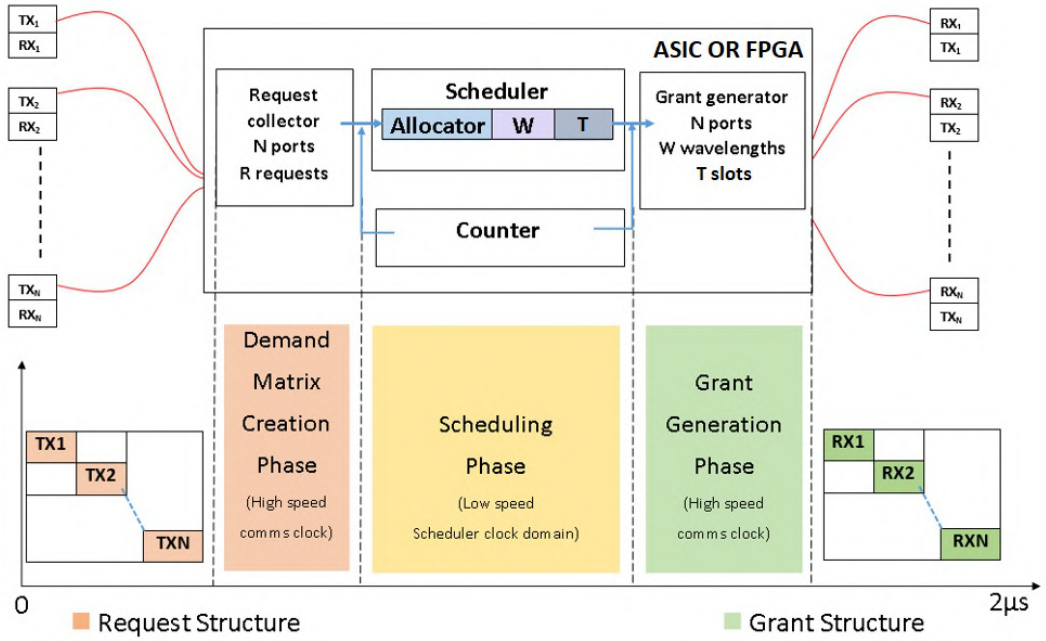


Figure 3.4: Scheduler overview: from request collection to grant generation

As shown in fig. 3.4, the scheduler must collect requests and create a demand matrix for an epoch, compute schedules and send grants within 2μs. Hence, the scheduler must have a time-dependant control. As shown by fig. 3.4, the communication of requests and distribution of grants is done at a higher clock speed with serialization, while the scheduling clock speed is determined by the hardware logic inside the scheduler (FPGA or ASIC). A counter is used as a global controller to control the 'phase' of the scheduler. The counter initially is in the 'Request Collector' phase; the counter waits for the collection of requests until a set time, after which no requests are collected, and toggles the design to the 'Schedule' phase. In

this phase, there are two stages: 'Allocator' resolves node contention resolution and 'Resource Assignment (W,T)' will assign wavelength and timeslots to the requests. As a highly parallel allocation is aimed for, it is important not to create clashing grants. Hence, a contention resolution stage is important. The 'Resource Assignment' phase allocates wavelength and timeslots, storing each of the grants created. Multiple iterations are performed either until all the requests have been processed or before the time set by the counter runs out; after this the counter toggles the design to the 'Grant Generation' phase. In this phase, all the grants that have been processed are sent back to the nodes by the grant generator. Once sent, the counter resets the scheduler design to collect requests for the next epoch.

3.4.2 Control Communication Protocol

Figure 3.5 shows the request structure sent by one node (server/ToR) and the grant structure it receives. When a node sends a request to the scheduler, it uses request structure as shown; each request specifies the destination node and the number of timeslots it requires. Once the scheduler computes the grants it sends back to grant structures: the wavelength and the timeslots. The wavelength grant is communicated to the transceivers of each node to configure the high-radix switch. The (time) slot grant is communicated to the end nodes to define which of the R requests was granted (to identify the set of packets it should send) and which timeslot it must use to send the packets to the right destination.

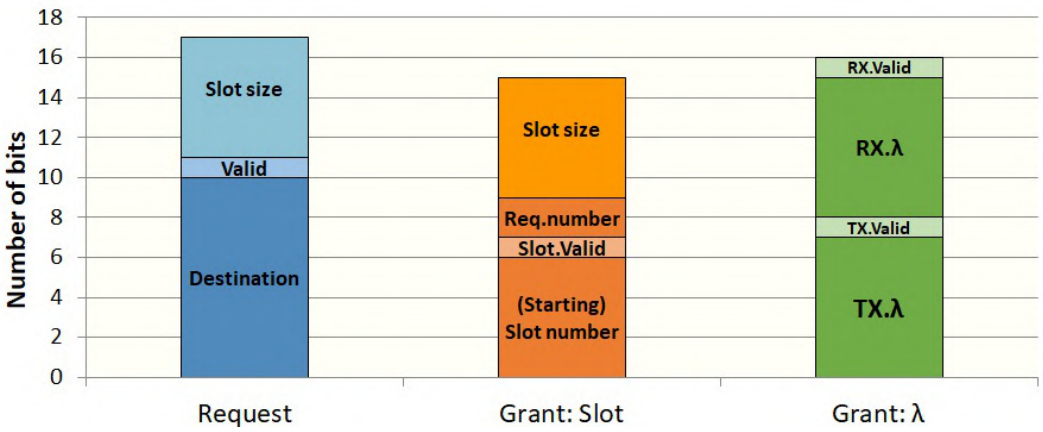


Figure 3.5: Control plane communication protocol: Request, Grant structures for N = 1024, W = 80, R = 4 and (T)= 50

Each request signal, shown in fig. 3.5, contains the destination node request ($log_2N = 10$ bits for 1024-ports), the slot size ($log_2T = 6$ bits for 50 slot system) and 1 valid bit. Each timeslot in the data plane corresponds to a 20 ns timeslot capable of communicating a payload of 256-bytes. Hence, each request has a total of $log_2N + log_2T + 1 = 17$ bits. Multiple nodes send their requests to the scheduler

in the control star. The requests are stored in a local memory (an array of registers or RAMs) in the request collector. Once a total of R requests are collected from all N nodes or once the timer/counter expires, the request collection is completed. Hence, this stage holds a total of $N \times R$ 17-bit registers.

At the 'Resource Assignment' state, each iteration of the scheduler will assign wavelength and timeslot configuration to non-contending node pairs. Each node (server or ToR) must know the correct timeslot(s) it must use in the data plane to communicate the relevant payload. Hence, the slot grant structure, shown in fig. 3.5, contains the request number ($log_2R = 2$ bits $\implies$ easier to communicate than the destination), the starting slot number ($log_2T = 6$ bits) and the slot size ($log_2T = 6$ bits). The number of bits required to communicate one grant timeslot information to one node is $log_2R + 2log_2T = 15$ bits per grant, which corresponds to a total of up to $N \times R$ 15 bits.

The wavelength allocation is done once per transceiver per epoch (no mid-epoch tuning). The star coupler network has increased flexibility because both the transmitter and the receiver are tunable to any wavelength channel in the system. However, what this means to the scheduler is that any configuration for specific timeslot(s) is complete only when both the source (transmitter) and destination (receiver) are tuned to the same wavelength. Every time a wavelength is assigned to a source node, the scheduler has to look up the destination node, to assign the same wavelength to the receiver.

Figure 3.5 shows the wavelength grant structure. Each transmitter at the source node has a wavelength ($log_2W + 1 = 8$ bits) and each receiver at the receiver node also has a wavelength ($log_2W + 1 = 8$ bits). A total of $2 \times (log_2W + 1) = 16$ bits is required to configure the star coupler network.

3.4.3 Round-robin Arbiter principle

Now, the focus is on what scheduling heuristics the central controller or scheduler should employ to make the design (a) implementable in hardware and (b) scalable for proposed high radix optical switches. A parallel multi-core hardware can perform complex tasks at an efficient rate compared to serial software based allocations. However, parallelism comes at a price; high parallelism introduces high contention. Hence, in the parallel hardware scheduler design, round-robin arbiters serve as key hardware elements. The scheduler requires a two stage arbitration for resource allocation in order to provide competitive matching. At every iteration, the scheduler considers up to N requests. The first stage of arbiters is used to resolve contention between multiple nodes requesting the same destination node. Secondly, although N requests can win the first stage, only W wavelength channels are avail-

able. Since $W << N$, multiple node pairs could request the same wavelength, which would require a second stage of arbitration. Hence, each iteration in the scheduler design is limited to generate up W grants.

Up to $N \times N-$port round-robin arbiters are required to implement contention resolution of destination. Hence, it is important to analyze the scalability of round-robin arbiters before going into the details of the design. The aim of this exercise is to find out the critical path length or the clock speed of the arbiters, as they would remain as the dominant logic in the hardware scheduler design.

3.4.3.1 Arbiter

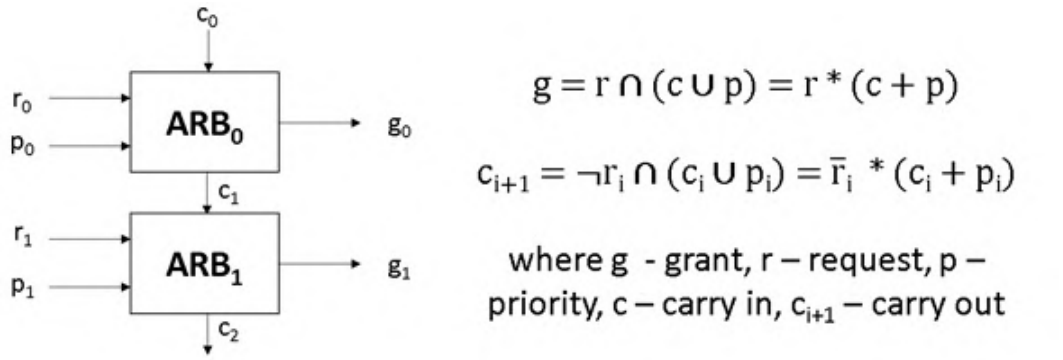


Figure 3.6: Arbiter black box: (a) Slice Abstraction (b) Equations - generate grant, carry

As shown in fig. 3.6(a), the hardware abstraction of a single-bit round-robin arbiter slice is a black box that takes in request, priority and carry-in signals as inputs and generates grant and carry-out signals as outputs. When multiple arbiter slices are tied together, as in constructing a full-adder digital circuit, a vector-based arbiter with N-bit input and N-bit output is formed. Hence, fig. 3.6(a) is the abstraction of a simple 2-bit arbiter, showing all input and output signals.

Figure 3.6(b) shows the functional representation of what happens inside the arbiter black box. The grant and carry generation are shown by the equations in 3.6(b) and re-iterated using equations 3.6 and 3.7 below:

$$g = r \ \& \ (c|p) \tag{3.6}$$

$$c_{i+1} = \bar{r_i} \ \& \ (c_i|p_i) \tag{3.7}$$

The carry is only generated if there is no request in the bit of interest and if there is a carry signal active or priority. If there is a carry signal active and a request is available in the bit of interest, a grant is generated. In this way, a parallel computation of grants is done using hardware arbiters. The details of the arbiter construction can be found in [85].

3.4.3.2 Round-robin priority

Each arbiter slice stores its priority in a programmable priority encoder (PPE). How these priorities are arranged decide if the arbitration is round-robin, variable priority iterative arbiters, fixed priority or weighted priority arbiter. The principle of a round-robin arbiter is that the request which was most recently served should have the lowest priority in the next arbitration iteration. To accomplish this, the current grant vectors are used to update the next priority vector. From a signals perspective, if g_n has recently achieved a grant, then p_{n+1} will go high in the next iteration. Hence, the priority update for the next clock cycle for a round-robin arbiter, as implemented in hardware, is shown in equation 3.8 as:

$$next\,p_n = |g\ ?\ \{g[n-2:0], g[n-1]\} : p_n \tag{3.8}$$

As shown in equation 3.8, vector g is an N-bit vector (N-1:0). If any grant is present, the grant g_{n-1} updates priority p_n, as shown in fig. 3.7; if no grant is present, the priority is not updated.

3.4.4 Arbiter: Implementation and Scalability

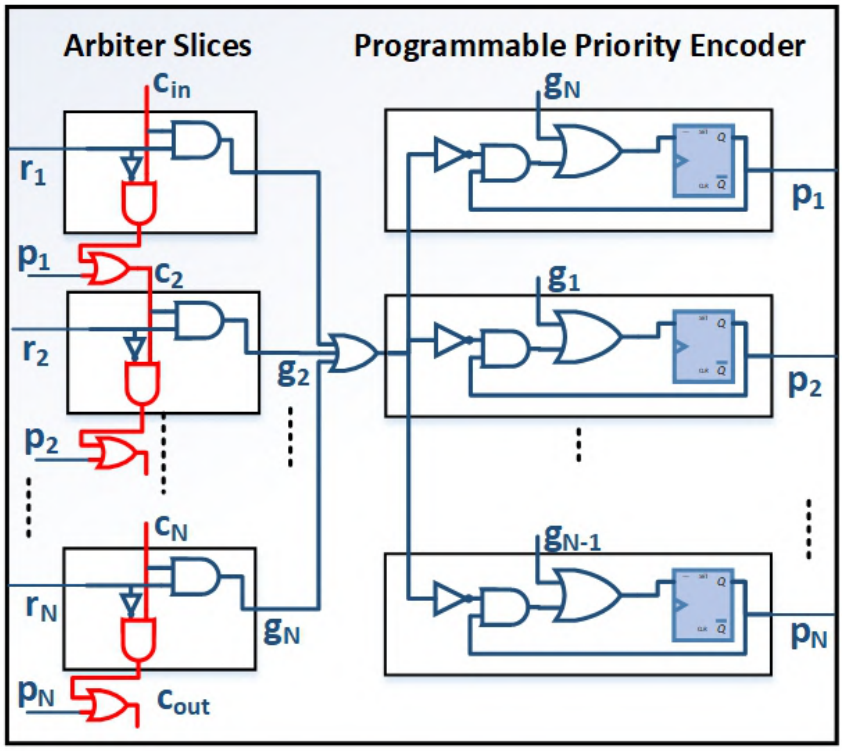


Figure 3.7: N-port Round-robin Arbiter: Logical schematic

The hardware logical representation of the round-robin arbiter is shown in fig.

3.7. The arbiter has two main sections as shown in the figure: the arbiter slice generate the grant and the carry, while the programmable priority encoder updates the priority for the next clock cycle. The registers shown in the priority encoder sections hold the priority for the next clock cycle.

N, Ports	Lah 0	Lah 2	Lah 4	Lah 8	Lah 16	Lah 32	Optimum
4	**0.55**	0.55	0.55	-	-	-	0.55
8	0.86	**0.72**	0.72	0.85	-	-	0.72
16	1.39	**0.83**	0.87	1.19	1.39	-	0.83
32	2.42	1.15	**1.04**	1.27	2	2.44	1.04
64	4.51	1.86	**1.42**	1.47	2.17	3.33	1.42
128	8.61	3.24	2.12	**1.85**	2.3	3.42	1.85
256	16.88	5.94	3.52	**2.48**	2.64	3.64	2.48
512	33.3	11.24	6.31	3.81	**3.31**	3.95	3.31
1024	66.22	21.84	11.63	6.53	4.68	**4.64**	4.64

Table 3.1: Arbiter clock period (ns): 45nm CMOS

The red line in the arbiter slices represents the critical path. The series of AND and OR gates create a large amount of delay. As N grows, the critical path length of the carry chain, when synthesized in 45nm CMOS ASIC with the OpenCell NanGate library using Synopsys Design Tools, of a 1024-port round-robin arbiter consumes a time period of 66 ns. Hence, there is a need, as in a digital adder, a carry look ahead prediction that breaks down the chain to speed up the arbitration by introducing additional logic to compute the chain. The carry look ahead generator computes the carry output for the next *Lah* (look-ahead parameter) values and groups of N/Lah arbiters will compute the carry well before the carry goes through the entire chain. Table 3.1 shows the clock period (ns) of an N-port arbiter for different values of N and *Lah* when synthesized in 45nm CMOS ASIC.

The carry chains at the beginning and the end followed the design pattern of the arbiters presented in [82]. The significant improvement provided by the look ahead generator is also shown by the table. The highlights in red show the optimum carry look-ahead value for a N-port arbiter. The N-port arbiter was also implemented on NetFPGA Sume to evaluate the minimum clock period required on FPGA.

Figure 3.8(a) shows the benefit given by the look ahead generator. An 13$\times$ improvement is shown by the use of a carry look-ahead generator, consuming a clock period of less than 5ns. The arbiter implementation is also shown to be feasible on FPGA at almost 2-3$\times$ higher clock period in fig. 3.8(b). Hence, the scalability of the round-robin arbiters make them competitive hardware elements in high-port count hardware scheduler.

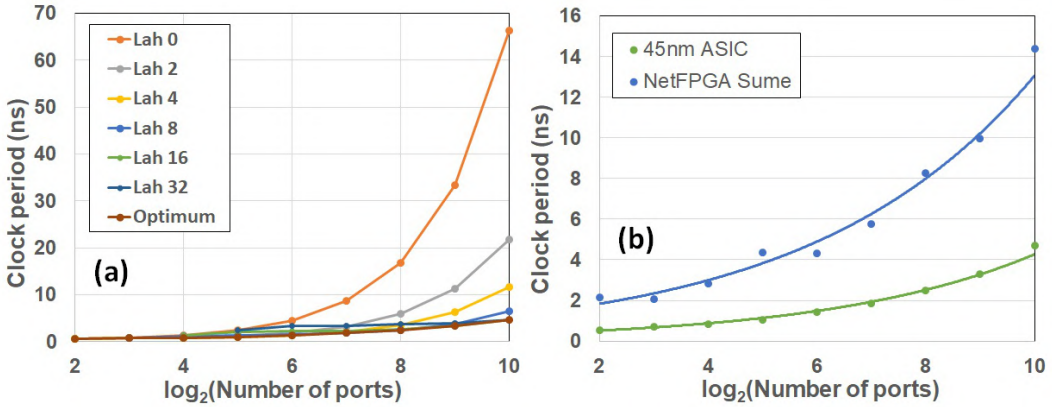


Figure 3.8: *N*-port Arbiter: (a) Different look ahead synthesis on ASIC (b) With optimum look-ahead on FPGA vs ASIC

3.4.5 Scheduler Hierarchy

As a blue print for the complete design, the initial hardware scheduler design can also be broken down into several sections. Figure 3.9 shows the different sections of the scheduler. Sections 1, 5 and 6 interface with the external communication links, while the other sections are responsible for scheduling and management. A total of 7 stages is shown in fig. 3.9 and each stage is explained in this section. Sections 2 and 4 are the two stages of arbitration mentioned before and they serve as the dominant logic in the hardware scheduler. Section 3 decides wavelength for fresh assignments and section 7 cancels already served requests.

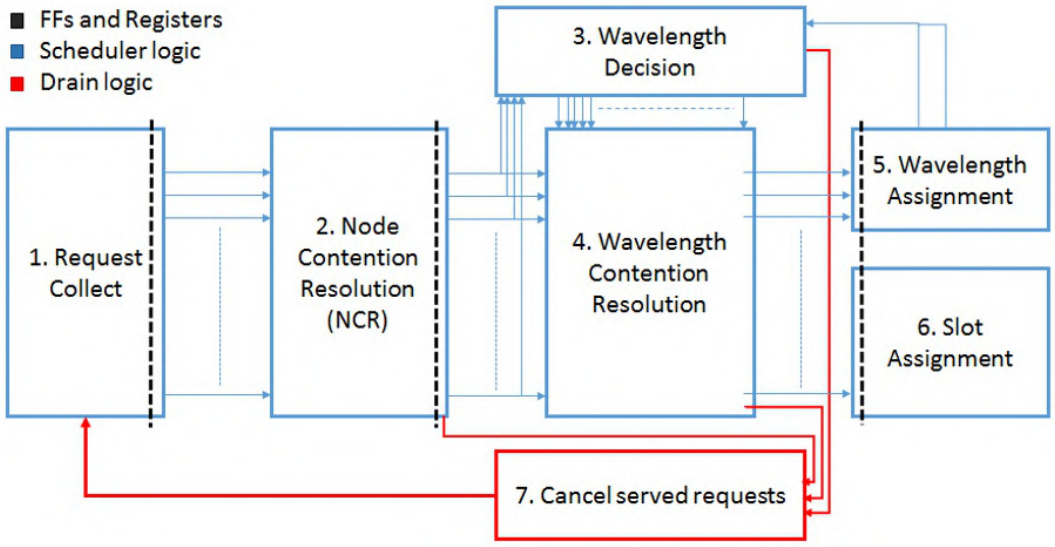


Figure 3.9: Scheduler logic breakdown

1. Request Collector: As can be seen in fig. 3.10, there are $N \times R$ flip-flop groups in the request collector. Each flip-flop group contains $[log_2 N + 1]$ D flip-

flops. The size parameter is not taken into account for the initial design. Hence, the total number of flip-flops used in this block are $N \times R \times [log_2 N + 1]$. The feedback mechanism in the 'cancel served requests' (section 7) logic controls which set of requests are to be processed by the scheduler for the current iteration (denoted by the pointer). The requests that queue in first are treated with the highest priority. The pointer shifts lower down the queue only after the higher priority requests have been granted.

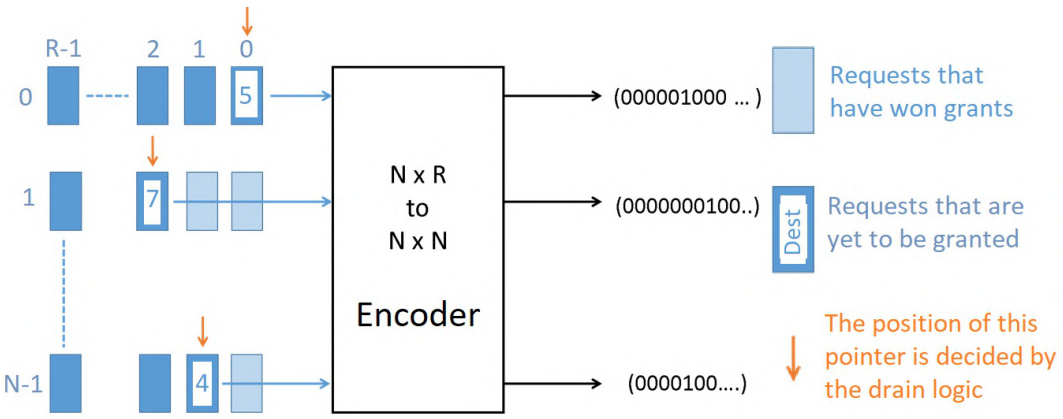


Figure 3.10: Request Collector Logic

The $N \times R$ to $N \times N$ encoder simply takes the $log_2 N$ destination signal, checks the valid bit, and makes one hot binary $1 \times N$ vectors. This is done because the allocator, in the contention resolution stage, contains arbiter elements that treat one bit at a time. The request collector block is expected to scale well, as it has a large array of flip-flops and a simple combinational logic.

2. Node Contention Resolution: In this stage, these requests are fed as N-bit vectors to round robin arbiters. Each N-bit arbiter takes an N-bit request input and gives out an N-bit grant output. The contention for the same destination node by multiple source nodes is resolved by the allocator (group of arbiters). Once resolved, the N-bit grant is converted back to the destination value, $log_2 N$ (and the valid bit) and stored in FFs.

Figure 3.11 shows the flow of the node contention resolution module. The dominant logic of this section is the allocator. The minimum clock period required for this logic is expected to be very close to that of an N-bit arbiter; because the N instantiations of the N:1 arbiters all run in parallel. Optimal look-ahead values that give the best timing results were used for the arbiters in this module.

3. Wavelength Decision: The previous stage in the scheduler dealt with resolving contention between node pairs. Now that the contention is resolved, the steps that follow in the scheduling algorithm allocate wavelengths and slots. The

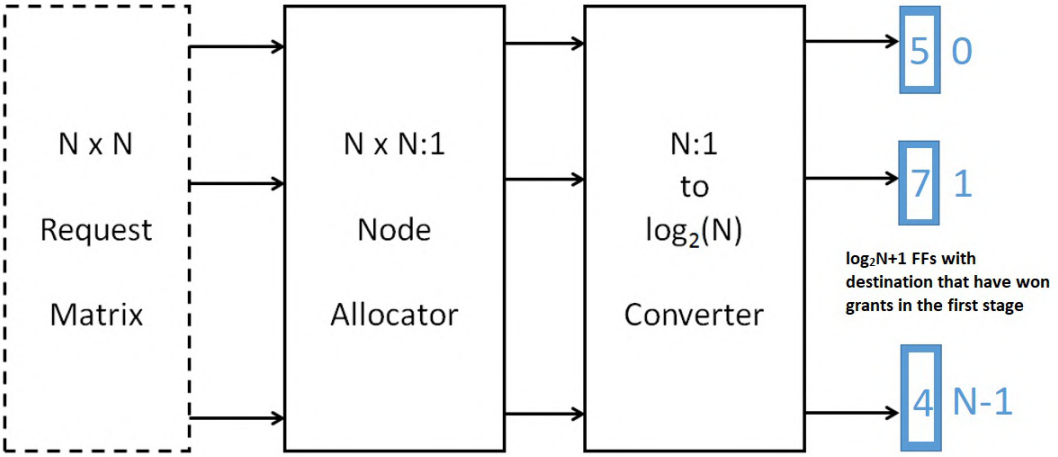


Figure 3.11: Node Contention Resolution Allocator

wavelength checker module checks the source and destination registers for wavelengths assigned by grants from previous iterations. As shown in fig. 3.12, the checker controls the wavelength request matrix; there are only five cases for the winning request from each node.

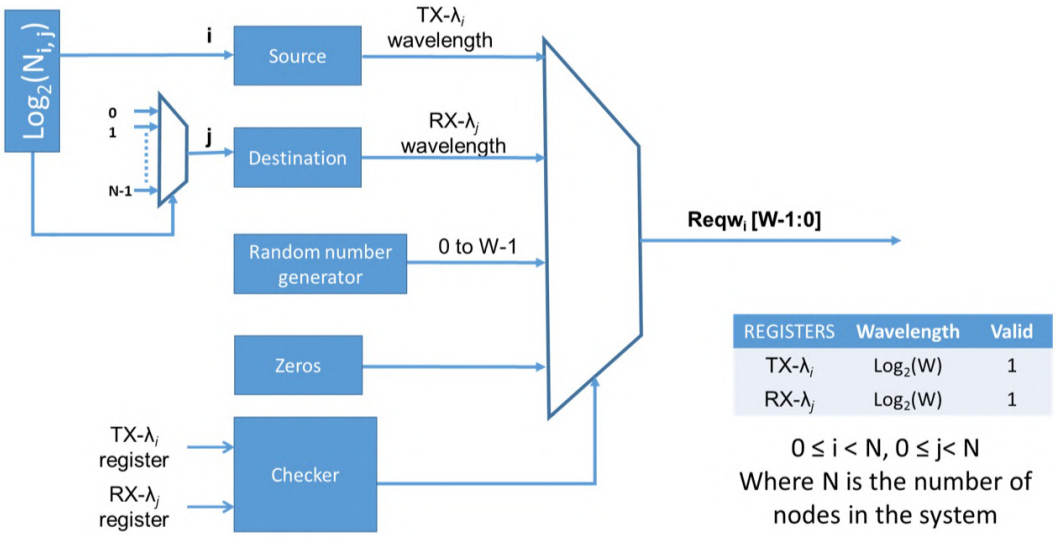


Figure 3.12: Wavelength Decision Module

In fig. 3.12, the wavelength decision is based on a multiplexer. The logic checks for multiple cases, which are discussed here.

Case 1: The transmitter of the source node, TX_i, and the receiver of the destination node, RX_j, are not assigned any wavelengths. A random number generator is used to form the wavelength request vector, $Reqw_i$.

Case 2: The transmitter of the source node, TX_i, is assigned λ_i, and the receiver of the destination node, RX_j, is not assigned any wavelength: The transmitter

wavelength of the source node is assigned to the wavelength request vector, Reqwi.

Case 3: The transmitter of the source node, TX_i, is not assigned any wavelength and the receiver of the destination node, RX_j, is assigned λ_j: The receiver wavelength of the destination node is assigned to the wavelength request vector, $Reqw_i$.

Case 4: Both the transmitter of the source node, TX_i, and the receiver of the destination node, RX_j, are assigned different wavelengths: Since the transmitter and the receiver is tuned to different wavelengths, this request cannot be granted in the current epoch; the $Reqw_i$ is filled with zeros. The 'cancel served requests' logic also uses this case to remove the request from the collected requests from each node.

Case 5: Both the transmitter of the source node, TX_i, and the receiver of the destination node, RX_j, have been previously tune to the same wavelength, λ: The hardware assigns transmitter wavelength λ to the wavelength request vector $Reqw_i$.

4. Wavelength Contention Resolution: In the wavelength decision stage, the wavelength request matrix is created. The non-contending node pairs can be assigned a wavelength provided that the transceivers of the source and destinations are not tuned to different wavelengths in previous iterations. The wavelength request matrix, $Reqw_i$, is fed into the wavelength allocator, as multiple nodes could request the same wavelength channel.

Figure 3.13 shows $N \times Reqw_i$ request coming in from each node and $N \times Gw_i$ grant going out; each $Reqw_i$ and Gw_i is $[W-1:0]$ vector. The wavelength allocator has W N:1 arbiter elements. In the wavelength contention resolution stage, multiple source nodes (transmitters) contend for multiple wavelengths. The wavelength allocator resolves these contentions fairly in a round robin fashion; the same wavelength is not granted twice for multiple sources in the same iteration.

5. Wavelength Assignment: The wavelength assignment module holds the

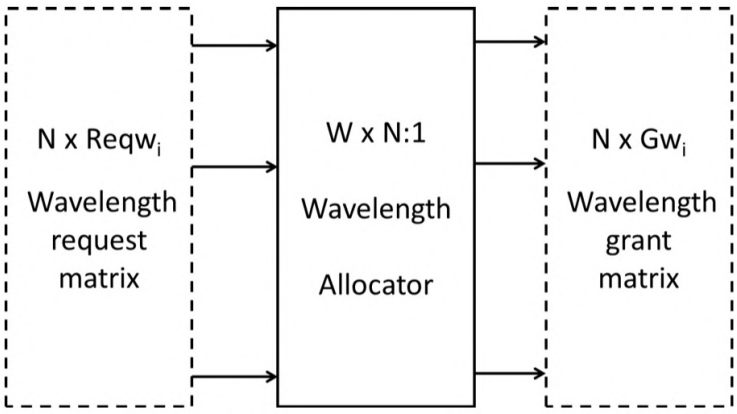


Figure 3.13: Wavelength Contention Resolution Module

transmitter and the receiver registers; once the winning grants from the 'Wavelength Contention Resolution' module are computed, the grant vectors are used to update the wavelength registers. The wavelength grant is updated provided that there are enough slots (T) available to communicate and if the maximum number of transceivers tuned to the same wavelength (M) is not reached.

Figure 3.14 shows the wavelength assignment block; the output grant matrix created in the wavelength contention resolution module is used as an input matrix to this block. As discussed before, the wavelength registers have the wavelength structure indicated in the figure: $log_2W + 1$ bits; each node has two wavelength structure registers: $TX - \lambda$ and $RX - \lambda$. Figure 3.14 shows how registers are updated for one node. There are N (=1024) instantiations of the $TX - \lambda$ and $RX - \lambda$ registers. W slot counters and $W \times M$ counters are implemented in this module; the counters not only count, but also generate pointers. If the maximum number of slots or the maximum M value (maximum limit on the number of transceivers using the same wavelength) is reached, the counters trigger a 'wavelength full' parameter that does not update the transmitter and receiver register values. If the counters are not full, the transmitter (source node, i) wavelength and the receiver (destination node, j) wavelength are updated. In the example in 3.14, the block is shown to update the transmitter and the receiver registers to use wavelength λ_7 for communication.

6. Slot Assignment: Once a wavelength has been assigned to a node, a slot is also assigned based on a slot pointer. The slot assignment module simply registers the slot number and request number granted for each node after it gets winning grant. Each grant structure FF, as discussed in the theory section, is a group of $log_2R + log_2L + 1$ D flip-flops; they store the request number, the slot

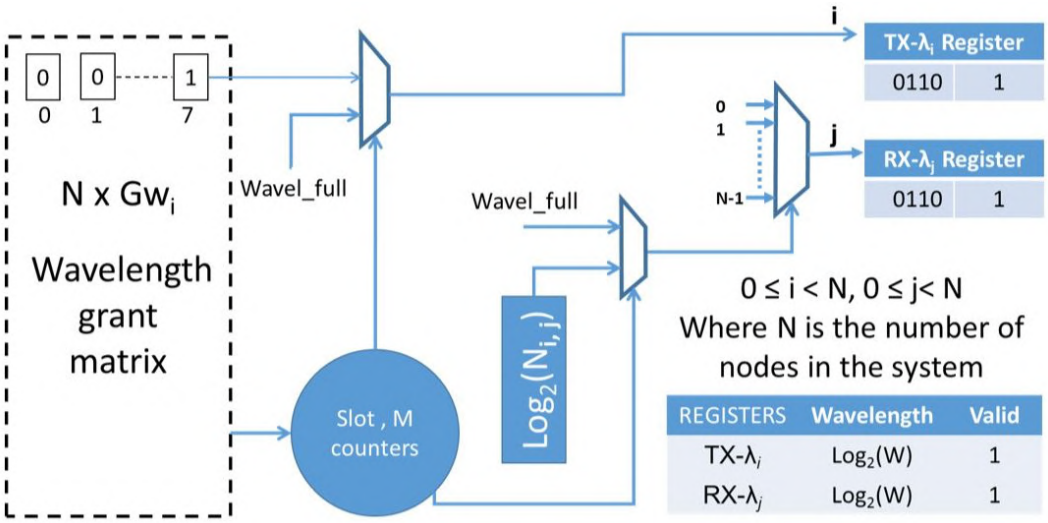


Figure 3.14: Wavelength Assignment Module

number that have been granted and the valid bit. This module uses N finite state machines (FSMs), one for each node and the pointers are updated once it is filled.

Figure 3.15 shows the working of the slot assignment module. On the right, the figure shows two states of the FSM implemented: IDLE and COUNT. The FSMs shown increment the count (shifts the pointer) once it sees a grant output, always pointing at an empty slot. On the left, the figure shows the winning grants from the wavelength contention resolution block; once a winning grant is seen, the relevant request number is also registered. For every node, the slot number and the request number are stored. The wavelength assignment module sends signals to the tunable transceivers and the coherent receivers, while the slot assignment module sends all its signals to the nodes in the network. The counter sends a stop signal to this module; when the time runs out for the number of scheduler iterations, the slot grant structures are sent back to the relevant nodes.

7. Cancel Served Requests: This module has a feedback mechanism from the two stage scheduler that cancels the requests collected initially. All the requests that have been granted and that cannot be granted in the current epoch are removed from the request registers so that the same requests are not granted twice and are not continuously contend with other requests in future iterations. The 'cancel served requests' logic has to wait for the contention resolution and the wavelength checker to finish computing the grant and the possibility of communication in this epoch respectively for this module to work. The critical path length is usually the longest combinational path between two registers. Hence, for close to realistic timing results the requests collection FFs, the node contention FFs, the wavelength contention resolution block and the relevant sections of the wavelength checker block have been replicated in this module.

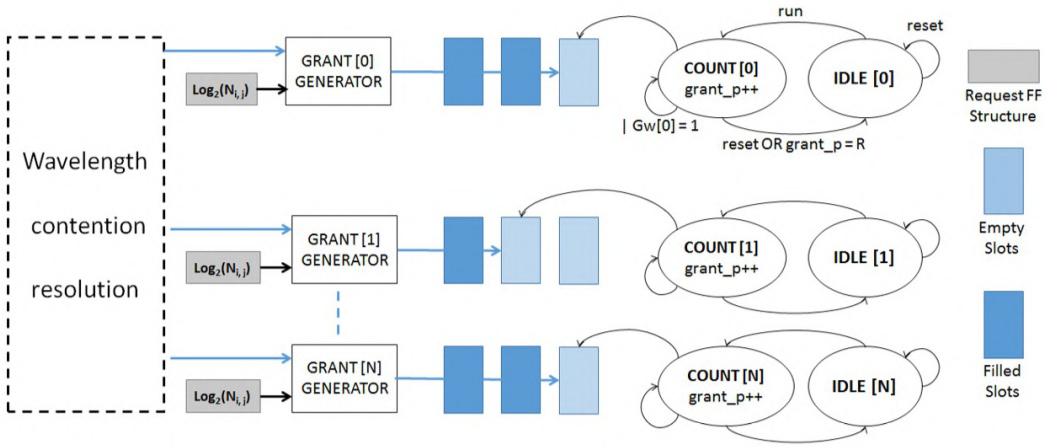


Figure 3.15: Timeslot Assignment Module

Figure 3.16 shows the working of the 'feedback' logic. There are three signals, in particular, that the 'cancel served requests' module has its' focus on:

1. The grant from the node contention resolution,
2. The grant from the wavelength contention resolution and
3. The different wavelength signal from the checker: If the wavelength decision algorithm gives an indication that the transmitter and the receiver of interest have already been assigned different wavelengths in a previous iteration, the request is cleared from the request collector FF but retained for the next epoch.

3.4.6 Scheduler Performance Analysis

3.4.6.1 Simulation environment

The scheduler design functionality was verified on modelsim and, interfacing a hardware testbench with MATLAB, the hardware model was analysed as to how it performs with uniform distributed network traffic. A 10ns clock period was assumed for these simulations and the system parameters were $M \leq 25$, $T = 50$ timeslots, $W = 80$ wavelength channels, $R = 4$ requests/server and $N = 1024$ nodes. The demand matrix was a many seeds of a random single epoch request written onto ROM file. Multiple runs were required to simulate the different loads of traffic. In this analysis, the number of requests were balanced to the resources available in the network to create the correct measure of throughput using a load factor (Lf), as shown in equation 3.9. Modelsim reads the request for each epoch and produces the performance analysis results shown in fig. 3.17 below.

$$N \times R \times Lf = W \times T \implies 1024 \times 4 \times Lf = 80 \times 50 \tag{3.9}$$

Figure 3.17(a) and (c) show the performance results using different metrics, resource utilization and number of grants generated; number of grants reflects how many out of 4000 requests are granted and resource utilization indicates the % of slots used (also out of 4000 slots). The plot in figure 3.17(a) suggests that the rate

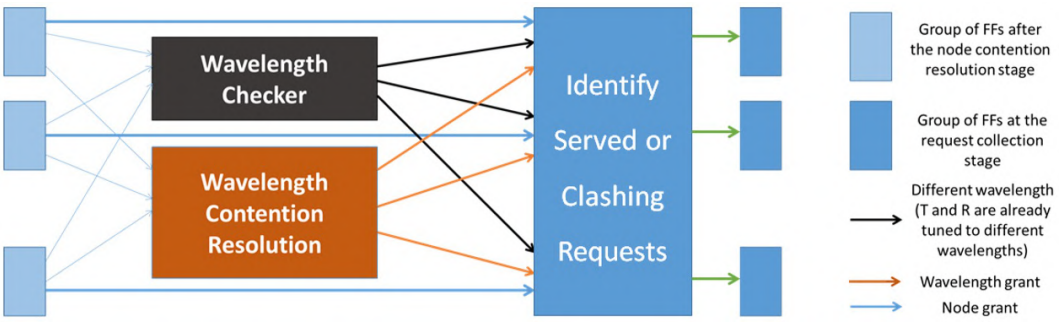


Figure 3.16: Cancel Served Requests Module

of grant generation decreases with the number of iterations. There are two regions in this curve: the linear region (up to 10-20 iterations for 10-100% traffic loads) and the saturated region (beyond 30 iterations).

Figure 3.17(b) shows that, in less than 5 iterations, all 80 wavelengths are granted to the 1024 port scheduler. This steep increase in the wavelength build shows that the scheduler performance can be improved by increasing the number of wavelength channels, W, made available. However, a significant increase in the value of W also increases the $(W \times T)$ grant matrix. At high values of W, an increased amount of channel resources is made available, which could also imply an increased complexity in the transceiver. Although reducing the value of the number of slots, T, could improve resource utilization, the current value of T (50) already implies a slot size of 120 bytes at 25 Gbps; reducing this further would mean a significantly bigger slot would be required for every grant.

3.4.6.2 Grants and resource utilization

The scheduler performance can be improved by using an additional size parameter for the scheduler, which specifies the number of slots required for communication. This could minimize the number of requests, R, whilst also significantly increasing the throughput. resource utilization can be improved by increasing the number of requests, R, as more set of new (valid) requests are reviewed until the linear region is active. Finding the optimal value of R to obtain maximal throughput is important.

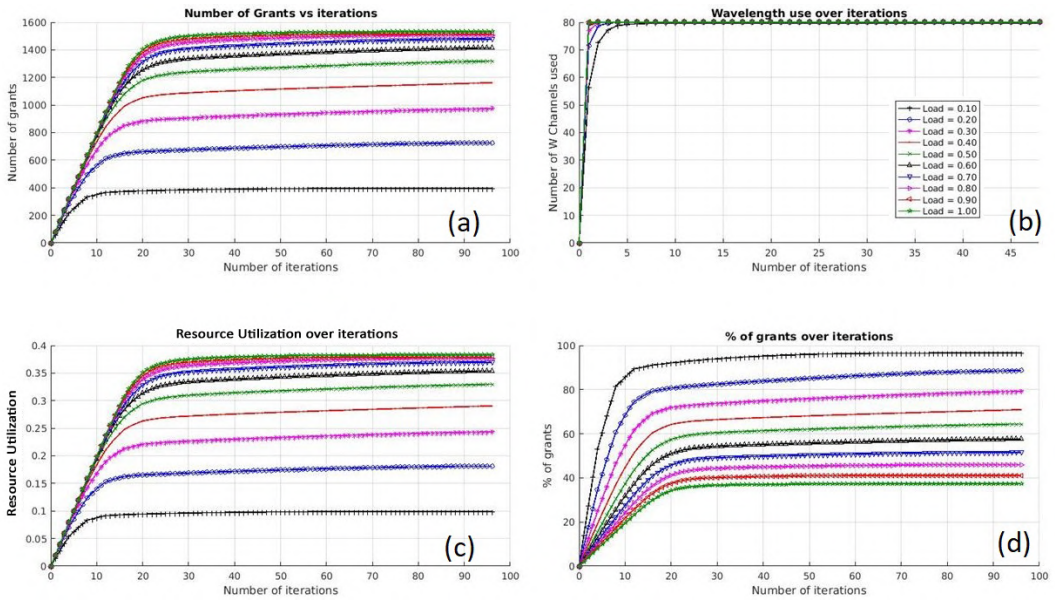


Figure 3.17: Performance analysis results for a scheduler system: (a) Number of grants generated (b) Number of wavelengths used (c) Resource Utilization (d) % of grants vs I iterations for traffic loads

Increasing R to extremely high values decreases the percentage of grants. The value of R has no effect on the dominance of wavelength locking in later iterations and hence, does not improve the performance significantly.

When less number of wavelength assignments are made, the scheduler is in the linear region, when the random wavelength assignment logic is dominant. Since every wavelength assignment invalidates a set of requests wavelength locking logic, where the wavelength selection is made based on previous assignment, becomes dominant; this corresponds to the saturated region. This analysis shows that any scheduler computation after the 30^{th} iteration does not generate much grants, implying that an increase in the value of the number of iterations, I, has no effect and slots are left unfilled; these scheduler iterations could be used elsewhere. From the performance evaluation above, it can be seen that the total number of input requests is 4000. Even under the highest load value of 100%, where every request is valid, the maximum number of grants generated for an epoch is less than 1600, resulting in a resource utilization of 40%. A highly efficient schedule should have a resource utilization that is close to 100%.

3.5 Multi-star topology

As the number of iterations efficiently used by the parallel scheduler is less than 30% and only a single-sized request was supported, the network built could not prove as an efficient optically circuit switched solution. Hence, in this section, I delve into the weaknesses in order to rectify and transform the scheduling and network architecture. The modifications introduced here evolve not only the scheduler, but also provide some key architectural benefits. A develop a neatly pipelined scheduler is developed and the area of using a single scheduler for multiple star coupler based topologies is explored, how they perform and compare it against software based heuristics is critically analysed and evaluated.

3.5.1 Transceiver upgrades

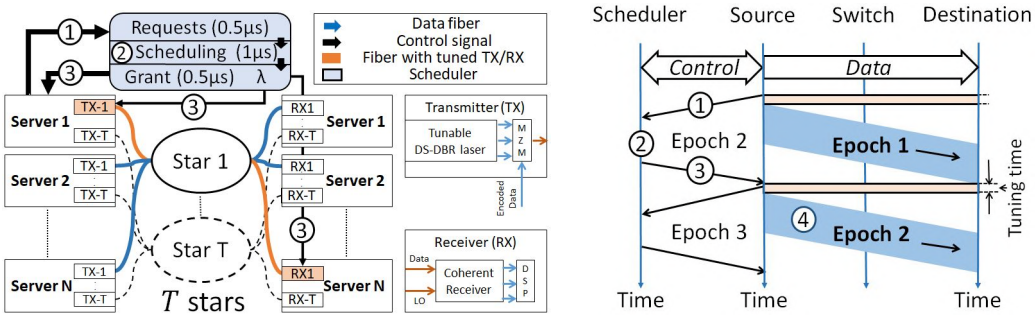
In the updated architectue, a simplified DSP hardware for the coherent receiver with reduced complexity, cost and energy consumption is proposed [25]. A 50 GHz ITU grid within the optical C-band is used to ensure minimal cross-talk and hence, each transmitter can use one of 89 possible wavelengths. As there are 1000 nodes to service but only 89 wavelengths, time division multiplexing is adopted to maximize throughput. In this work, a DP-QPSK modulation is proposed for transmission to achieve a line rate of 100 Gbps (symbol rate of 25Gbaud/s).

To reduce the complexity of scheduling and switch configuration, a globally synchronized switch is proposed with a synchronous tuning time. Recent work has

shown that 90% of 89 available wavelength channels (80 channels) complete tuning in 40ns [5]. Hence, 80 wavelength channels and 40ns synchronized tuning time is proposed for configuring all transceivers and receivers connected to the switch. Whilst considering both scheduler complexity and the effects of switch configuration time on throughput, a 2μs useful communication time (non-tuning time, defined as epoch) is proposed, bringing the overhead down to 2%. Each node sends requests to the scheduler one epoch in advance and awaits response and switch reconfiguration.

At 100 Gbps, the 2μs epoch allows each wavelength channel to communicate 25kB (kilobytes) effectively. The use of 100 time-slots in the system targets the transfer of 250 bytes per slot, which corresponds to the overall median packet size across various data center workflows [83]. As the globally synchronized optical switch can carry the bare payload of the packet, the effective packet length is slightly more than an average Ethernet packet . At each transmitter, smaller packets ($<<$250 bytes) to the same destination are collected for effective slot usage. The fast time-slot switching with a 1-bit guard band, minimizing TDMA overhead, was previously demonstrated [25].

3.5.2 Network Architecture upgrades



(a) Central Scheduler controlling the proposed switch data plane

(b) Dataflow: data and control plane

Figure 3.18: A 1000-port Optical Circuit Switch Architecture

The parallelism of the scheduler algorithm allows for the computation of wavelength map across multiple stars, within the required time. Hence, a switch architecture with $T = 4$ star coupler cores that connect 1000 nodes is proposed, where T is the number of transceivers per node. Using 80 wavelengths per star an optical switch with a shareable bandwidth of 32 Tbps is proposed, at a line rate of 100 Gbps. The overview of the switch architecture is shown in Fig. 3.18a.

3.5.2.1 Harmony

The control communication and scheduling take place one epoch in advance to data communication. Although scheduling is done out-of-band, the optical circuit switch relies on high resolution synchronization and harmony, as shown in fig. 3.18b, between the control and the data plane. Not only is the scheduler required to have strict time awareness, but also the nodes and transceivers in the system. Fig. 3.18b shows the sending of requests, schedule of grants and distribution of wavelengths happening within the 2μs scheduling deadline. The (1), (2), (3) and (4) in both figures in fig. 3.18 correspond to request collection, scheduling, grant (wavelength and timeslot) distribution and eventual communication respectively.

3.5.3 Control network upgrades

3.5.3.1 Control communication protocol

As discussed in the previous sub-section and shown in Fig. 3.18b, the nodes communicate the requests to the scheduler one epoch in advance. The scheduler has 2μs to collect the requests, compute schedules (wavelength configuration and grants) and communicate the schedules. The first 0.5μs is allocated for request collection. The hardware can only accept a fixed number of requests, constrained by the scheduler memory and switch bandwidth. For a given epoch, the control plane requires the total request size to be less than or equal to the bandwidth available in the optical switch. Once, requests are collected, a time-based control triggers the scheduler into the scheduling state, taking into account the data plane limitations. Within 1μs, the parallel scheduler performs multiple iterations (up to 138), as detailed in the next sub-sections, to achieve maximum throughput. Storing the grants at the end of every iteration becomes costly, requiring a large buffer (buffer size = grant size$\times$138000 bits). So, grants are sent out to the nodes at the end of every iteration in a synchronous manner to minimize hardware resource utilization. The wavelength map is not distributed while scheduling, as previous wavelength assignments need to be considered in scheduler subsequent iterations. Hence, the last 0.5μs is used for wavelength distribution.

The system parameters proposed for the optical switch are shown in table 3.2. To communicate the requests to the scheduler, each node uses the request structure (RS) in equation 3.10. Every node can communicate up to R requests per epoch to the scheduler. This means that the total request size is 112,000 bits, as deduced from equation 3.11.

$$RS = log_2 S + log_2 N + Valid \tag{3.10}$$

Variable	Parameter	Value
N	Number of ports	1000
R	Number of requests per port	8
S	Maximum number of slots per request	7
W	Number of wavelength channels	80
L	Number of time-slots per channel	100
T	Number of stars	4
M	Maximum number of transceivers with same wavelength per star	25
I	Number of iterations	138
D	Number of small control stars	64
B	Equal number of nodes per small star	16

Table 3.2: 1000-port Optical Switch - System Parameters

$$Request\ Size = RS \times R \times N \tag{3.11}$$

Grants are distributed as they are generated by the scheduler, during the 1μs scheduling time, to minimize resource utilization. The grant structure (GS) used by the scheduler to communicate the grants, at the end of each iteration, is shown in equation 3.12. A checker on the scheduler only permits a grant structure to be sent, if it is valid; If not, a time-slot value of greater than L is sent. The iterations of the scheduler I is divided into T sections for scheduling and hence, the nodes will identify the transceiver they must use based on when (on which value of I) the grant was generated. From equation 3.12, each grant structure is found to use 13 bits. The total size of grants generated is 1,794,000 bits, as computed from equation 3.13.

$$GS = log_2R + log_2L + log_2S \tag{3.12}$$

$$Grant\ size = GS \times N \times I \tag{3.13}$$

The wavelength structure (WS) that is generated by the scheduler contains transmitter and receiver wavelength maps for all T transceivers in each node, as shown in equation 3.14. From equation 3.15, the total size of the wavelength map generated is found to be 60,000 bits.

$$WS = 2\log_2 W + Valid \tag{3.14}$$

$$Wavelength\ map\ size = WS \times N \times T \tag{3.15}$$

3.5.3.2 Control plane architecture

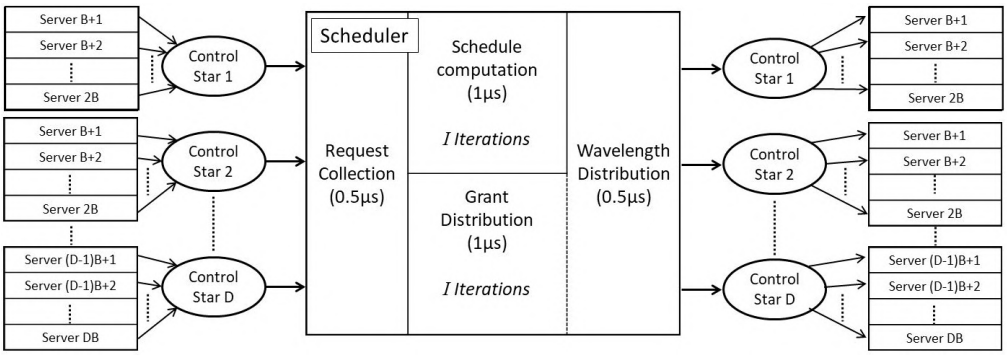


Figure 3.19: A 1000-port Optical Circuit Switch Architecture

A control plane for the 1000-port optical switch that uses D bi-directional star-coupler networks is proposed, shown unfolded in Fig. 3.19, for communicating control information. A group of B nodes are assigned a specific transmission and reception star. Within these control stars, nodes use a single wavelength laser source in their transmitters and direct-detection receivers, greatly reducing the cost of deploying the control plane. Each node within a star is assigned specific time-slots to transmit requests and receive grants or transceiver wavelength tuning information.

The size of the total request size and the wavelength map are relatively smaller than the grants generated by the scheduler. The distribution of grants during the scheduling stage requires the use of high line rate, and thereby, defines the control plane bandwidth.To ensure a scalable scheduler, distributed storage of grants over N nodes was proposed. Although this technique removes the need for storing the grants at the scheduler, a new challenge is introduced. A total of 1000 grant structures ($13\times N$ bits), although only 80 successful, are generated at the end of every scheduler iteration. As to be discussed, the high-speed scheduler takes 7.2ns to perform each iteration (defined as *clock period*). $13 \times B$ bits are required to be communicated within 7.2ns, forcing the grants to be distributed at an an extremely high bandwidth.

Increasing D increases the number of pins on the scheduler and reduces the number of nodes within each group. On the other hand, increasing B increases the bandwidth required per small star. The total bandwidth of a small star as the

bandwidth for each grant transmitter channel can be defined as shown in equation 3.16, where BW_{TX} is the bandwidth of each grant transmitter channel and BW/D is the bandwidth per small star.

$$BW/D = BW_{TX} \times B = B \times Grant/clock\ period \tag{3.16}$$

From equation 3.16, using B = 16, the grant structure as 13 and the clock period as 7.2 ns, the bandwidth required per small star is found to be 28.9 Gbps. Taking these parameters into account, the total control bandwidth required across all stars, CBW, is found to be 1.8Tbps as shown in equation 3.17.

$$CBW = D \times 28.9\ Gbps = 1.8\ Tbps \tag{3.17}$$

As current SERDES technology is able to produce high speed serial signals at 28-32 Gbps, I propose $D(=64)$ small stars and $B(=16)$ nodes within each star. However, a slight lenience on the control plane bandwidth could be achieved by implementing buffers on the scheduler and spreading out grant and wavelength distribution over 1.5 μs. This reduces the control bandwidth to 1.6 Tbps and allows a line rate of 25 Gbps in each star. The control plane architecture could be simplified and the bandwidth per star can be reduced with the use of WDM and wavelength filters at each receiver.

3.5.4 Scheduler Upgrades

3.5.4.1 Principles

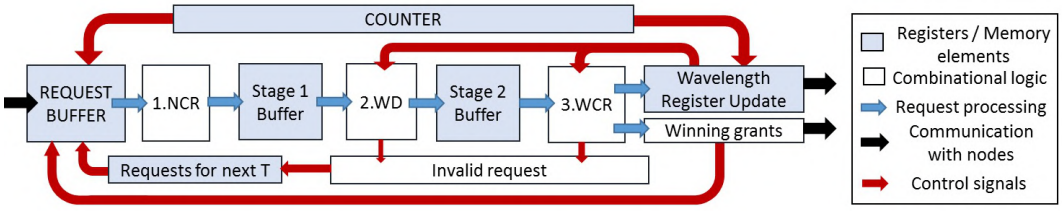


Figure 3.20: Multi-star topology: Optical Circuit Switch Scheduler Architecture

Fig. 3.20 shows the new scheduling algorithm for the scalable fast optical circuit switch. The encoder and the node contention resolution stage combine to create one continuous combinational logic block (1.NCR). As the wavelength decision block involves a series of multiplexers before arbiters, combining these two stages to form a longer combinational chain could cause delays and longer critical paths. Hence, wavelength decision block is divided into a separate block (WD).

The third stage contains the wavelength contention resolution, which is followed by wavelength and timeslot allocation.

An additional upgrade to the scheduler is the granting of slots per request. All requested slots are allocated consecutively if the wavelength has enough timeslots available for the epoch. If not, available slots are granted and the request size is reduced for retry in another star. Once all slots are granted, the served requests are cancelled. If clashing requests are made where a specific transmitter-receiver pair have been assigned different wavelengths, the same pair can retry for a connection in the next star.

Another upgraded feature in the hardware is the temporal parallelism that is introduced. Pipelining the hardware means that the scheduler is as fast as the slowest of these three sub-modules. As shown in fig. 3.20, the invalidation of requests is now separated into two phases as pipelining means that the second stage has no knowledge of the last cycle.

The scheduler aims to send out timeslot grants at the end of every iteration instead of storing them as before. The wavelength configuration needs to be registered for future checking and hence, are sent just before the beginning of the epoch.

3.5.4.2 Hardware verification

The verification of the design of large port-count scheduler is highly important. Hence, same sets of requests were given to the software model of the scheduler (designed in MATLAB) and the hardware scheduler, as shown in fig. 3.21.

As shown in fig. 3.21, Matlab was used to generate traffic and create requests as required by the scheduler and export into a file. Modelsim was used to simulate the hardware RTL design. Modelsim reads the request file and generates a grant file in response with timeslot and wavelength assignments. Matlab reads the output of the hardware and verifies the data with the software model for various loads and seeds. Once a 100% match was achieved, the hardware design was implemented on 45nm CMOS ASIC using Synopsys synthesis tools.

3.5.4.3 Implementation on ASIC

The scalability of one N:1 round robin arbiter was previously demonstrated. As parallelism does not significantly affect the critical path, N and W parallel round robin arbiters are used in the NCR and WCR sub-modules respectively. Hence, the scalability of these sub-modules are determined by the scalability of the arbiters. The wavelength decision (WD) stage, however, depends on the scalability of the checker, which uses simple scalable N:1 multiplexers to perform the checking (critical path of 60ps per 2:1 multiplexer).

Once verified, the scheduler was synthesized on ASIC, using the Synopsys

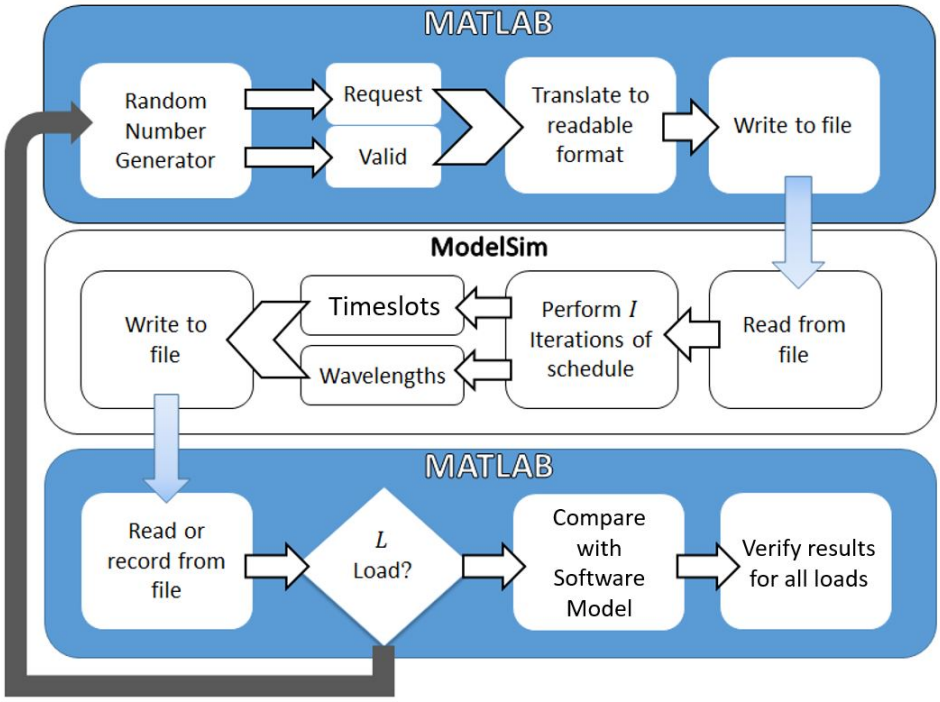


Figure 3.21: Verification of software model with hardware implementation of algorithm

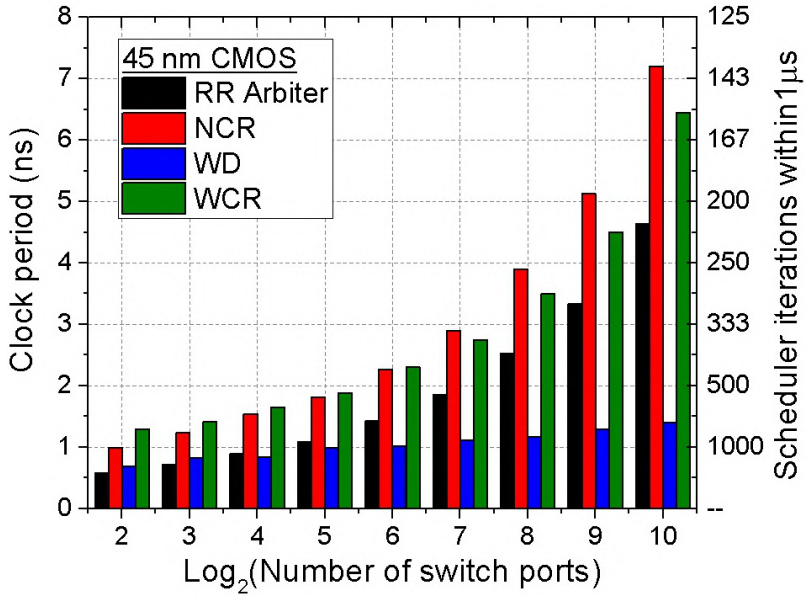


Figure 3.22: Scalability of N-bit NCR, WD and WCR sub-modules on 45nm ASIC, where N is the number of server

tools and the 45nm CMOS Nangate Opencell library.

Figure 3.22 demonstrates the scalability of the scheduler as the sub-module ASICs are scaled to support 1000 ports on a 45nm CMOS Opencell library. The NCR module has the longest of the critical paths, requiring a minimum clock speed

of 7.2ns. This means that a total of 136 iterations can be performed by the scheduler in 1μs. The area consumed by the scheduler ASIC is 52.7mm^2, including the buffer, without including the SERDES in the control plane.

The programmable priority encoder in the NCR and WCR arbiters allow specific wavelength resources to be assigned to specific flows (large or small), if necessary. The scheduler also introduces reconfigurability into the system, by allowing serial allocation for each star. The scheduler can be programmed to choose any set of requests to be processed over any iteration or star. The hardware exploits parallelism spatially and temporally. Parallelism to $N = 1000$ ports is used in all blocks. Pipelining is used to isolate each module and reduce the critical path length of the large combinational logic.

3.6 Summary

In this chapter, the concept of a coupler-based high-radix optical switch is shown. The optical transceivers at the end-nodes employ ultra-fast wavelength and timeslot selection, O(200 ns), to reconfigure the network. A centralized controller that can reconfigure the transceivers within 1μs computation time was shown. Once parallel round-robin arbiters are found to be scalable with optimal look-ahead values, the design of each of the working modules inside the parallel hardware scheduler were showcased. The hardware design was synthesized on 45 nm CMOS ASIC and shown to consume 7.2 ns for a 1024-port scheduler (slowest pipeline stage). The parallel scheduler generated up to 80 (=W) grants per iteration and being able to perform up to 138 iterations in 1 μs, it reached a point of saturation within 20 iterations achieving 35-40% grant and resource utilization. Hence, as a majority of the iterations remain unused, a multi-star topology was adopted. The upgrades required in the data plane, control plane network and scheduler to support more stars are discussed. Moreover, the automated verification environment built using hardware (Modelsim) and software (Matlab) tools was outlined.

Chapter 4

Hardware vs Software Performance

In this chapter, the performance of the synthesized parallel hardware scheduler is compared against serial software based allocation techniques in order to evaluate scheduling bottlenecks. Minor modifications are introduced to the proposed hardware design to increase performance boost. The performance of the scheduling unit in terms of request per node, wavelength-timeslot utilization, packet retries are checked in order to evaluate how performance can be gained or boosted. Other network concerns and bottlenecks, that severely degrade performance are analysed. For more details on the performance of the parallel hardware scheduler and its comparison with software scheduling heuristics, the reader can refer to the papers published in IEEE HotInterconnect 2017 [81].

4.1 Software based scheduling heuristics

A demand matrix is created by the request buffers that collect up to R requests from each of the 1000 nodes. Every scheduler algorithm must consider 8000 requests (as $R = 8$) and allocate resources: 80 wavelength channels, 100 time-slots for each of the 4 stars over one epoch $(W = 80, L = 100, T = 4)$. In this section, the theory behind the scheduling algorithms that are designed for the 1000-port microsecond timescale optical switch is presented. All scheduling algorithms use the parameters in table 3.2 and take into account the data plane limits of transmitters per wavelength, M, number of wavelength channels, W, and time-slots, L.

4.1.1 Serial resource assignment heuristics

The serial algorithm stores requests in a circular queue. Following the traditional dynamic wavelength assignment techniques used in wavelength routed optical networks (WRONs) [86], the algorithm considers one node pair in any given clock cycle. The serial algorithm is conscious to the data plane limits and it updates registers in each iteration. The pseudo-code for the serial algorithm is given in Appendix section A. The serial heuristics first check if the source and destination node

already share a wavelength across any star. If both the source and destination nodes are unassigned, an available wavelength is allocated in a random, least loaded (LL) or least used (LU) fashion. The random technique randomly selects a star and randomly selects an available wavelength (line 8 in pseudo-code). The least loaded technique scans across all stars to find the wavelength that uses the least number of slots (line 9). The least used considers wavelength usage and finds the least used wavelength with the minimum number of requests granted (line 10).

Clock cycles resulting in a non-assignable request are not given a grant and the corresponding request is added back to the queue for future epochs. As the serial wavelength assignment heuristics check for pre-assignments first, retrying failed requests does not improve throughput. The serial algorithm has to work on $N \times R$ elements in the demand matrix, resulting in 8000 clock cycles. The time allocated for scheduling in this system is 1μs, thus the serial scheduling scheme would require an unrealistic clock speed of 8 GHz. Hence, the serial scheduling algorithm is not a realistic solution to adapt and is only used for performance comparison.

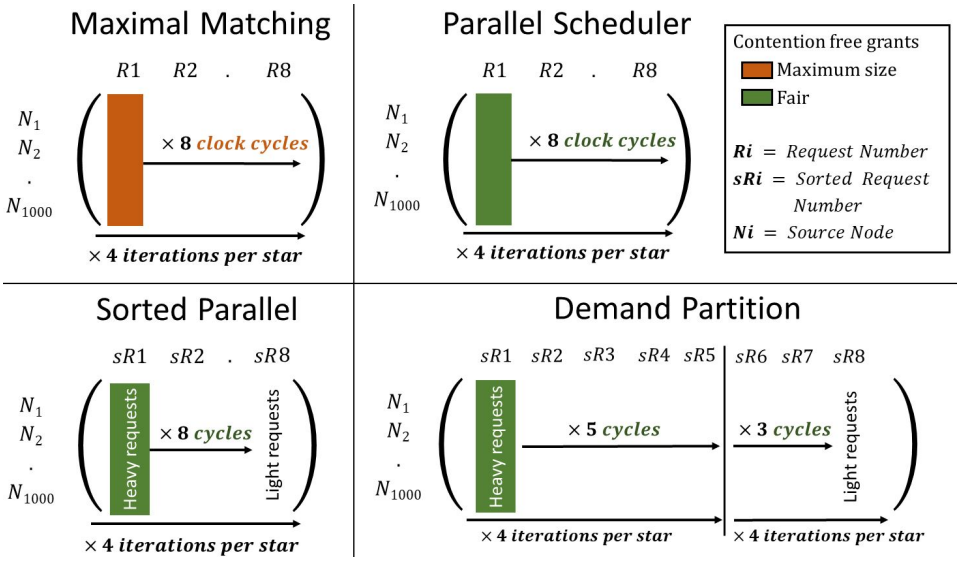


Figure 4.1: Demonstration of clock cycle usage in Parallel scheduling algorithms

4.1.2 Maximal matching

The Maximal Matching algorithm concept originates from *maximal matching* in graph theory and aims to achieve identify the heaviest requests for granting timeslots in every clock cycle. A weighted bipartite graph is created with one set of source nodes, N_1, requesting access to another set of destination nodes, N_2, R edges from each node and each edge has an average weighting of S_{avg}. Finding the minimum *maximal matching* (MMM) is an NP-hard problem [87]. Not only does the scheduling need to find the MMM, but also map the matching to available channel

resources. To simplify this problem, firstly, one out of R requests are considered in the order of request number, *Ri*, as shown in figure 4.1. Secondly, the algorithm is applied to each star independently and sequentially; conflicting requests are isolated for scheduling in subsequent stars. The Maximal Matching scheme selects W heaviest requests in every clock cycle using sorting elements. The hardware implementation of maximum weighting algorithm requires high-speed sorting elements. Although traditional sorting algorithms are highly serial and follow a compare and rearrange methodology, a few parallel sorting hardware algorithms have been proposed [88, 89, 90]. Although high performance is achieved by these algorithms, their complexity makes them non-scalable for packet time-scale scheduling. The Maximal Matching algorithm also uses the same number of clock cycles (128) as the other parallel scheduling schemes to serve as a comparison metric.

4.1.3 Sorted Parallel

The Parallel Scheduler scheme works on requests in the order of request number, *Ri*, as shown in figure 4.1. The Sorted Parallel scheme proposes to sort the order of requests to *SRi* (Sorted Request), re-arranging them, starting from the heaviest or highest slot size request to lightest or lowest slot size request, as shown in figure 4.1. Once sorted, the Sorted Parallel uses the Parallel Scheduler hardware mentioned above. Unlike sorting elements in Maximal Matching scheme that require $N = 1000$ element sorters, these sorters only require small $R = 8$ element sorters in parallel. Maximal Matching scheme requires sorting to be done every clock cycle but Sorted Parallel needs to sort only once before scheduling. The 8-port parallel merge hardware sorter in [90] is shown to achieve a clock speed of 6ns when implemented on a Virtex-7 FPGA, which can be further reduced on ASIC. However, multiple passes (clock cycles) are required to sort and a rough time estimation of ~O(NlogN) passes is given for merge sorters. An alternative is to sort the requests at the nodes, where aggregation takes place, before sending the requests.

4.1.4 Demand Partition

Here. a scheduling technique that uses the same parallel hardware scheduler discussed in the previous section and the same three submodules (NCR, WD and WCR) is presented. The Demand Partition scheme as shown in figure 4.1 also works on sorted requests, *SRi*. As shown in figure 4.1, the Demand Partition scheme gives priority clock cycles to the first 5 *SRi* heavy requests and after 4 iterations, the second partition with 3 *SRi* lighter requests are considered. The partitioning values of (5,3) were chosen after a simple performance test showed to give higher throughput for (5,3) partitions compared to (6,2) and (4,4) partitions.

4.2 Performance Analysis

In this section, the performance of the algorithms used in the hardware scheduler is evaluated and compared with the algorithms discussed in section 3. Software models of all the algorithms were made in Matlab and used for performance evaluation. The system parameters used for each of these scheduling algorithms are shown in table 3.2. In all the experiments performed, 100% network workload corresponds to requesting the complete switch capacity, 31.3725 Tbps, taking into account the capacity lost to tuning time (40ns). A uniform random traffic was used to evaluate the performance of the schedulers. Under uniform random traffic, each node requests a pseudorandom destination node with a probability of 1/N.

4.2.1 Scheduler performance

As discussed in section 2, the scheduling algorithms receive up to $R = 8$ requests from each of the $N = 1000$ nodes, forming an $N \times R$ demand matrix (8000 requests). The scheduler is required to process all requests and compute grants (time-slots) and wavelength tuning map within 1 μs. The hardware implementation of the scheduler permits 138 clock cycles within 1 μs. In the experiment, the performance of the parallel scheduling algorithms over 128 clock cycles is evaluated. The serial wavelength assignment schemes, discussed in Section 3.1, process one node pair at a time and hence would take 8000 cycles to compute all the requests.

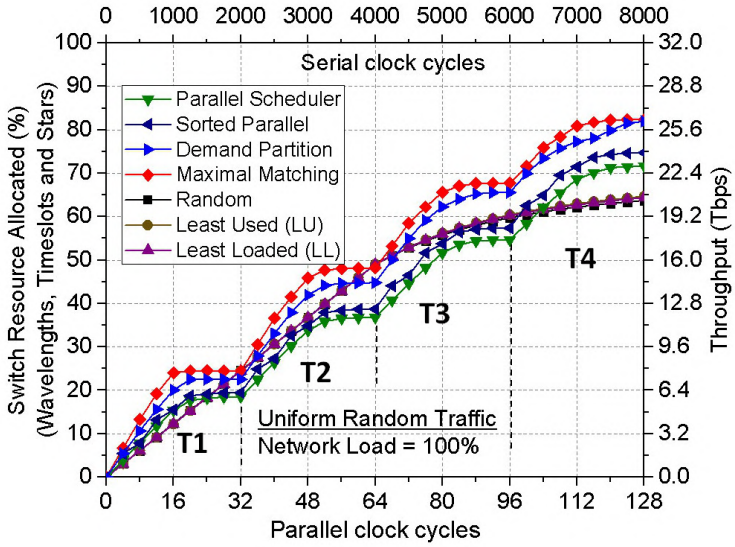


Figure 4.2: Switch resource allocation/throughput achieved by serial/parallel algorithms over clock cycles/time when scheduling grants for one epoch

Figure 4.2 shows how the scheduling algorithm perform allocation for one epoch over time for a network load of 100% (requesting all ~32 Tbps). In figure 4.2, the parallel scheduling algorithms (Parallel Scheduler, Sorted Parallel, Demand

Partition and Maximal Matching) use 128 clock cycles (bottom x-axis) to compute a schedule. The traditional software based serial algorithms (Random, Least Used and Least Loaded) use up to 8000 clock cycles (top x-axis). The software approach allocates available resources (wavelength and time-slots) across any star in a given clock cycle. In contrast, the parallel algorithms schedule for one star (denoted by $T1, T2, T3$ and $T4$ in figure 4.2) at any given clock cycle and failed requests are considered in consecutive star(s). At 50% clock cycles, all three serial schemes, exploiting resources across all stars, achieve 10-12% better performance than the parallel scheduler and the sorted parallel schemes. However, both these parallel schemes achieve 8-10% better performance at the end of the scheduling process, owing to contention resolution arbiters.

Maximal Matching, achieving the highest throughput in every iteration, schedules with 10% higher throughput compared to the parallel scheduler scheme. At 128 clock cycles, the sorted parallel scheme, as expected, marginally outperforms the parallel scheduler by 3-4%. The demand partition scheme, giving priority to heavier requests, achieves almost 26 Tbps throughput at 128 cycles, as high as the Maximal Matching schedules, outperforming other parallel schemes.

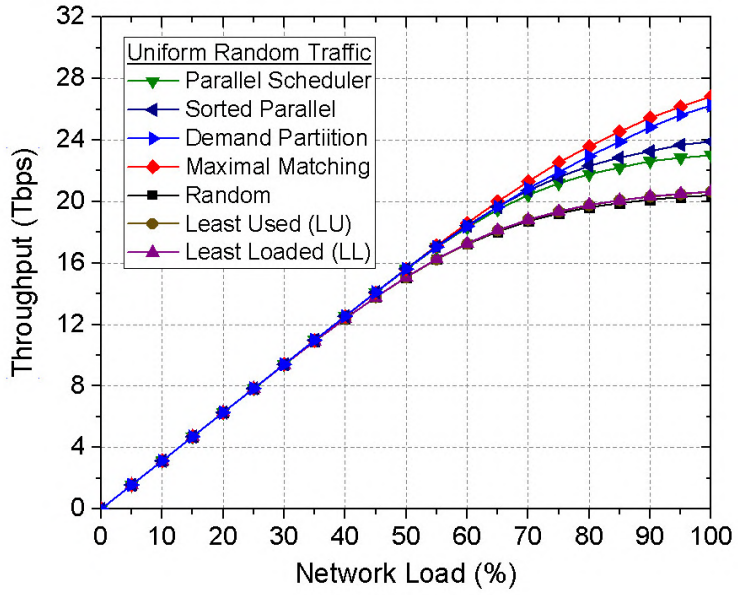
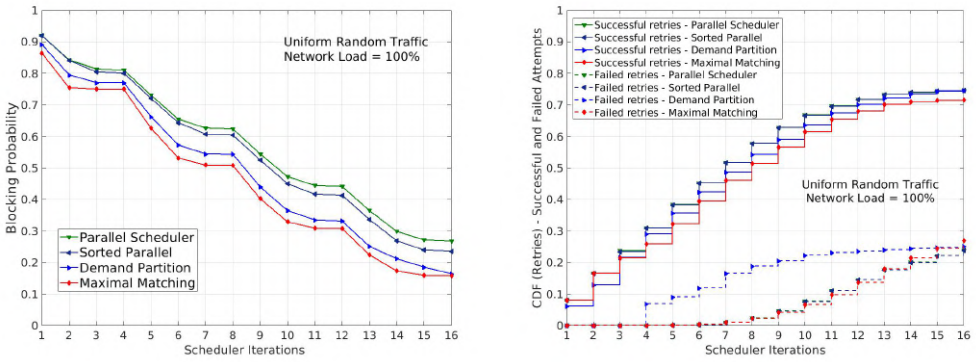


Figure 4.3: Scheduling/matching performance of scheduling for increasing load under uniform random traffic

Figure 4.3 shows the performance of the scheduling algorithms under increasing uniform random traffic network loads. Up to 50% network loads (requesting ~16 Tbps), all scheduling schemes are close to achieving a 100% throughput. At network loads above 70%, the parallel scheduling schemes outperform the serial

algorithms.

4.2.2 Parallelism and Iteration



(a) Parallel algorithms: Multiple iterations reduce blocking probability

(b) Cumulative distribution function (cdf) of request retries

Figure 4.4: Matching performance of scheduling algorithms under different request scenarios

The serial algorithm is performed but once across the entire demand matrix (8000 clock cycles) to compute the schedule. However, parallel scheduler algorithms only compute W grants out for every N requests and perform multiple retries for failed requests. As shown in figure 4.1, the parallel algorithms considers $N = 1000$ requests in one clock cycle and needs $R = 8$ clock cycles to process all 8000 requests and perform one iteration. Hence, 128 clock cycles allows the parallel algorithms to perform up to 16 iterations and up to 16 'retries' (1st iteration here is 1st retry), a maximum of 4 retries in each star. Focusing on parallel algorithms, the effectiveness that iteration offers to the scheduling algorithms is reviewed. Figure 4.4a shows how the allocation blocking probability reduces in consecutive iterations for a schedule with 100% network load (worst-case scenario). As shown also shown in figure 4.4a, the 16 iterations comprise of retries in $T = 4$ different stars. As shown in figure 4.4a, the first iteration of the requests (in any star) gives the highest reduction in blocking probability. This is expected because the first iteration in every star (iterations 1,5,9,13) primarily has less contradictions and secondly, high resource availability. The blocking probability in the parallel algorithms reduce from ~90% to less than 30% for the (sorted and non-sorted) parallel scheduler schemes and to less than 20% for the demand partitioning and maximal matching schemes, owing to parallelism that allows these iterations.

The scheduling algorithms can invalidate retry requests in the current star, if the request is found to contradict existing wavelength assignments or if the requested

resource is not available. Figure 4.4b shows the cumulative distribution function (CDF) of successful and failed request retries for 100% network load traffic (worst-case scenario, same traffic as in figure 4.4a). In figure 4.4b, the sum of successful (solid line) and failed retries (dashed lines), gives the CDF of all request retries. The graph shows that, at the end of 16 iterations, the retry success rate is 72% for Maximal Matching and 75% for all other parallel algorithms. The demand partitioning scheme gives priority to the first $R = 5$ heavier requests in all 4 stars. According to figure 4.4b, up to 32% (0.8 out of 0.25 in figure) of the request retries in the demand partitioning algorithm that fail in all 16 iterations, have failed after 12 retries (from iteration 4 onwards). These are low-priority, lighter (small slot size) requests (from $R = 6, 7, 8$) that were not granted any wavelengths in 3 or 4 (multiple) retries of each star. Inversely, in the Maximal Matching, Parallel scheduler and Sorted Parallel algorithms, not giving priority to request numbers, have an average of 5 failure retries with a uniform spread.

4.2.3 Resource usage: Wavelength, time-slots and stars

Figure 4.5 shows how each of the scheduling schemes, allocate wavelength, time-slot and star resources when the network load is 100% (worst-case scenario) under uniform random traffic. The colour bar on the left shows how many transceivers are scheduled for using the same wavelength and time-slot across all stars. The more the number of transceivers per wavelength, the higher the resource utilization, throughput and scheduling performance.

Figure 4.5 shows that on all scheduling schemes, except serial random, almost all 4 transceivers are scheduled (orange) up to slot number ~50. The Maximal Matching algorithm, shown to achieve the relatively maximum throughput in 100% network load (worst case scenario), uses at least 1 transceiver per wavelength per star even when slot number is 100. The serial least loaded scheme has the highest number of empty slots with no transceiver transmitting data (dark blue). The top row of figure 4.5 shows that the empty slots are reduced when changing the algorithms from Parallel Scheduler to Sorted Parallel and from Sorted Parallel to Demand Partition, showing improvement in throughput. The slots used in the demand partition scheme are mostly grants from heavier (big slot size or packets) requests and hence, occupy more slots in comparison to other parallel algorithms.

4.2.4 Increasing Requests per Node

Increasing the number of requests per node, R, increases node contention whilst scheduling and increases the invalidation of request retries, reducing throughput significantly. As shown in figure 4.1, the request numbers are denoted by *Ri* and *SRi* for non-sorted (Maximal Matching, Parallel Scheduler) parallel schemes and

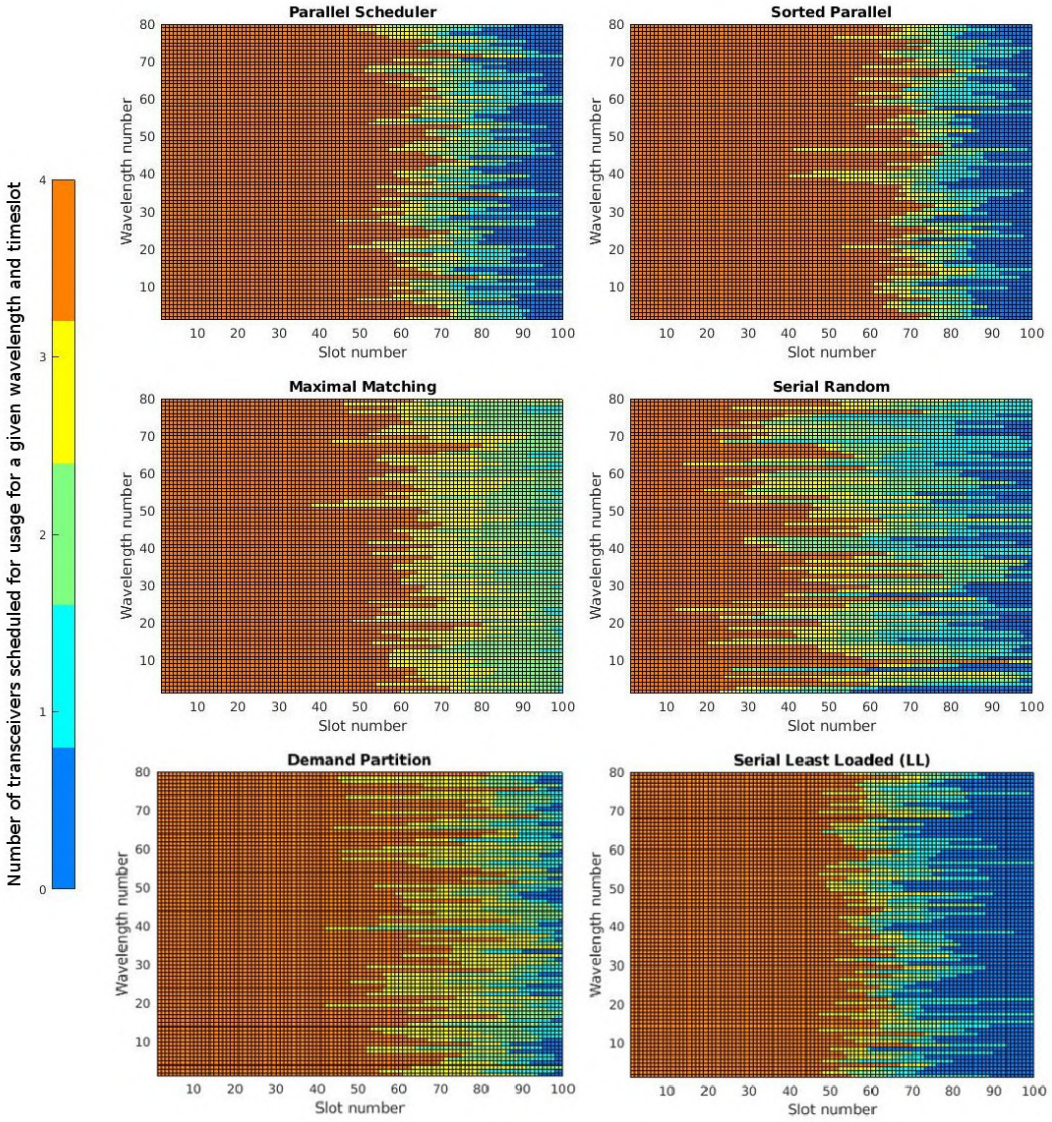


Figure 4.5: Scheduled Resource Utilization: Number of transceivers using wavelength and time-slot across all stars

sorted (Parallel and Demand Partition) parallel schemes respectively. All scheduled grants for $R = 8$, with 100% network load (worst-case scenario), were grouped by request numbers (*Ri* and *SRi*). The overall blocking probability across all iterations for increasing request numbers, (from $R1$ or $SR1$ up to *Ri* or $SRi = 8$), is evaluated and shown in figure 4.6.

Figure 4.6 shows the increase in the blocking probability, as the number of requests per node, R, increases. For serial algorithms, there is negligible blocking in the system for up to 4000 requests. As serial scheduling algorithms have access to all stars in a given iteration, priority is always given to lower request num-

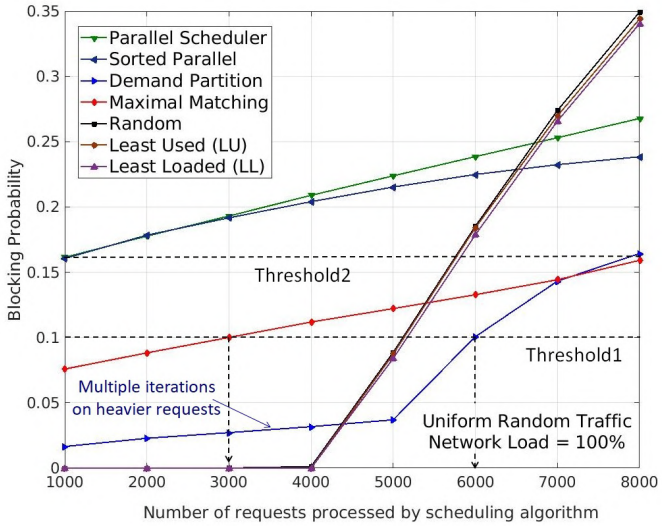


Figure 4.6: Blocking probability of each scheduling scheme vs number of requests in a given epoch

bers ($R1, R2, R3$ and $R4$), increasing blocking probability up to 35% for the second set of 4000 requests. By setting a blocking probability threshold of 16% (Threshold2 in figure 4.6), the Parallel Scheduler and the Sorted Parallel algorithms can only process 1000 requests, while the Maximal Matching and Demand Partition schemes can process all 8000 requests. This is because Maximal Matching gives priority for heavy requests (high request slot size) in each Ri, while the Demand Partitioning gives priority to all request numbers that have all heavy requests. At a blocking probability threshold of 10% (Threshold1 in figure 4.6), the Demand Partition schemes can process 6000 requests, while the Maximal Matching can only process 3000 requests. The Demand Partition scheme performs its initial iterations on heavier requests, $SR1$ to $SR5$, and hence, is shown to achieve increased throughput. However, there is a drastic increase in blocking probability, from 4% to 16%, for the last 3000 requests that are treated with low priority in the Demand Partition scheme. The results presented above are for a scenario where the maximum number of requests per node, R, is fixed to 8. A scheduler hardware can only accept a limited number of requests, which is constrained by the bandwidth of the control plane and scheduler memory. However, to test the limits of the scheduling algorithms and the switch data plane, the performance was evaluated for increasing values of R.

In every epoch, the scheduling algorithms receive a maximum of $N \times R$ requests. Figure 4.7 shows the effect of increasing the maximum number of requests per node. In these measurements, the maximum number of slots requested, S, was varied to ensure that 100% network workload capacity (~32 Tb/s) is requested

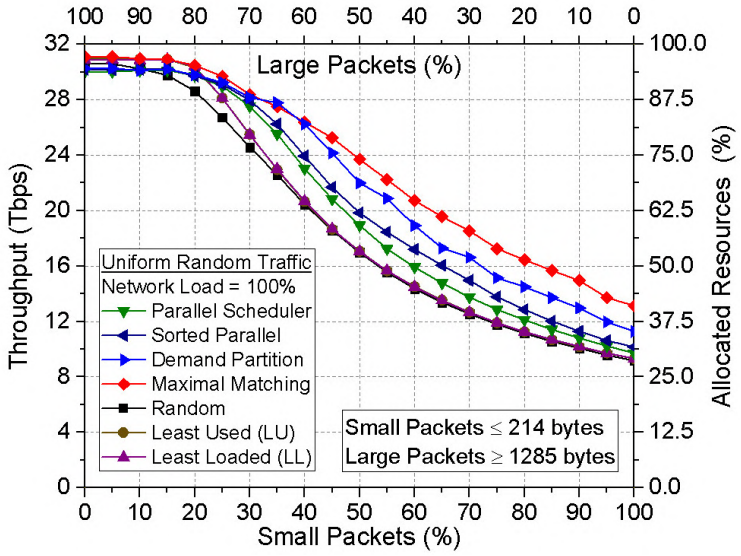


Figure 4.7: Performance of scheduling algorithms on increasing number of requests per node

even if R varies. In figure 4.7, 95%-5% large-small packets implies $R = 1$ (one large request per node), requesting up to 63 slots (average of 32 slots), while 0%-100% large-small packets correspond to $R = 20$ requests, each requesting 0 or 1 slot. Lower numbers of requests thus represent a high concentration of elephant flows, where each request is for large number of slots. Conversely, high numbers of requests correspond to a high concentration of mice flows, where many small destinations per node are requested within one epoch. When R increases to high values, scheduling algorithms experience more contention and are only allowed to perform fewer iterations, degrading the scheduling performance. The Maximal Matching algorithm has the highest performance relative to other scheduling algorithms, achieving ~45% throughput even when $R = 20$. The scheduling performance significantly drops to below 37.5% for smaller packets and $R = 20$, for all hardware implementable scheduling schemes and conventional software schemes, with the demand partition scheme achieving ~36%.

4.2.5 Increasing Wavelength Channels

Increasing the number of wavelength channels, W, increases the overall capacity of each star. From a scheduling perspective, as each clock cycle produces W grants, an increased throughput is scheduled every clock cycle. However, in future clock cycles, requests between two nodes that have already been assigned different wavelengths are invalidated and retry invalidation rate also increases.

Figure 4.8 shows the blocking probability of the scheduling schemes, when the number of wavelength channels is increased for a fixed request size, 32000 slots,

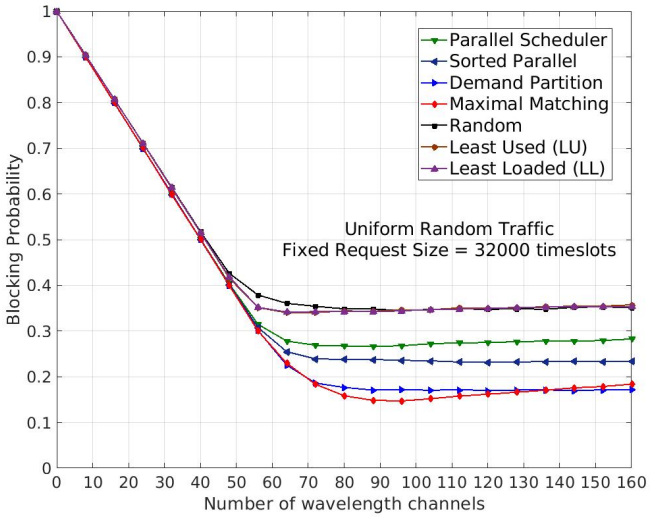


Figure 4.8: Scheduling algorithms blocking probability on increasing number of wavelength channels

under uniform traffic. The figure shows that the blocking probability is 100% when there are no wavelength channels and significantly reduces to less than 40% when the wavelength channels are increased. As shown by the behaviour of the parallel schemes in fig. 4.8, after a certain increase in the number of wavelength channels (beyond (~88 channels), the blocking probability stabilizes. This means that for all scheduling heuristics, with a fixed request size, $W = 60 - 90$ is the best number of channels to use to have maximum usage of minimal resources. The serial schemes have decreased blocking probability beyond ~64 wavelength channels. Hence, the scheduler algorithms, giving priority to wavelengths over time-slots, limits its usage to 'optimum' number of wavelength channels. This can be improved by finding the optimum number of wavelengths for the algorithm and limiting wavelength and time-slot grant assignment to W_{opt} per clock cycle.

4.3 Network Concerns

There are some key concerns with the optical circuit switched network proposed in this chapter. In this section, these concerns are highlighted; moreover, how these affect the network performance and how they can be changed to improve the overall scheduling and networking are discussed. These are then addressed to, in the next chapter, by introducing a novel network, transceiver architecture and scheduling heuristic.

4.3.1 The waiting scheduler

In the control plane architecture, the scheduler has a waiting phase in every epoch when it collects requests/send grant. During this phase, the scheduler is idle and simply waiting for collection and sending to be complete. This dedicated idling time reduces the efficiency of epoch usage. In the 2μs epoch, only 50% (1μs) is used for scheduling while the remaining 50% (1μs) is mainly waiting time. To reduce such inefficiencies, the control plane network must also be pipelined. In other words, there should be no idling stage for the scheduler. The scheduling for the current epoch (stage 2) must take place while the previous epoch grants are being sent (stage 3) and next epoch requests are being collected (stage 1). The time taken by the slowest of the pipeline stages limits the delay of the entire pipeline stage. However, by reducing the distance from the servers to the scheduler, the delay due to propagation of sending grants/collecting requests(Stages 1 and 3) can be reduced. The best case latency of each slot communicated is at least the size of 1 epoch. Hence, the epoch size being the slowest pipeline stage (Stage 2), more on this discussed later, must be reduced.

4.3.2 Broadcast wastage: Limitations of network efficiency W/N

The broadcast-and-select network proposed is not only lossy in terms of optical power. The network efficiency (NE) is defined by the number of wavelength channel resources available in the network and the number of transceivers per star, as defined by the equation 4.1.

$$NE = W/N\ (\%) \tag{4.1}$$

Although there are N transceivers (servers/ToRs) per network, only W wavelength channels can be used in one epoch. Hence, in the architecture proposed with $T = 4$, 4000 transceivers and 320 wavelength channels. 1000 servers are equipped with 4 transceivers each and each transceiver can communicate at 100 Gbps (a total of 400 Tbps). However, the 320 wavelength channels allow communication of 32 Tbps. Hence, the total network efficiency is as low as 8%. This means that 100% of scheduling throughput (excluding tuning overhead) corresponds to 8% of network efficiency in the proposed architecture. The main reason to have such high bandwidth wastage is because W is significantly lower than N (i.e. $W << N$). Either the number of wavelength channels (W) has to be increase to the number of nodes (N) or the number of nodes (N) has to be reduced close to (W).

4.3.3 Multi-epoch latency analysis

The performance of the scheduling algorithm has only be evaluated for a single epoch. However, the latency has not yet been evaluated. Hence, the performance of the scheduling algorithm can only be critically evaluated by measuring the latency of the scheduler.

4.3.4 Wavelength locking

Every wavelength assignment is done in parallel. When a source-destination transceiver pair is assigned a wavelength, requests from the same source and to the same destination are bound to use the same wavelength. What this means to the scheduling algorithm is that when a wavelength is assigned, many requests are locked to request a specific wavelength. If contradictory wavelengths are requested within the same epoch, the requests are immediately buffered to request either in another sub-star if $T > 1$ or in another epoch.

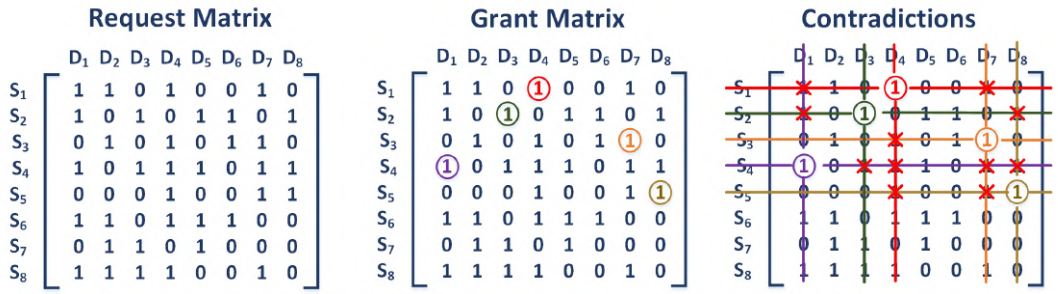


Figure 4.9: Example of requests, parallel grant selection and wavelength assignments creating contradictions in the first iteration

Fig. 4.9 shows the example of a 8×8 for an 8-port network with 5 wavelength channels (62.5% network efficiency). When parallel non-contending node pairs are assigned wavelengths in parallel, many contradictions are created as explained. In this example, the 5 parallel non-contending node pairs generate the winning grants and the selections are shown by the different colours in the 'Grant matrix'. Parallel wavelength assignments are made (shown by the coloured circles) and they are shown to cause 11 contradictions, marked by the red 'X's. The number of contradictions per iteration (or parallel assignment) is dependant again on the ratio of wavelength channels to the transceivers available in the network. As the number of wavelength channels increases, the number of contradictions also increases. Although the percentage of contradictions per iteration could be smaller as W/N is only 8% in the proposed network, repeated iterations can cause multiple contradictions to be created, increasing the blocking probability in the network and directly affecting the scheduling performance.

4.3.5 Scalability

The network proposed scales to 1000 end-points and the scalability of capacity is limited by the number of wavelength channel resources and transceivers/server. A data centre should be network is required to scale to more end-points to support the growth of the network. The splitting loss of the star-coupler does not let the network scale beyond a 1000 points, even with a coherent receiver. Hence, a new architecture is required to scale to more end-points.

The number of wavelength channels used defines the capacity. Although the parallel networks can increase capacity, increasing number of transceivers per server (T), this increases the cost and the power involved in employing such a network. Another approach would be to increase wavelength channels (W), which would result in employing more sophisticated transceivers. Hence, to enhance scalability, the network needs to be re-structured.

Chapter 5

PULSE: Towards sub-microsecond Circuit Switched Optical Networks

In this chapter, the aim is to showcase the PULSE architecture, which achieves high network efficiency, throughput and sub-μs scheduling latency. The chapter explores the benefits of (1) the novel OCS network architecture (2) the novel transceiver architectures that enable sub-nanosecond wavelength-timeslot tuning, (3) increasing wavelength channels ($W = N$) to improve network efficiency, (4) unlocking wavelength between each time-slot, (5) introducing a novel slot-level scheduling algorithm that can establish circuit connections every timeslot (20 ns) and (6) reducing epoch sizes to reduce latency. Then, the verification environment of the hardware elements is discussed. Once verified, the hardware scheduler algorithm is synthesized on 45nm CMOS ASIC using the OpenCell NanGate library in Synopsys Design Vision. The implemented design shows that the node contention resolution hardware sub-module has a critical path of 2.3 ns and 7.2 ns for a 64-port and 1024-port hardware scheduler respectively, consuming an area of 52.7 mm^2. Finally, the requirements that the PULSE architecture must meet are highlighted to facilitate a complete fast optical circuit switched network. The first requirement is the synchronization of all timeslots blades or transceivers in the sub-network. The second is the clock and data recovery (CDR) as all blades in the same sub-network are required to be in the same clock domain. For more details on the architectural principles and network characterization of PULSE and it's NsPU, the reader can refer to the papers published in PSC 2018 [91], ECOC 2019 or the papers submitted to OFC 2020 (Invited talk) and the JLT Special Issue on Optical Interconnects 2020 (under review).

5.1 Ultra-fast OCS Network

5.1.1 Data Plane Architecture

5.1.1.1 1-Plane

PULSE is a synchronous ultra-fast transceiver based architecture with tunable transceivers and passive star-coupler cores. The architecture is reconfigured by tuning the wavelength (WDM) and allocating the timeslot (TDM) at the transceivers to dynamically establish light paths (circuits). Fig. 5.1 shows the PULSE 1-plane OCS architecture, supporting up to x N-blade racks. The 1-plane PULSE architecture is an architecture, where only the number of transceivers (x) is used to scale the network.

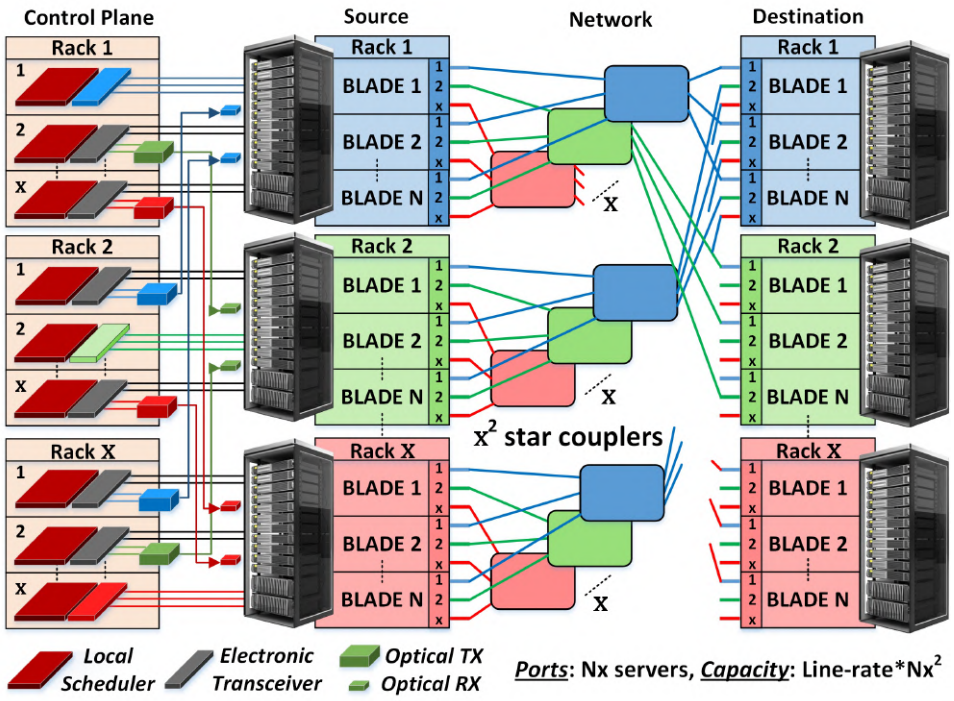


Figure 5.1: PULSE - Fast parallel OCS network architecture with distributed hardware schedulers

The data plane of PULSE consists of parallel OCS sub-networks, shown by right side of Fig. 5.1. Each sub-network has a passive star-coupler at its core, which creates a broadcast and select network [24]. There are upto x racks, where each rack contains N blades. Each blade houses up to x optical transceivers, where each transceiver connects a blade to a different star coupler and thereby, a different sub-network or rack. For connection establishment and eventual data transmission across the switch, the source transmitter and the destination receiver must both tune to the same wavelength and timeslot. The parallel OCS network proposed requires

x^2 N-blade star couplers and schedulers. The PULSE architecture scales to support up to Nx blades with a capacity of BNx^2, where B is the effective line-rate of each transceiver. The architecture described here is advantageous as it offers modularity - the control plane, data plane, synchronization, CDR locking and scheduling only have to scale each sub-network to the N blades; the scalability problem is defined by N, rather than Nx.

5.1.1.2 2-Plane

As described in the previous section, the architecture described above scales by increasing the number of transceivers a blade can support. As silicon integrated photonic (SiP) technology are becoming prominent, MBOs can enable the support of multiple transceivers per port. However, to avoid dependency of scalability on transceivers alone, another alternative is proposed. Figure 5.2 shows the 2-plane PULSE architecture, which is an architecture with two dimensions of scalability: number of clusters (p) and number of transceivers per server (x).

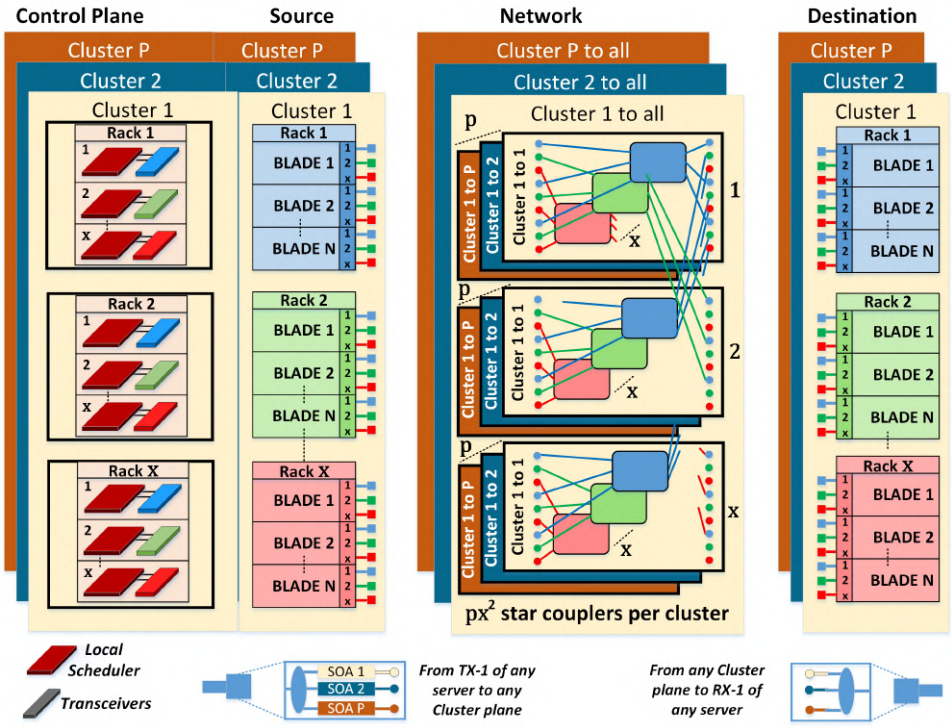


Figure 5.2: PULSE - Two dimension scaling

As shown in Fig. 5.2, another dimension of scalability is introduced with the aid of a split/broadcast and select unit after/before the transmitter/receiver. A $1:p$ splitter makes the transceivers available across p sub-networks instead of one while a semiconductor optical amplifier (SOA) gain-gate element allows for one of p sub-networks to be used every time-slot and compensate for the added optical loss. A

transmitter in cluster p connects to all destination planes (1-to-p) on the 'Cluster P to all' plane. A receiver in cluster P connects to a specific cluster (p) of each 'Cluster P to all' plane. Each transceiver is now equipped with an additional SOA gate to compensate for the split and select a different destination cluster and hence, enhance scalability of network. In addition to introducing a plane of clusters, the number of transceivers and the schedulers required to reach end blades is minimized by a factor of P. The transceivers and schedulers scale to px^2 instead of p^2x^2. However, this introduces harder constraints into the scheduler, requiring it to perform with increased efficiency as P sub-stars will have to share the scheduling iterations among them. Hence, the scheduling technique must also be modified to support this scaling solution.

The 1:p splitter/coupler that follows/precedes each transmitter/receiver, have a total insertion loss of $6log_2P$. In addition , the splitting loss of a star coupler is $3log_2N$. Assuming 64 blades per rack, N=64 split-sub-star and a split of P=8, a 36 dB loss is experienced. The SOAs that are selecting the path can boost the gain by 27 dB [92] bringing the optical power budget requirement to just 9 dB. This can be supported by either direct detect system and coherent receiver.

5.1.2 Control Plane Architecture

As shown by the left side of fig. 5.1, for every star coupler in the data plane, the control plane has a corresponding N-blade scheduler, which processes the requests for that particular star. The figure also shows that x local schedulers are hosted within the rack to minimize the round-trip propagation delay for the request-response handshakes. Each scheduler deals with the wavelength and timeslot allocation of one particular sub-network. The schedulers that handle inter-rack communication are equipped with optical transceivers to enable communication with the receivers of different racks.

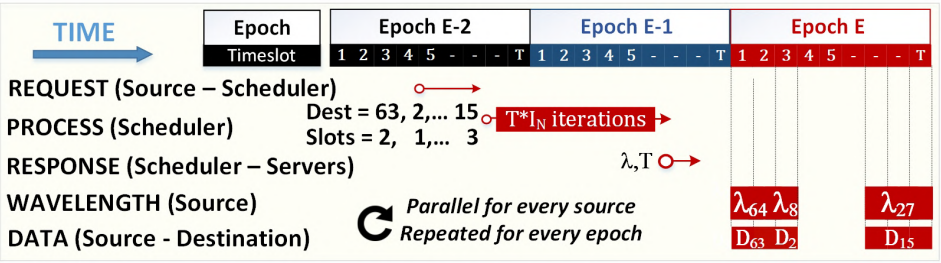


Figure 5.3: PULSE - Control Plane Requirement

Figure 5.3 shows the task required of the control plane from the data plane. As the demands generated by each blade requests destination and timeslots (size of

data), the control plane is required to give out grants of wavelength and timeslot resource. The corresponding request ID is also communicated so that the servers can send the relevant data in the designated timeslots. As shown, the reconfiguration computation time is limited by the epoch size. Smaller epochs reduce the number of scheduler iterations that can be performed and hence, reduce the throughput performance of the PULSE OCS system. Hence, the control plane is given a task to perform as many iterations as possible in a given epoch size and communicate grants in response to requests from each blade in a sub-network.

Each node sends requests to the scheduler a *few* epochs in advance (depending on the dominant propagation delay) and awaits response before reconfiguring transceiver wavelengths for the subsequent epoch. A control sub-network requires all 64 blades to be connected to the local scheduler with a 2 Gbps link per blade. Each request sent from each node contains has the following structure: requested destination (6 bits), slot size (5 bits) and epoch stamps (13 bits). Once received, the central scheduler stores requests from all blades in a 1.2kB buffer, which can store up to R(=6 requests) per epoch. The control request communication needs to communicate 24 bits in 20ns, hence requiring a 2 Gbps link, as previously mentioned. While running up to I iterations to process the requests, the scheduler stores the wavelength-timeslot pair grant information in a second buffer of 1.2kB. Once the schedule is computed for an entire epoch, the wavelength-timeslot pair (25 bits) for each blade is communicated for every timeslot (20ns), again using the 2Gbps transceiver link. As shown in Fig. 5.6, the control plane of each sub-network is co-hosted within the source rack to keep the request-response handshake propagation delay to a known minimal constant ($\approx$ 30 ns). Regardless of where the destination rack is hosted, the control plane propagation is a constant, as discussed earlier.

5.1.3 Protocol Handshake: Control and Data

Fig. 5.4 shows the timing diagram for the handshake protocol that integrates PULSE's data and control plane networks. As shown by Fig. 5.3, the communication timeline groups timeslots to form *epochs*. The scheduler also computes, for an epoch, wavelength-timeslot grants for a specific epoch. Fig. 5.1 shows a source blade sending requests to the relevant scheduler (labelled as 1 in fig. 5.4) that is associated with the destinations of interest (D_{63}, D_2, D_{15}) with timeslots requested (2,1,3 respectively) an epoch in advance. The scheduler performs as many iterations as it can in one epoch to compute the wavelength ($\lambda_{64}, \lambda_8, \lambda_{27}$ in Fig. 5.1) and timeslots for each request (labelled as 2 in fig. 5.4). The granted resources (wavelength and timeslots) are sent back to the sources and transceivers (labelled as 3 in fig. 5.4). This is followed by the communication of data from the source to the

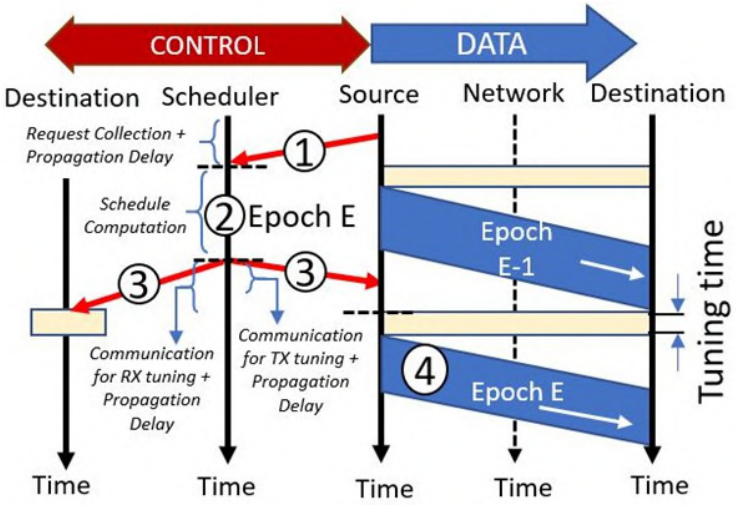


Figure 5.4: Control: Source-Scheduler-Source-Transceivers, Data: Source to Destination

destination using the relevant wavelength and timeslot (labelled as 4 in fig. 5.4).

5.1.4 Advantages of PULSE Network Architecture

PULSE is a single-hop network that inherently supports uni-, multi- and broadcast traffic with maximized net throughput, as purely the packet payload is communicated and the need for addressing is removed. PULSE can provide ultra-fast topology reconfigurations taking sub-μs timescales to create stable graphs. A wide range of circuits lasting O(10ns) to hours can be created. The PULSE network is a scalable network that has zero packet loss and no in-network queueing/switching with zeros switching power consumption (passive cores).

As such, the PULSE architecture does not require in-network a) routing/switching, b) buffering and network addressing. However, it requires ultra-fast a) tunable wavelength switching b) filtering, c) clock and data recovery [3], d) ps-timescale synchronization and e) scheduling. Research is being carried out on several aspect of PULSE while this work focuses on the scheduling aspects, based upon previously demonstrated physical layer demonstrations.

5.2 Ultra-fast transceivers

The work in [5, 25] proposed each node to be equipped with a tunable DS-DBR laser and a coherent receiver with an independently tunable DS-DBR local oscillator laser. This enables fast wavelength selectivity, O(10ns), at both the transmitter and receiver, and the high sensitivity of the coherent receiver also enables scalability in the data plane since it allows for a larger system loss budget and thus a higher port count star-coupler [5, 25]. Although a 1000-port star coupler network can be created, experiments have shown that only up to 89 wavelengths can be supported

by the laser hardware positioned on a 50 GHz ITU grid within the optical C-band to ensure minimal crosstalk between them ($W \ll N$). Extending the same grid to the L-band could allow the use of 160 wavelengths.

However, Fig. 5.5 shows the growth of this tuning time when using an extrapolated regression model with greater than 99.9% confidence interval; this predicts that a 160-wavelength system takes more than 1μs to tune all transceiver pairs. The large tuning time has a direct impact on switch latency and tail latencies, which degrades network performance, setting a limit on the wavelength channels allowed. This performance degradation has an impact on the scalability of the OCS network in terms of capacity, limiting $W \ll N$. Regardless of the number of wavelength channels used, the large tuning time would require a dedicated synchronized tuning time prior to every epoch, which lasted in O(1μs) to minimize overhead.

5.2.1 Transmitter options

Following from the previous argument, methods of approaching a larger number of wavelength channels (W) are explored, while making sure that optical transmitters do not impede switching times. With that aim, 3 WDM transmitter architectures that achieve sub-nanosecond switching are considered. Electro-optical amplifier switch based on chip-on-carrier SOA with tunable lasers were experimentally demonstrated to achieve switching times of up to 115 ps by [93]. Here, k such SOAs are employed at the transceiver, in order to enable reconfiguration at a faster rate of the O(10ns), connected to one of three laser source options that are proposed

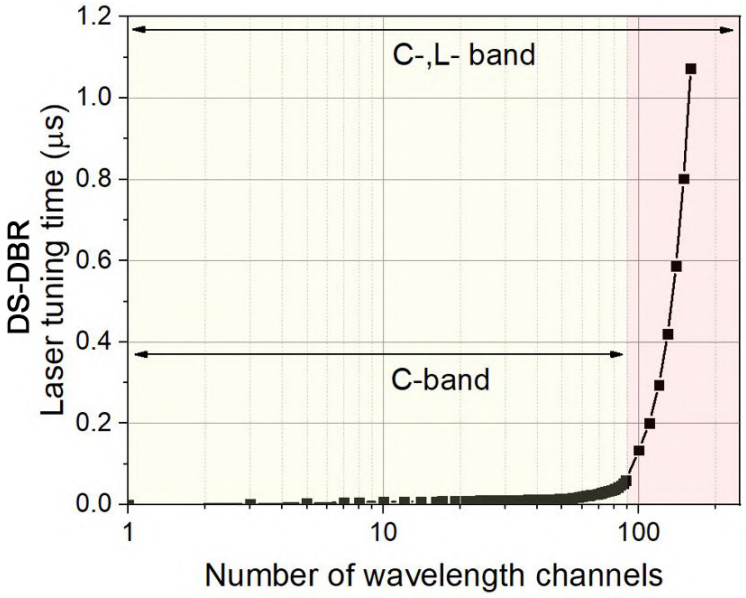


Figure 5.5: DS-DBR laser tuning time prediction versus the number of available wavelength channels with 99.9% regression on [5].

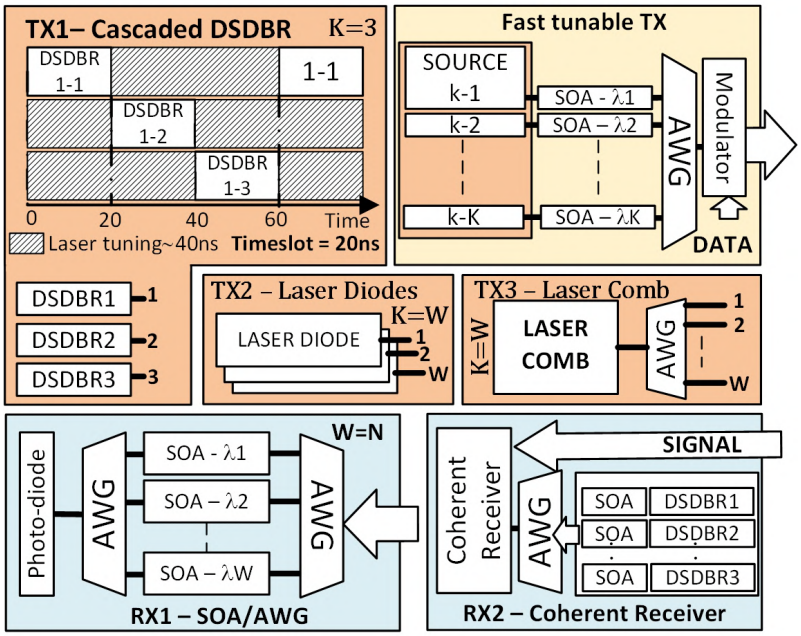


Figure 5.6: Transceiver options for PULSE: TX1-Cascaded DS-DBR, TX2-Laser Diodes, TX3-Laser Comb, RX1-SOA/AWG, RX2-Coherent Rx.

as shown in Fig. 5.6. In the transceiver proposal, the first option is to employ k tunable DS-DBR lasers described in the previous sub-section in a cascaded fashion, connected to k SOA gates, TX1 in Fig. 5.6. Prior experiments have shown that 90% of transitions between any transceiver pair of the 89 available wavelength channels complete tuning within 40 ns [5]. As shown by top-left of Fig. 5.6 (TX1), $k = 3$ cascaded DS-DBRs are required at the transmitter and the receiver to create 20 ns timeslots. The aim is to increase the network transceiver efficiency to 100% by increasing the number of channels to the number of blades in the sub-network ($W = N$). Hence, I investigate the network performance for (N=) 64 blade racks and (x=)16 racks, scaling to 1024 blades. The second transmitter option is the use of k(=W) VCSEL laser diodes [94], one for each wavelength, at the source connected to k SOA gates as shown in Fig 5.6. The third option is to use a laser comb that generates k(=W) wavelength sources connected k SOA gates [95].

5.2.2 Receiver options

Two options for the receivers are proposed, as shown at the bottom of Fig. 5.6. The first (RX1) contains an array of W SOAs surrounded by AWGRs, followed by a direct detection photodiode. The selection of the SOA allows the fast wavelength selection (O(100ps)) at the receiver. The disadvantage of such a receiver is the requirement of many SOA gates, which has an impact on overall power consumption. The second receiver (RX2) contains k(=3) DS-DBR tunable transmitters with SOA

gates, as at the transmitter, which serve as local oscillators (LOs) for a coherent receiver.

5.3 Ultra-fast Scheduler

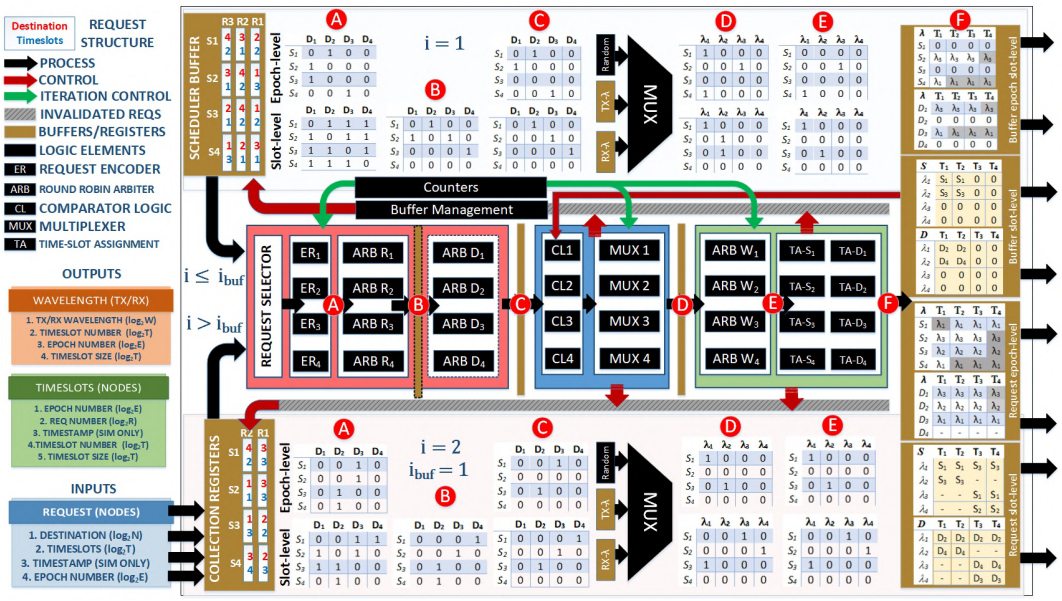


Figure 5.7: Epoch/Slot-level algorithm: 4-port central scheduler showcasing three hardware stages with a 4-port scheduling example showing two iterations, first handling requests from buffer (top), then from the blade (bottom) (when $i_{buf} = 1$).

This section describes two hardware implementable scheduling algorithms, which can use any of the fast tunable transceiver options shown in Fig. 5.6. To meet the strict timing requirements, a parallel design that considers R requests from each of the N blades is adopted. Although parallelism speeds up the scheduling process, it also introduces contention. Round-robin arbiters are used in the hardware design to ensure the fair selection of up to W requests, with unique source and destination ports, per clock cycle. The selected contention-free node-pairs are assigned wavelengths (WDM) and timeslots (TDM). The scheduler aims to perform I iterations within one epoch to maximize throughput. The number of iterations is limited by the epoch size ($E \in 120, 360, 600$ns) and clock speed of the hardware (2.3ns). Requiring four cycles to boot up, the maximum number of scheduler iterations is $I = \lfloor E/\text{clk} \rfloor - 4$. After I iterations, failed requests are buffered and are given a chance to retry in subsequent epochs. The major differences between the epoch-level and slot-level scheduler is in the strategy used to allocate resources. The buffer management technique (Algorithm 1), epoch-level (Algorithm 2) and slot-level (Algorithm 3) scheduling algorithms using pseudo-code in the Appendix section A.

5.3.1 Hardware Scheduler Design

The pseudo-code in Appendix section A shows two distinct algorithms for resource (wavelength and timeslot) allocation. Fig. 5.7 shows the epoch-level and slot-level scheduling for a 4-port scheduler for and iteration dealing with buffer ($i = 1$ shown by the top half) and dealing with requests from blade ($i = 2$ shown by the bottom half). This section describes the hardware or logical elements used to synthesise and implement the algorithms. The dominant element used in the realization of the scheduling algorithm is the round-robin arbiter shown in Fig. 5.7. The critical path lies in the carry chain of the arbiters, like in digital adders. Using optimal carry look-ahead generators can provide high-speed arbitration with low-critical paths.

5.3.1.1 Node Contention Resolution (NCR)

In epoch-level scheduling (Algorithm 2), each clock cycle deals with one request per blade (N requests) (line 7-10 and denoted by A in Fig. 5.7). Having no source contention (one request per source), N parallel N-port arbiters are used to resolve destination contention to form contention-free node pairs (line 11 and denoted by C in Fig. 5.7).

In the slot-level scheduler algorithm (Algorithm 3), the NCR block considers all requests from all sources (line 9-14 in pseudo-code and denoted by A in Fig. 5.7). Hence, this stage is composed of $2N$ parallel N-bit round robin arbiters that are used to arbitrate and select up to N contention free node-pairs. N arbiters are used to resolve source contention (line 15 and denoted by B in Fig. 5.7), while the other is used to resolve destination contention (line 16 and denoted by C in Fig. 5.7). The winning node-pairs are stored in registers and forwarded to the next stage.

5.3.1.2 Wavelength Decision (WD)

The second stage is composed of parallel multiplexers, one per node, that check previous assignments and select the wavelength based on the check. This stage works on a contention-free node-pair (C in Fig. 5.7) and uses information from registers to perform a parallel check on node-pairs. This is shown by the red line feeding back from registers to comparator logic in Fig. 5.7 and lines 12-27 in Algorithm 2 (repeated as WD in line 17 of Algorithm 3). Parallel contention resolved node-pairs (source transmitter and destination receiver) are compared against previously assigned resources (wavelength and timeslot assignments). If contradicting wavelengths have been assigned, the request is invalidated for the current epoch (shown by shaded gray lines going to request collect registers and red line going to buffer or blade). If either only the transmitter or only the receiver has been assigned a wavelength, then that same previously assigned wavelength is selected (shown by the MUX in the diagram), subject to time-slot availability. If there is no wavelength

assignment history for the transmitter and receiver and resources are available, a random wavelength is assigned, subject to time-slot availability. The selection of wavelength in this stage does not guarantee a winning grant, as there is the possibility of contention in the parallel resource allocation process. Hence, the selected wavelength assignments and the node-pairs are stored in registers and forwarded to the next stage.

5.3.1.3 Wavelength Contention Resolution (WCR)

The third stage reads the wavelengths requested by node-pairs (D in Fig. 5.7, line 28 in Algorithm 2 and line 18 of Algorithm 3). Using W parallel N-bit round robin arbiters, the contention between wavelength requests is resolved (E in Fig. 5.7). Up to W winning grants, with no wavelength or destination node contention, are generated in parallel by the arbiters. In epoch-level scheduling (Algorithm 2), the winning grants of the arbiters are granted as many time-slots as they have requested, subject to availability. If only fewer slots are available, then available slots are granted and the request is re-validated (shown by the red line to the buffer in Fig. 5.7), updated with the new slot size and buffered for processing in subsequent epochs. In slot-level scheduling, there are two phases: coarse allocation and fine allocation. In coarse allocation, parallel requests are allocated a multiple slots as requested, provided availability. In fine allocation, one slot is allocated per request. All requests with winning grants are marked to avoid repeating requests in future iterations (also shown by the red line to the buffer). The requests that win the initial iterations (based on round-robin arbitration) are given priority and use coarse allocation, while subsequent requests use fine allocation to maximize matching. Due to pipelining, the register sequencer stage does not have knowledge of the most recent wavelength update and can still request contradicting wavelengths. In such cases, the requests are rejected and invalidated for requesting in the current star (shown by red line going to Invalid Request). The wavelength-timeslot configuration is updated, registered and sent to the transmitters and receivers of each node.

5.3.2 Iteration/Buffer Management

5.3.2.1 1-plane

Pseudocode Algorithm 1 in Appendix A shows how the scheduler iterations are managed to cater for the requests from both the buffer and blade every epoch. The size of request residing in the buffer (buffer size) is constantly kept in account. The iteration ratio (buffer:blade) is controlled by i_{buf} as shown in Fig. 5.7. Since up to W grants are generated every iteration, the total buffer size is divided by W and multiplied by a buffer coefficient (R_p in pseudocode, usually between 1.6-2.5),

which ensures minimal buffer accumulation. In epoch-level scheduling, which is a pointer based scheduling, the pointer choosing the requests in the buffer hold until a minimum percentage of requests are granted or a threshold value is met. After this, the pointer shifts to the next set of requests. In slot-level scheduling, there is a collective look at all demands and it does not require this threshold value or buffer pointer monitoring.

5.3.2.2 2-plane

While the modules discussed above form the core processors in PULSE's network schedule processing unit (NsPU), a smart management of iterations is required to maximize throughput. PULSE NsPU requires 4 clock cycles to boot up every epoch. This corresponds to 9.2 ns in a N=64-port sub-network. Apart from booting up, the scheduler now has to schedule requests across p sub-networks. To increase the QoS, PULSE NsPU employs coarse (multiple slots granted per iteration per node) and fine timeslot allocation (one slot per iteration per node) in coordination with priority iterations for buffered requests, i.e. failed requests from previous epochs, in order to minimise latency.

In order to avoid convergence requirements, the number of schedulers is not increased with each of the p splits in the 2-plane architecture. Hence, for all P sub-networks, the scheduler must use the same number of iterations (I) and continue to support a high throughput and low latency. Hence, there is a requirement to manage iterations efficiently; requests for each sub-star are dealt in batches and hence, divide iterations across P networks: I/P iterations per sub-star. In order to maintain fairness, a pseudo-random generator shuffles sub-star priority in a round-robin fashion.

5.3.3 Scalability Review

The scalability of PULSE NsPU modules when synthesized on 45 nm CMOS ASIC is shown on Fig. 5.8. The NsPU primarily uses the same hardware modules proposed previously in the parallel hardware scheduler. As the NsPU NCR stage is pipelined into 2-arbiter stages to consider all requests in every iteration, the critical path for the NCR remains the same. As the number of wavelength channels has been increased to match the number of ports to maximize network efficiency, $W \times N$-port arbiters have a longer critical path compared to previous solution matching the clock speed of the NCR. As highlighted in Fig. 5.8, a 64-, 128- and 256-port NsPUs are shown to have clock speeds of 435, 345 and 256 MHz respectively when synthesized on 45 nm CMOS ASIC.

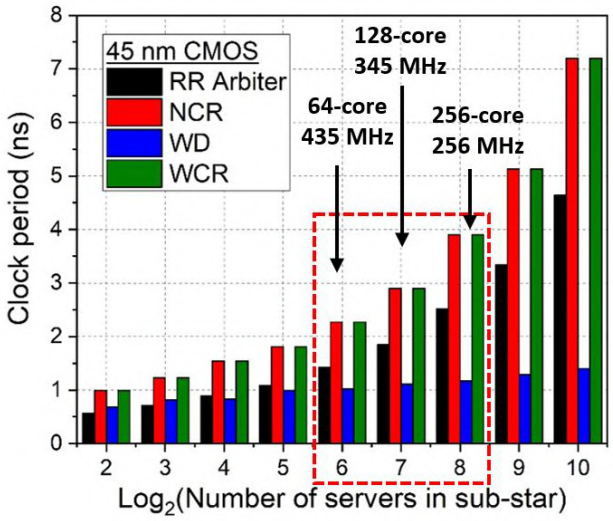


Figure 5.8: Scalability of NsPU modules on 45 nm CMOS ASIC
NCR: Node Contention Resolution, WD: Wavelength Decision, WCR: Wavelength Contention Resolution

5.4 Requirements

5.4.1 Synchronization

The transceiver-based optical switch requires nanosecond resolution time-slot synchronization (each time-slot is 20 ns). Although practical realisation of this synchronisation is beyond the scope of this thesis, it is still a crucial requirement for error-free communication. Prior research has shown optical fiber clock distribution to 1000-ports with jitter less than 12 ps, using mode-locked semiconductor lasers [96]. Reliant on the use of a dedicated 1 Gbps synchronization plane, the White Rabbit project can achieve a clock accuracy better than 1ns and precision better than 50 ps spanning distances over 10km [97].

5.4.2 Clock Data Recovery

Recent practical demonstrations by [3] have shown clock data recovery (CDR) locking to be achieved in < 625 ps using *phase caching*. The phase information is required to be updated only once every minute, which makes the CDR settling time a negligible overhead and hence, they are not considered in latency and throughput measurements. This removes preamble needs, although phase has to be updated once per several million epochs. This technique has been demonstrated for a 25 Gbps OOK modulation in real-time and is practical for a 100 Gbps transceiver.

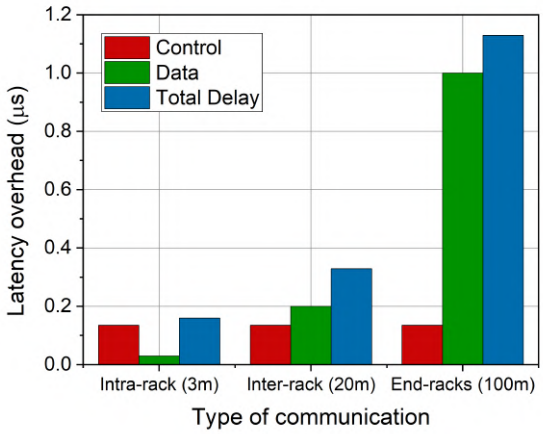


Figure 5.9: Control, Data Latency overhead: propagation delay

5.5 Network Characterization

5.5.1 Propagation delay: Latency overhead

The modulation, serialization, transceiver latency and the propagation latency all contribute to the total latency overhead. This sub-section aims to highlight the dominant latency overheads caused by propagation delay within the network. Fig. 5.9(a) shows the latency overhead if the communication in the PULSE architecture is intra-rack, inter-rack or end-racks, assuming lengths (L) of 3m, 20m and 100m respectively. Each of these links in the data plane corresponds to fiber runs of L meters to the coupler and L meters to the destination. The architecture, as shown in Fig. 5.1, has the control plane scheduler co-located with the source blade racks within a 3m reach, regardless of where the destination rack resides. The co-location of the scheduler means that the latency overhead of the control pane is a known constant (shown in Fig. 5.9) and, since the data plane overhead dominates, the configuration of the network is done long before the data arrives. As shown in Fig. 5.9, a total latency overhead of 0.15μs, 0.33μs and 1.12μs are incurred when communicating intra-rack, inter-rack and end-rack.

5.5.2 Scalability: Capacity and Port count

5.5.2.1 1-plane

By increasing the number of wavelengths channels available to the number of transceivers in the network ($W = N$), the network efficiency of each sub-star can be increased. Hence, exploiting WDM, multiple routes (or pipes) can be created and the scheduler has a higher choice of route selection. The parallel SDM networks created by the x^2 sub stars enable the network to have increased capacity and

allows re-usability of wavelength channels in parallel sub-stars. The new scalability unlocked can either increase $x\times$ more blades or increase $x\times$ more capacity for N blades. The control plane network complexity is also reduced by creating multiple local and distributed schedulers, instead of a global centralized scheduling scenario previously proposed.

Table 5.1 shows how the network, requests, racks, cables, channels and capacity scale while increasing the number of transceivers and racks ($x \in$ 4,8, 16, 32 and 64) at 64 blades per rack ($N = 64$).

Network Parameters	**Transceivers per blade (x)**				
	4	**8**	**16**	**32**	**64**
Total blades	256	512	1024	2048	4096
Req/blade/epoch (R)	24	48	96	192	384
Racks (x)	4	8	16	32	64
Sub-stars (x^2)	16	64	256	1024	4096
Cables ($N \times x^2 \times 4$)	4096	16384	65536	0.26M	1.04M
Channels (Wx^2)	1024	4096	16384	65536	0.26M
Max capacity (Tbps)	100	400	1598	6394	25575

Table 5.1: PULSE: Scalability, Capacity, Complexity at $N = 64$

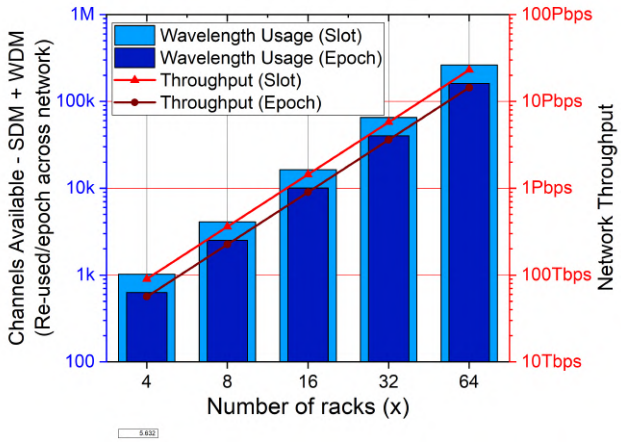


Figure 5.10: PULSE: 1-plane scalability of capacity and wavelength channel re-use (SDM)

The spatial division multiplexing (SDM) created by the parallel and distributed star-couplers enables the re-use of wavelengths. The complete independent nature of individual sub-network means that local schedulers have no dependency on the traffic or resource usage faced by other stars. The synchronization problem is also reduced to a local sub-network and not required at a global level. Each 64-port sub-star uses 64 wavelengths and the SDM enables 4096 parallel usage of wavelengths

allowing a total of 0.26M channels and a network with an enhanced capacity of ~ 25.6 Pbps, accounting tuning overhead.

As shown in Fig. 5.10, as x increases the number of racks and the capacity increases, exploiting WDM. The number of wavelength channels re-used exceeds 200K, across 4096 sub-stars and hence, a network with high capacity is made.

5.5.2.2 2-plane

Table 5.2 shows the scalability of PULSE with the 2-plane topology when using (x=)16 and 64 transceivers per blade or racks per cluster and introducing a split at the transmitter and the receiver.

Parameters	**Eq.**	**(Split, p) (x=16)**			**x=64**		
		2	**4**	**8**	**2**	**4**	**8**
Blades (#)	Npx	2048	4096	8192	8192	16384	32768
TX/RX (#)	Npx^2	32768	65536	0.13M	0.5M	1M	2.1M
Sub-stars (#)	x^2p^2	1024	4096	16384	16384	65536	0.26M
Channels (#)	Wx^2	0.26M	1M	4.2M	4.2M	16.8M	67.1M
Capacity, Pbps	$BNpx^2$	3.2	6.4	12.8	12.8	25.6	51.2

Table 5.2: PULSE 2-plane scalability (N=64, W=N, B = lr*e, where linerate (lr) = 100 Gbps, efficiency (e) = 0.976)

High blade counts of 10,000s can be supported by extending the topology by introducing a split after the transmitter and coupler before the splitter. While the re-usable channels and the capacity remain the same as Fig. 5.10, more number of racks and blades are supported (increased by P). The cost of scaling with such network architectures is the requirement of high number of NsPU (x^2). This corresponds to 1 NsPU per one blade in the server in the 1-plane architecture and 1 NsPU per P blades in the 2-plane architecture. In order to reduce the number of NsPUs per blade, large radix switches like N=256, 512 and 1024 port must be supported. However, at such high numbers for efficient network utilization, $N = W$ wavelengths are required, impacting the transceiver complexity.

5.5.2.3 High Capacity Networks

As shown in table 5.1, each node uses up to 6.4 Tbps. PULSE is a network solution, where each node can be a high-performance computational resource in a heterogeneous cloud DCNs - CPU, GPU, TPU, HBM (>1 Tbps) or ToRs. Resources like HBMs and GPUs [98] require more than 1 Tbps bandwidth. Large multi-core chips are also being explored to support fast AI calculations [99]. There has been considerable growth in the performance of GPUs, nearly 1.5 times a year [100] and reaching 1.5 Tbps by 2020 [98]. Although network and computational

processing are capable of handling high throughput (100 Tbps), NIC or data ingest capacity constraints (100 Gbps) create bottlenecks, leading to inefficient systems that constraint applications to operate locally and degrade the overall application performance. Hence, the FastNICs program initiated by DARPA [101] aims to boost network NIC and stack capacity by 100 times to accelerate distributed application performance and close the gap between processing and network capability. Hence, it is expected that in the future end-nodes will support high bandwidths (6.4 Tbps). Today's expensive network switches are over-designed and under-utilized [26]. PULSE intends to maximize bandwidth utilization even if high capacities are generated at the nodes.

5.5.2.4 Multi-transceiver/blade

Integration of large channel bandwidth-dense transceivers as integrated SiP mid-board optics (MBOs) has been shown in [102], proving the feasibility of supporting densities of 64 Gbps/mm^2 (as of 2014). In 2018, an ASIC switch with in-package optical transceiver ports was demonstrated [103]. A similar co-packaging with FPGA was reported in [104] and demonstrated at [105]. Dense SiP integration of transceivers can enable the accommodation of 64 transceivers on a PULSE node.

5.5.3 Power consumption

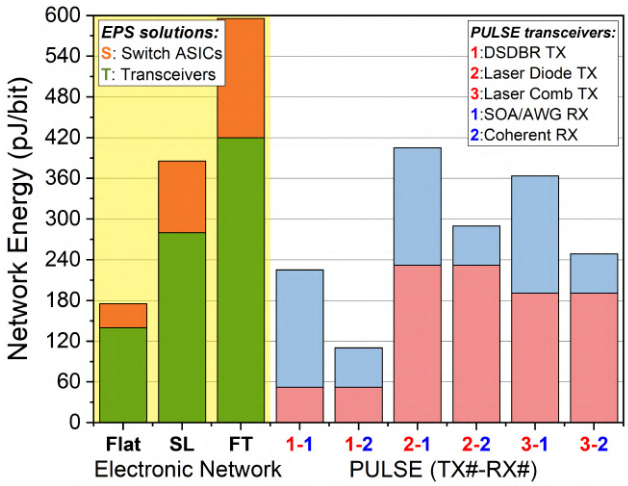


Figure 5.11: Network energy consumption comparison with equivalent electronic networks

PULSE is a small-scale high bandwidth data center network that scales to support up to 4096 blades. Novel architectures that can scale to more blades are being explored. In Fig. 5.11, the network energy consumption of three electronic architectures is shown and compared with PULSE. The details of the components used, and references are found in table IV. The Flat architecture simply replaces sub-stars with electronic packet switches, using the switches and 2x transceivers (dou-

Device	TX1	TX2	TX3	RX1	RX2	Power per Unit (mW)
SOA [106]	3	64	64	-	-	260
Comb [95]	0	0	1	-	-	1000
LD [94]	0	64	0	-	-	80
AWG	1	1	2	-	-	-
MOD [107]	1	1	1	-	-	1460
DS-DBR [108]	3	-	-	-	3	1000
RX SOA [106]	-	-	-	64	3	260
RX AWG	-	-	-	2	2	-
CO-RX [109]	-	-	-	0	1	2000
PD [110]	-	-	-	1	0	630

Table 5.3: Component count, power assumptions per TX/RX

ble hop). Although state-of-the-art electronic transceivers consume 7 W per 100 GBE port [111] while switch ASICs assume 225 W per 6.4 Tpbs [112], their overall contribution to the network is 17.5W/path in the Flat network architecture. SL architecture stands for a Spine-Leaf (2-tier) network, which has 3x more switches per path (multi-level) and 4x more transceivers per path (multi-distance). The FT architecture stands for a Fat Tree (3-Tier) network with 5x more switches per path (multi-level) and 6x more transceivers per path (multi-distance). Figure 5.11 showcases the power consumption of transceiver proposals for PULSE, as shown Fig. 5.6. The power values assumed to estimate the network energy and their references are shown in table III. Cascaded fast tunable DS-DBR lasers with coherent receiver technology is shown to achieve the lowest power of 110 pJ/bit (11 W/port), which 65 pJ/bit lower than the equivalent Flat electronic network. The power minimization in PULSE is mainly due to two main reasons. The first is the reduction it offers to the number of transceivers being a single-hop network. The second is the removal of active switch elements in the path and their contribution to the overall network energy consumption.

5.5.4 Cost estimation and comparison

In order to fairly compare PULSE with state-of-the-art electronic networks, the cost of deployments with equivalent bandwidth performance (6.4 Tbps), blades and full bisection bandwidth is evaluated. Cost estimates are normalized to the capacity that each component supports ($/Gbps); for example, the cost of a leaf/spine 100 GBE transceiver cost is $3/Gbps [113]. The end-to-end cost in a network is determined by the network diameter and traffic locality on EPS multi-tier systems. A full-bisection bandwidth is assumed and hence, the cost is determined by the

Component	$/Gbps	#/path			
		Flat	SL	FT	PULSE
100GE Transceiver (3m)[113]	0.1	-	2	2	-
100GE Transceiver (20m)[113]	3	2	2	4	-
Arista 7170 (6.4 Tbps) (Tier 1) [114]	10	1	2	2	-
Arista 7328X (12.8 Tbps) (Tier 2) [115]	10	-	1	2	-
Arista 7516R (25.6 Tbps Tier 3) [116]	12	-	-	1	-
100GE Transceiver (Co-Rx low)	6	-	-	-	1
100GE Transceiver (Co-Rx med)	9	-	-	-	
100GE Transceiver (Co-Rx high [117])	12	-	-	-	
Star-Coupler [118]	0.04	-	-	-	1
Total ($/Gbps)		**16.2**	**36.2**	**64.2**	**6.04-12.04**

Table 5.4: Cost estimate of PULSE network compared with electronic DCNs (Flat: PULSE equivalent with EPS, SL: Spine-Leaf, FT: Fat-tree) as of 2019

longest path/diameter. In contrast with PULSE, the spine-leaf (2-tier) and fat-tree (3-tier) have multiple layers of switches interconnected, where each device adds to the additional cost as shown by table 5.4. An electronic switch uses two ports (downlink and uplink) to form an interconnection and hence, the normalized cost is divided by a factor of 2. The major cost contributors in electronic networks are the switching devices; both the number of levels and the level number influence the cost heavily. From this analysis, the cost of the PULSE equivalent Flat network costs $16.2/Gbps, while spine-leaf topology costs about $36.2/Gbps and a fat-tree topology costs $64.2/Gbps. The table shows the dependency of cost on the locality of traffic; the total cost depends on the ratio of traffic that is exchanged between the tiers. In PULSE, a flat transceiver-based architecture, the normalized cost per path is determined by the transceiver architecture while the transport layer cost is kept to a minimum of 0.04$/Gbps, assuming a 64-port coupler cost of $240 based on double the cost of a 64-port splitter in [118]. The price of the 100G coherent receiver transceiver is dependent on the complexity of the receiver architecture and DSP required. Simple to complex coherent receivers are assumed to cost 2, 3 and 4 times the 100 GBE direct detect receiver RUC (relative unit cost) based on the report by ASTRON [117] (as coherent transceivers offer the best energy efficiency

- Fig. 5.11). Using [117], the RUC for adapting the PULSE transceiver architecture is estimated to have a worst case RUC is 2.97 (3x) compared to 100 GBE transceivers. This shows while employing high complexity coherent transceivers, the cost of PULSE is $12.04/Gbps, achieving 1.25 $\times$, 3-7$\times$ and 6-11$\times$ cost efficient compared to state-of-the-art electronic Flat, SL and FT networks respectively. The analysis in [113] shows that price of transceivers is significantly dropping every year, which would benefit a transceiver-switched architecture like PULSE. Moreover, the report in [119] predicts an annual cost reduction of coherent solutions by 15% and that beyond 16 WDM channels, they will be more cost effective. The report in [69] predicts that 800G coherent modules employed in data center networks could approach the cost of $1/Gbps by 2024.

The study shows that, in electronic networks, transceivers consume higher energy per capacity than network switches. On the other hand, network switches impact cost per capacity more than transceivers. Hence, this conventional combination affects both power and cost. While PULSE reduces latency by 3 orders of magnitude, the novel optical architecture also delivers lower cost and consumes lower power by reducing the number of switches and transceivers required.

5.6 Summary

Following a critical review of performance limitation in chapter 4, this chapter attempts to remove some of the network performance limitations and develops an ultra-fast scalable OCS network architecture called PULSE. PULSE is a SDM/WDM/TDM optical architecture that employs parallel distributed hardware scheduling to support more blades. PULSE scales in one/two dimensions when employing the proposed 1-/2-plane architecture, where the first dimension is dependent on # transceivers and the second dimension introduces a splitting/coupling after/before the transmitter/receiver. The 2-plane architecture uses one scheduler and one transceiver per p clusters. The modularity introduced in PULSE eases the scalability requirements on each sub-network. Sub-microsecond speed transceivers (SOA-aided TX/RX) that can achieve tuning between each timeslot in an epoch is introduced. A new scheduler hardware architecture that employs an additional set of arbiters and management units to schedule resources at a slot level are shown to achieve similar scalability to the parallel hardware scheduler. The modifications required to support and schedule for a 2-plane network is also shown. Mentioning the CDR and synchronization requirements, the network is characterized in terms of latency overhead, scalability, power consumption and cost. A latency overhead of 0.15-1.12 μs depending upon location of destination rack is indicated. The 1-plane PULSE network is shown to scale to up to 4096 blades, while the two plane

network is shown to scale to 32,768 blades. A power consumption per path of 110 pJ/bit (115) is shown for a 1-plane network (2-plane) compared to an equivalent 180 pJ/bit electronic network for the proposed transceiver architecture. The worst case normalized costis shown to be $12.04 per Gbit per path for PULSE compared to $16.2 per Gbit per path of an equivalent electronic flat network.

Chapter 6

PULSE performance evaluation

In this chapter, I evaluate the performance of the slot-level scheduling algorithm and compare it with epoch-level allocation. Once the performance boost is showcased, a scalability and latency reduction is analysed (labelled 'Larger racks, Smaller epochs'). Finally, the scheduling performance of a 2-plane architecture without employing additional scheduler units (and transceivers) is reviewed. Generating demand traffic with diverse distributions, I study the effect of varying epoch sizes and request loads on throughput, latency, tail latency, transmitter/scheduler buffer size, wavelength usage and energy consumption. Firstly, software models equivalent in functionality to the hardware algorithms were modelled in MATLAB. A request traffic generator is used to feed the scheduling algorithm models with demands and the evaluation of performance is detailed in this chapter. For more details on the performance evaluation of PULSE and it's NsPU, the reader can refer to the papers published in PSC 2018 [91], ECOC 2019 or the papers submitted to OFC 2020 (Invited talk) and the JLT Special Issue on Optical Interconnects 2020 (under review).

6.1 Traffic Pattern

At every epoch, the requests that are generated by the source are sent to the relevant local scheduler. Each request packet consists of: the destination blade, the number of time-slots required and the origin epoch number. Firstly, to model the demand matrix generation, a uniform random distribution was used to select the destination node with a probability of $1/N$. Secondly, the average size of each request corresponds to the number of slots available in an epoch divided by the requests per blade per epoch ($S_{avg} = T/R$). Up to R requests are generated by each source per epoch and a Poisson distribution is used to model the inter-arrival rate of requests. All the requests that arrive within the start of epoch are computed, else they are buffered for the next epoch. A uniform distribution of time-slot sizes of requests

is used to create the slot traffic distribution (TD) with an average slot size of S_{avg}. TD1 corresponds to a single size request, where all requests are allowed to request a time-slot of one specific size or none at all. In TD2, a total of three size values are allowed; in TD3, a total of five size values are allowed. To get a grasp of this double interpolation of uniform random distributions, Fig. 6.1 shows the number of slots requested by source-destination pair for a unique TD and R. Focusing one sub-plot, the y-axis shows the source blade number of the source rack and the x-axis shows the destination blade number of the destination rack. The color of the heat-map indicates how many slots are requested over a span of 2000 epochs. The generated traffic is for a 360ns epoch at 100% input load.

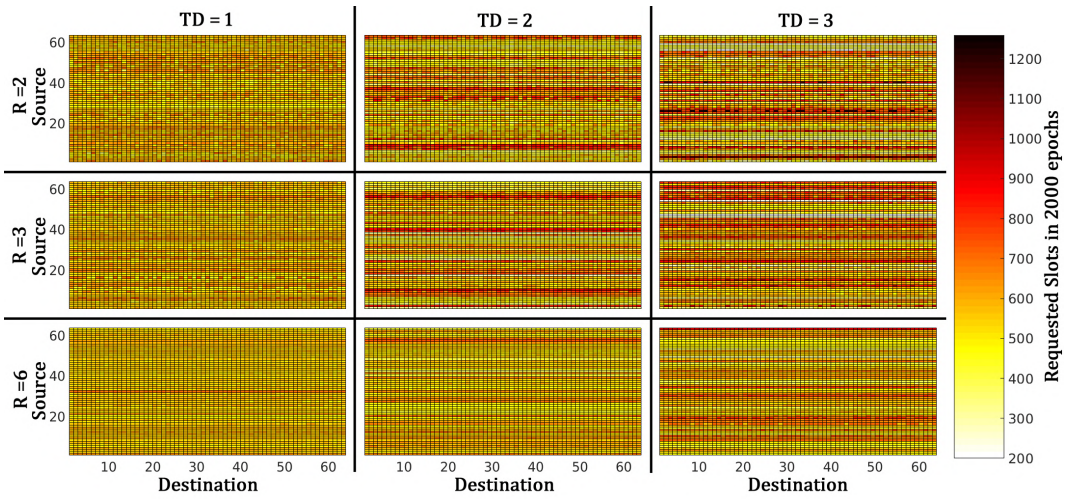


Figure 6.1: Active blades: source-destination Demand size for 2000 epochs for 360 ns epoch at 100% input load.

As shown in Fig. 6.1, the uniform random request (on both the slot and destination) traffic generation has created a very non-uniform pattern. At $R = 6$ and $TD = 1$, there is the least amount of variation in slot size between source-destination pairs. This is because as the number of request increases, the average timeslot requested is low and all requests are of the same size (TD = 1). As R reduces to 2, the size of each request increases creating a higher range of variance. As TD changes from 1 to 3, it can be seen that certain source blades become hot in the network, which demand more resources (wavelengths-timeslots) than others. Such demands are emphatic as high TD value with low R is approached. This is the type of traffic used to evaluate the performance of the scheduling algorithms.

6.2 Throughput

6.2.1 Epoch vs Slot level Tuning/Scheduling

The matching performance and throughput of a $N = 64$ blade sub-network was evaluated under varying traffic patterns and epoch sizes. The throughput of both the scheduling algorithms (slot-level and epoch-level) were evaluated for increasing input load. In Fig. 6.2, the effect of changing TD (column) and epoch size (row) on the throughput of both the slot-level and epoch-level scheduling algorithms is shown. Within each sub-plot, the effect of varying R is shown. The values shown take into account the relevant tuning times: 500ps for 20ns timeslot for the slot-level algorithm and 500ps for the entire epoch length for the epoch-level algorithm.

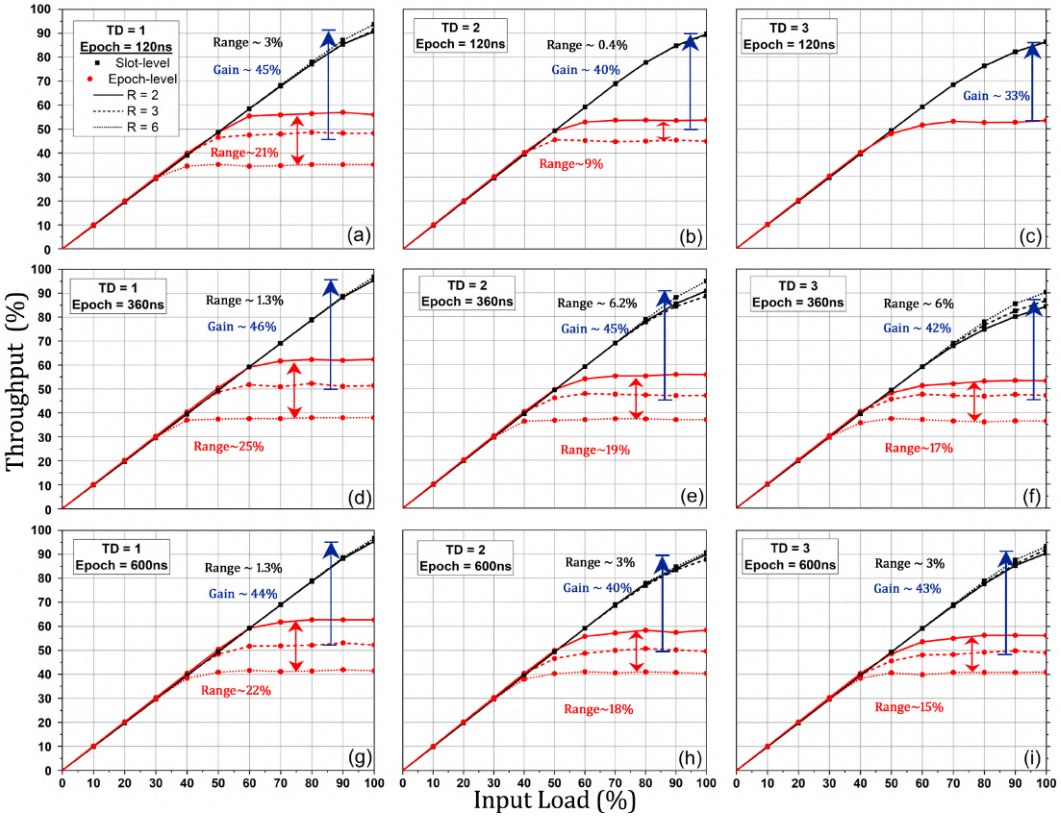


Figure 6.2: Scheduler throughput vs Input load for varying values of R, epoch sizes, TD.

The epoch-level scheduling algorithm in all sub-plots of Fig. 6.2 reaches a saturation point at approximate input loads of 60, 50 and 40% for 2, 3 and 6 requests per blade. In contrast, the slot-level achieves maximum throughput at 100% input load. The saturation point for the slot-level algorithm is between 85-95% while the epoch-levle saturates between 35-62%. As evident, the slot-level scheduling algorithm offers an average gain in the matching performance by 32-48%, compared to epoch-level scheduling for all values of TD and epoch size.

In the epoch-level scheduling algorithm, each wavelength allocation locks the transmitter and receiver pair to that wavelength for the entire epoch. This increases blocking probability, as future iterations have to work around this *locking*. In contrast, the slot-level allocation has the flexibility of changing the wavelength every timeslot. This flexibility is permitted by the architecture of PULSE's transceiver and network. The variance in throughput in the epoch-level scheduling between $R = 2$ and $R = 6$ also ranges from 18-30%. However, this variation is contained in the slot-level scheduling to less than 5%, showing the algorithm to be more tolerant to variation in request volume. As TD increases, the throughput in both the slot-level and epoch-level algorithm decreases.

6.2.2 Larger racks, smaller epochs

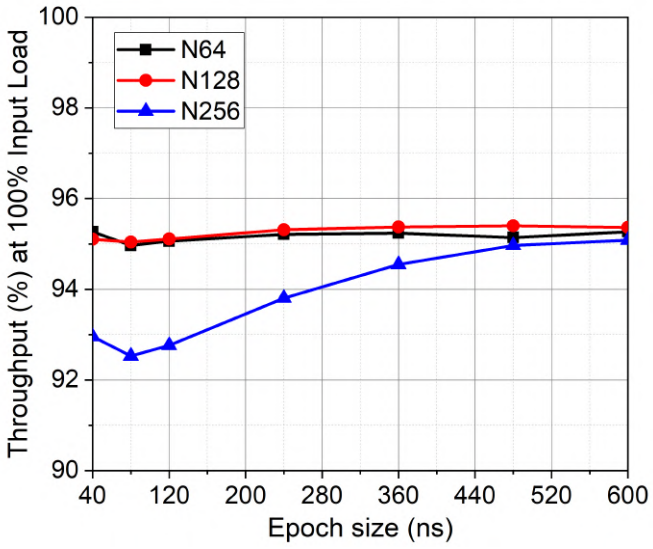


Figure 6.3: Scaling PULSE racks and reaching shorter epochs (throughput vs epoch size)

As the benefits of tuning and scheduling at the slot level is evident, the following sub-sections focus on optimising the scheduling to support larger racks, smaller epochs and the 2-plane architecture while maximising throughput and minimising latency. As can be observed from Fig. 6.2, the resulted traffic variations from the variation between R and S are also reduced (the missing R=3,6 in epoch 120 ns). In addition to this, the latency resulting from an OCS system can also potentially be reduced by reducing the epoch size. Larger flows can also be handled by using wavelength reservation schemes, wherein a set of source and destination nodes are locked to a wavelength for several epochs. Hence, the scalability of N=64,128 and 256 for lower epoch sizes were explored. The generated demand traffic sends up to 2 requests/server per epoch ($R = 2$) with uniform random destination ($P(1/N)$) and

slot demand ($P(R/T)$). A poisson distribution with a mean inter-packet arrival time of T/R is used.

Fig. 6.3 showcases the throughput achieved by the individual distributed hardware schedulers at 100% input load. The PULSE network achieves a sustained 95% throughput for 64 and 128 blade racks, regardless of epoch size. For a 256 blade rack (or sub-network), the throughput is 92.5% for small epoch sizes (40 and 80ns) and it gradually increases to 95% for a 600ns epoch. The PULSE scheduling algorithm achieves sustained throughput of >92%, taking into account a 500ps tuning overhead for every timeslot [93].

6.2.3 2-plane

Here, the effect of introducing a clustered architecture on throughput for N =64 blade rack is shown. As P increases, the schedule has to collect demand across p sub-stars and use the low number of iterations (due to low epoch size) and still continue to produce a high throughput that achieves optimal matching. A uniform random demand from sub-networks ($P(1/P)$) is assumed in addition to the traffic metrics discussed above. P=1 represents the comparison scenario, where a 1-plane architecture is used.

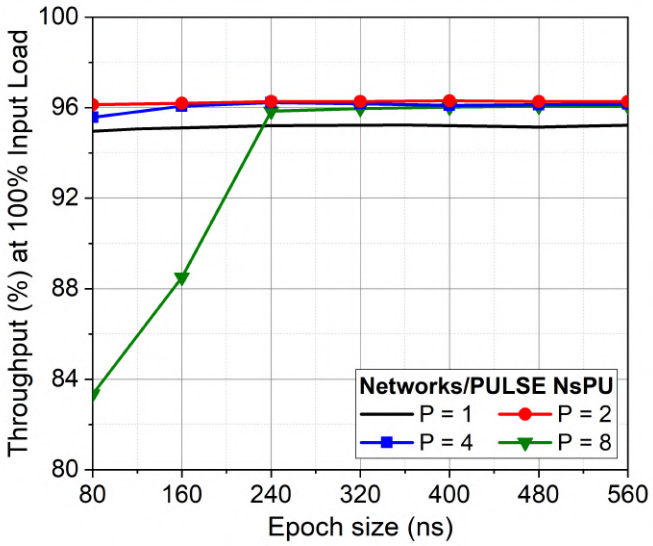


Figure 6.4: Scaling PULSE with 2-plane architecture (N=64, throughput vs epoch size)

Figure 6.4 shows the scheduler performance penalty experienced when the same NsPU is used to schedule wavelengths and timeslot resources for P =2, 4 and 8 sub-networks (tuning overhead included). At P=8 and 80 ns epoch, only 3 iterations are available per sub-star per epoch and hence, a 12% throughput penalty is experienced, reducing the matching performance to 83% as expected. However, an

epoch size of 240 ns is a better operating point for P=8 with above 95% throughput.

6.3 Wavelength usage

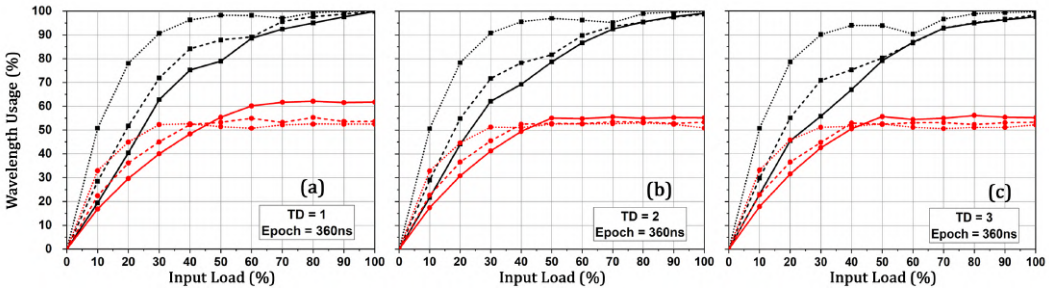


Figure 6.5: Wavelength usage vs Input load for varying R, traffic distributions: (a) TD1, (b) TD2, (c) TD3 in a 360ns epoch.

Fig. 6.5 shows the average percentage of wavelength channels (resources) used to achieve the matching for epoch-level and slot-level scheduling algorithms for increasing input loads in a 360ns epoch. For all values of TD, the maximum wavelength usage in the epoch-level scheduling algorithm is 49-62%. In contrast, the slot-level scheduling algorithm achieves a maximum wavelength usage of 97-100%. In both scheduling algorithms, the input load at which the saturation point is reached in the epoch-level algorithm matches the input load at which usage of resources also saturates. At 100% input load and low requests/blade ($R = 2$), that is low congestion, the saturation point is at 50-60% load. At the best performance, the epoch-level algorithm is able to use 32-38 wavelength channels every epoch out of 64. However, at 100% input load, the slot-level scheduling algorithm saturates very close to 100%, showcasing a network that utilizes up to 64 wavelength channels.

6.4 Average Latency and Transmit Buffer

6.4.1 Increasing input Load

To measure the latency, each request packet was marked with a time-stamp when generated. The scheduler has a buffer that stores failed requests in the current epoch to retry the same requests in future epochs. Once successfully and completely (all requested slots) granted, another time-stamp is stamped to mark when the request will translate into communication in the data plane and reach the destination. The difference in the time-stamps is taken for each packet to get the scheduling latency distribution.

Fig. 6.6 shows the end-to-end overall average latency taken for packets over 2000 epochs when the network is scheduled using epoch-level and slot-level algorithms for increasing input load, while varying values of requests/blade R (within

plot), epoch sizes (row) and traffic distributions TD (column). Each sub-plot also shows the load where maximum throughput is achieved (saturation point).

The propagation delay and tuning overhead are not considered in these measurements; they are discussed at the end of this section. The high throughput achieved by the slot-level algorithm means that many requests are granted without buffering. Hence, the slot-level scheduling algorithm is expected to have a low latency compared to the epoch-level scheduling.

The epoch size and requests/blade have higher impact on the latency compared to TD. As expected, Fig. 6.6(a-c) shows that, at 120ns epochs, slot-level scheduling has an average latency of 500-600ns at around 90-94% saturation load, dependent on R, while the epoch-level scheduling algorithm consumes 1-7μs at 35-55% saturation load. In the epoch-level scheduling algorithm, higher requests/blade (R) have a higher latency, whilst in the slot-level algorithm this variation is quite low. At 100% load for a 600 ns epoch (worst-case), the significant impact of epoch size on the latency of the scheduling algorithm is clear; the epoch-level scheduling algorithm incurs an average latency as high as 200μs, while the slot-level scheduling algorithm incurs an average latency of 4μs. Hence, the smaller the epoch size, the lower the latency. At 100% input load, the latency reduction offered by the slot-

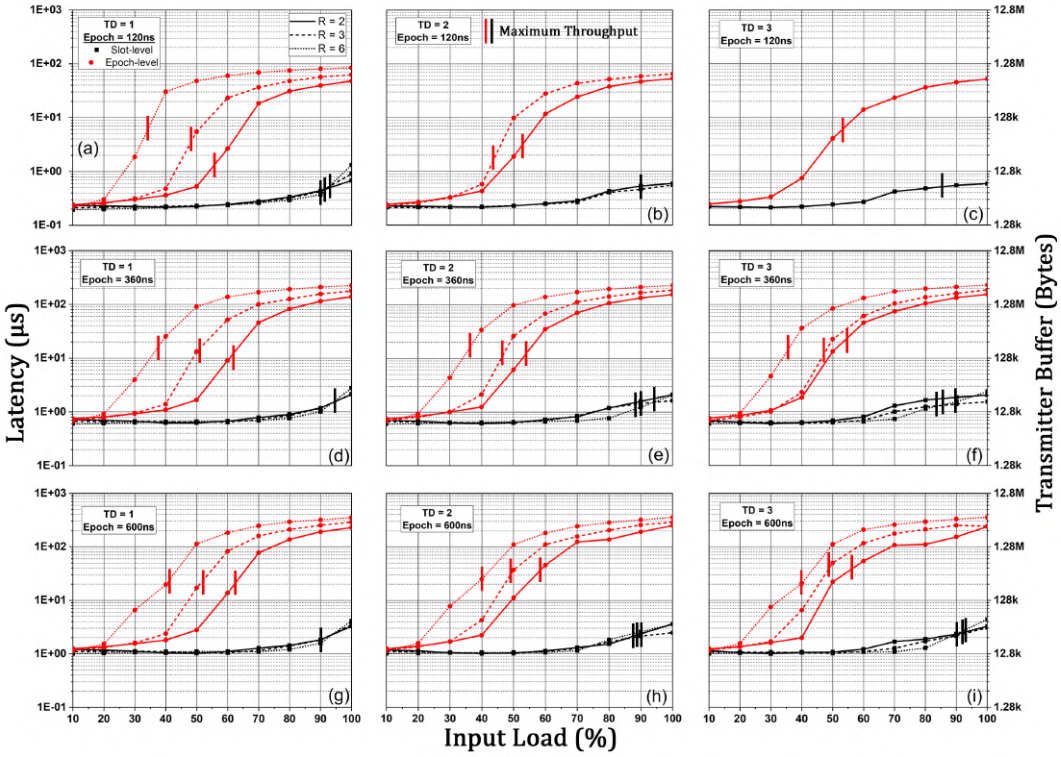


Figure 6.6: Scheduler average latency vs Input load for varying values of R, epoch sizes, TD.

level scheduling compared to epoch-level scheduling is 40-80 times, when using a 120ns epoch.

Latency has a direct impact on the average transmitter buffer size that is required at the transmitter, as shown by the second y-axis on Fig. 6.6. While requests are buffered in the scheduler, the data packets are buffered at the source where they await the grant from the scheduler. An average packet size of 250 bytes is assumed for every time-slot (20ns) delay. Just as latency is reduced by an order of magnitude by the slot-level algorithm relative to the epoch-level scheduling algorithm, the buffer size is also reduced by an order of magnitude. Assuming a 100% input load, the average buffer size required for a 120ns slot-level scheduling algorithm is 15.36kB, relative to the 1.09MB buffer size demanded by the epoch-level scheduling algorithm.

6.4.2 At maximum operable loads

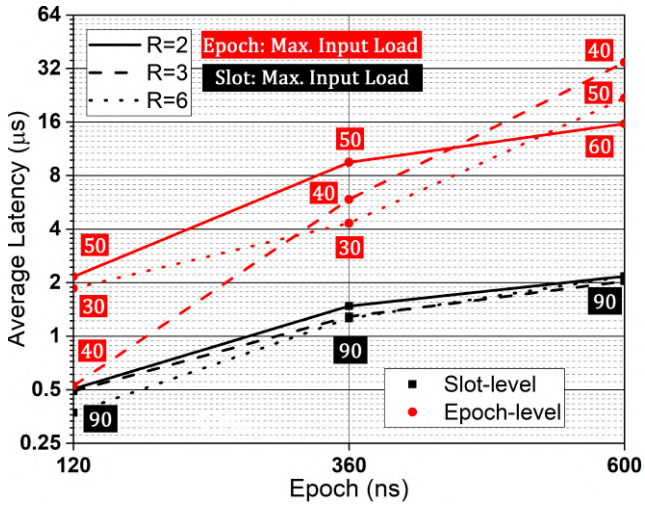


Figure 6.7: Average latency vs Epoch size at maximum operable load.

Figure 6.7 shows the effect of varying epoch size on average latency. The numbers in black and red indicate the load at which the latency is measured in slot-level and epoch-level algorithm respectively. As shown, epoch-level is able to handle much less load compared to slot-level algorithm. This effect is also evident when showcasing the growth of the scheduler buffer with input traffic load.

6.5 Latency Distribution

6.5.1 Epoch vs Slot level Tuning/Scheduling

All requests with a successful grant (contain both birth and grant timestamps) are considered for these latency measurements. For 2000 epochs, considering all N-ports and R requests/blade/epoch, a range of successful requests came out with

grants. In Fig. 6.8(a-i), the distribution of latency is shown using a cumulative distributed function (CDF) for different epoch lengths ($\in 120, 360, 600$ns) and input loads ($\in 50, 70, 90\%$). Each row in Fig. 6.8 shows the scheduling behaviour under different epoch sizes - 120ns, 360ns and 600ns respectively. The curves clearly show a shift to the right in both the median latency and tail latency. The proportion of this shift in average, median and tail latency is directly proportional to that of the epoch size. When epoch sizes are increased 3 times (360ns) or 5 times (600ns), the latency also increases by the same factor. Within each figure, the best and worst case for varying values of R and TD are shown to showcase how it affects the latency for both the slot-level and epoch-level algorithm. A sharp rise in the CDF indicates fast switching with minimum latency, while a slower and gradual rise indicates high latency. The inset in each Fig. 6.8(a-i) shows the behaviour of the tail end of the CDF showing the packets that are buffered for several epochs. This is important to consider because the performance of applications in current data center networks are limited by the tail latency; applications have to wait for hundreds of milliseconds and hang at very high workloads [19].

It is important to understand, however, that the slot-level algorithm can achieve

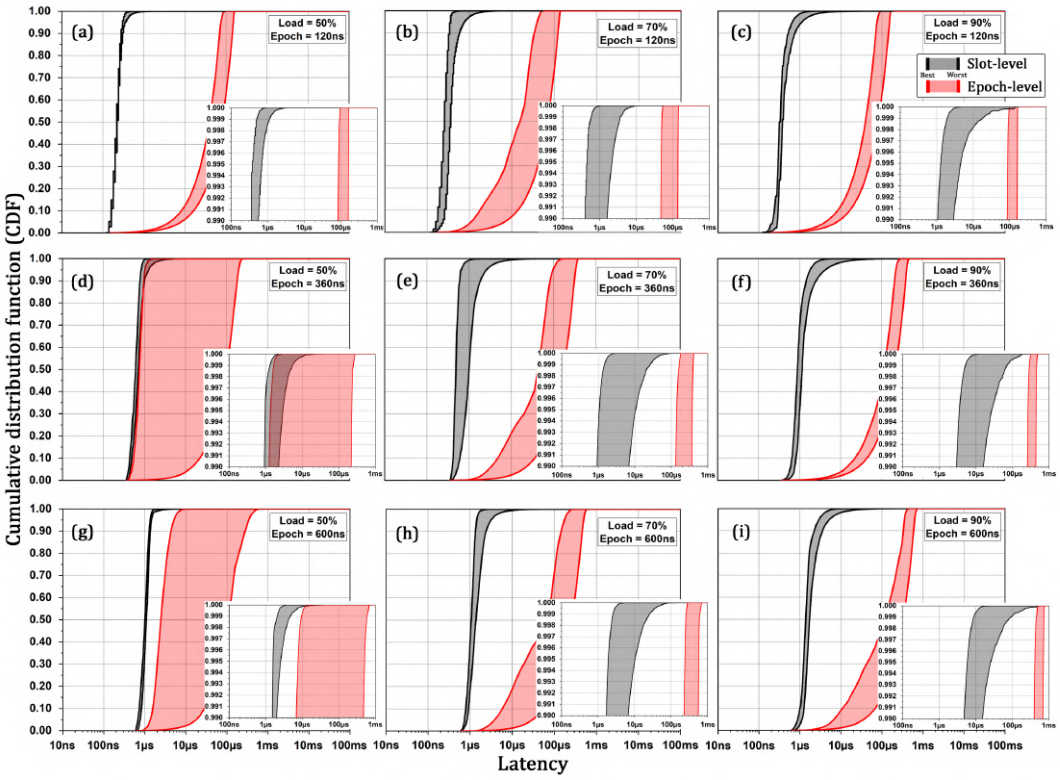


Figure 6.8: Scheduler latency CDF distribution (inset: tail) for varying values of R, epoch sizes, TD.

unsaturated throughput above 90%, while epoch-level algorithms can have their saturation point between 35-62%. This means that the red curves represented in Fig. 6.8 can only show the best case with many requests still remaining in the buffer beyond the saturation point. The slot-level algorithm shows a two orders of magnitude lower worst-case median and tail latency compared to epoch-level algorithm for all values three values of input load. In a 120ns epoch in Fig. 6.8(a-c), the slot-level algorithm achieves a best-case median and tail latency of 0.4μs and a tail latency of 5μs respectively, compared to 35 μs median and 99μs achieved by epoch-level algorithm. The worst-case tail in the slot-level algorithm at 90% input load is at around 120μs. This corresponds to latency of almost 1000 epochs and it is mainly caused by few small packets (<0.05%) at high load.

6.5.2 At maximum operable loads

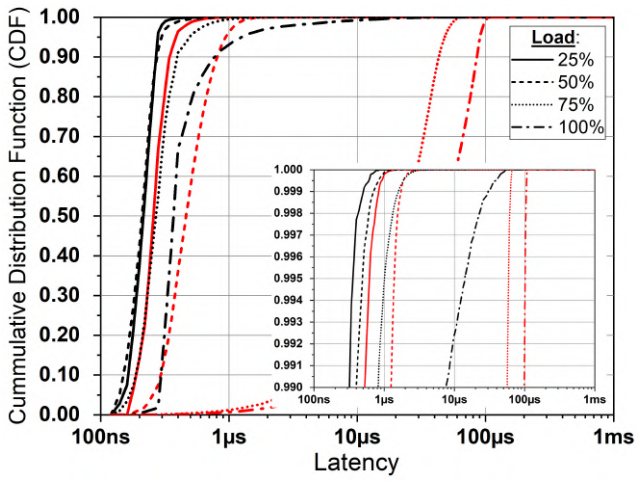


Figure 6.9: Epoch vs Slot level tuning/scheduling: Latency distribution at 25%, 50%, 75% and 100% input loads

The effect of latency on smaller loads at 120 ns epoch and N=64 were also explored. The slot-level scheduling delivers a median latency of just 270 ns (almost 2-orders of magnitude lower than epoch level (24 μs)) and 4.7 μs (14 times lower than epoch level (66 μs)) tail latency at 100-percentile loads. This is due to substantially increased wavelength usage. As shown, the tail is also substantially low (in the order of O(1μs)) when operating below 75% for the slot-level scheduling algorithm. Hence, two conclusions can be drawn from these latency distribution results. The first is that wavelength locking for an entire epoch (epoch-level) substantially reduces epoch. The second is that the coarse and fine level scheduling employed in the slot-level algorithm efficiently use iterations to remove timeslot fragmentation and substantially reduce latency. The following two sub-sections focuses solely on the performance of slot-level tuning/scheduling.

6.5.3 Larger racks, smaller epochs

Similar to the throughput analysis, here, an analysis on latency as the rack scales is shown for the slot-level tuning/scheduling. The latency of multiple operable loads (50, 70 and 90%) are also included in the analysis)

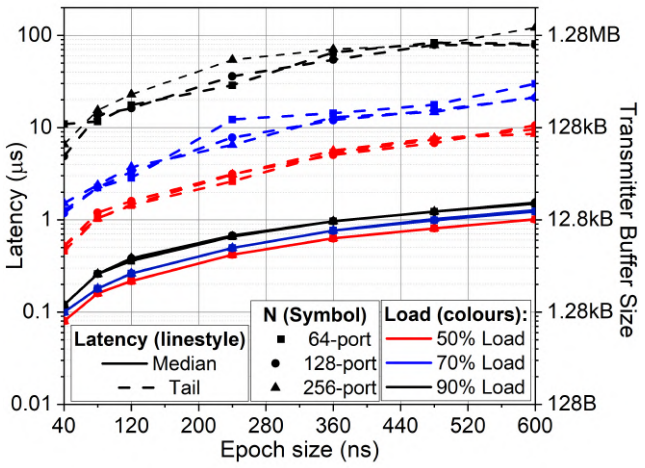


Figure 6.10: Scaling PULSE racks and reaching shorter epochs (median/tail latency vs epoch size) for varying loads

Since scheduler duration matches the epoch size, longer epochs also result in latency increase. Fig. 6.10 showcases the median and tail scheduling latency, excluding propagation and transceiver (serialization, coding) delays, of (N=) 64, 128 and 256-server PULSE racks for 2000 epochs at 50%, 70% and 90% input loads for different epoch sizes (40-600ns). Sub-μs median latency is achieved for epoch sizes less than 360ns. At 90% input load (N=256), for 40, 80 and 120ns epochs, the median latency is 120, 260 and 383ns respectively. The tail latencies are less than 2 orders of magnitude higher at 6.6, 15.4 and 22.9μs. While awaiting the scheduler's response, the transmitters have to buffer the data. Hence, the transmitter buffer size required (Fig. 6.10 right axis) to support these median and tail latencies is less than 2.56 and 512kB respectively. The control system of PULSE scales with high tolerance to latency as N scales (upto 256 is shown in Fig. 6.10) and has no scaling dependency on x.

6.5.4 2-plane

The latency impact of the 2-plane architecture is affected as the same scheduler now has to handle more sub-stars. This results in a spread out collection of buffers that grow if not scheduled efficiently, as the number of sub-stars (P) increases.

In Fig. 6.11, the median and latency are shown at maximum operable loads. For P=8, the operable loads are below 92% for epochs smaller than 240 ns. P=1

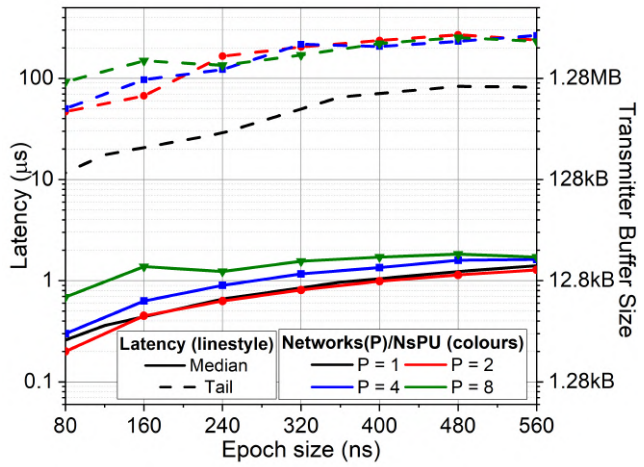


Figure 6.11: Average latency vs Epoch size at maximum operable load.

represents the comparison scenario, where a 1-plane architecture is used. An epoch size of 240 ns is a better operating point for P=8 with median latency at 1.23 μs and a tail of 145 μs. For $P = 1$, 2 and 4, a close to equivalent median latency to $P = 1$ is achieved. However, a 3-4$\times$ penalty is experienced in the tail latency. Specifically, the tail latency is 4$\times$ at 240 ns epoch compared to $P = 1$ and it decreases to $< 3\times$ for 560 ns epoch.

6.6 Scheduler Buffer

In every iteration, buffered requests are given more priority. Although this is the case with both the algorithms, the iteration management in slot-level algorithm has an improved performance. This is expected because the wavelength locking problem is removed by introducing wavelength tunability before each timeslot. The following fig. 6.12 shows the scheduling efficiency by showcasing the growth of the buffer at varying loads when running in both the epoch-level and slot-level algorithms at 120 ns for 2000 epochs.The buffer at the scheduler holds each request as a structure of destination, slot-size, the request number and epoch number information. A total of 3 bytes/blade is required for each request in the control plane to buffer the structure for up to 4096 epochs. Fig. 6.12 shows the growth in buffer size at the scheduler for TD=1 at 120ns epoch for both the scheduling algorithms:(a-c) for epoch-level scheduling and (d-f) for slot-level scheduling.

The log scale explains the wild oscillations at the bottom of each graph. At these loads, where the oscillations are near the floor, the buffer is being cleared as it is being filled. However the increasing loads imply that the buffer is ever-growing and the scheduler cannot operate at this load for multiple epochs. As shown, the epoch-level algorithm cannot take loads greater than 30%, 50% and 60% at R =

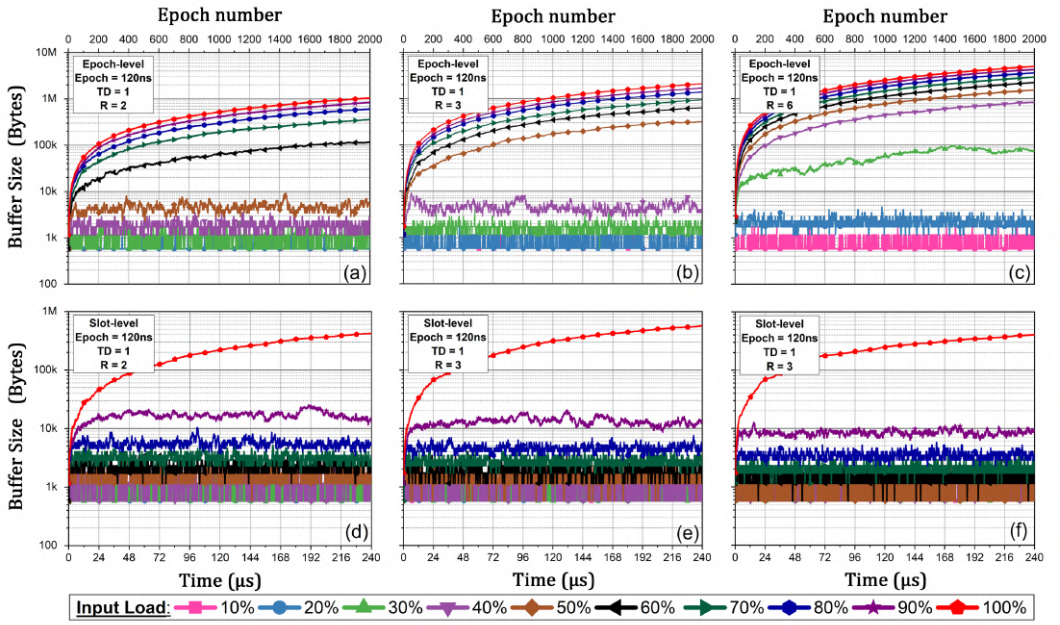


Figure 6.12: Scheduler Buffer size required to handle varying input loads for 2000 epochs

6, 3 and 2 respectively. As the number of requests per blade, R, grows, the load that the epoch-level algorithm drops. In contrast, the slot-level algorithm is able to handle higher loads up to 90% and is tolerant to the value of R. Hence, the slot-level algorithm has very high scheduling efficiency that is tolerant to a range of workloads.

6.7 Summary

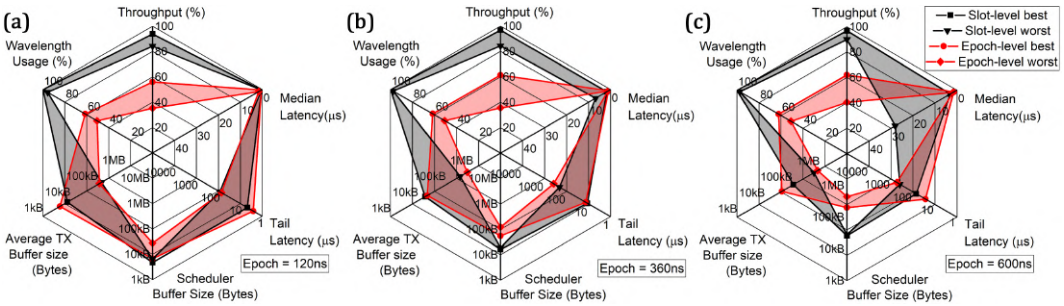


Figure 6.13: Radar plot - overall performance of epoch-level and slot-level scheduling algorithms at maximum operable load.

In this chapter, the performance of PULSE NsPU (slot-level scheduling algorithm) was evaluated in terms of throughput, latency, latency distribution (median, average and tail) and compared with epoch-level scheduling. The impact of increasing rack size and reducing epoch sizes on throughput and latency were studied. The performance was also evaluated for scheduling for a 2-plane architecture, where one

NsPU schedules for multiple stars. In Fig. 6.13(a-c), a radar plot showing six different axes: throughput, median latency, tail latency, average scheduler buffer size, average transmitter buffer size and wavelength usage, for 120ns, 360ns and 600ns epoch sizes. For all values of TD and R, the best and worst values for identified to make this plot. A curve that has a large radius or has a large opening shows the most efficient algorithm with high throughput and maximal wavelength usage, low latency and buffer size. The best and the worst values are shown in all axes to show the variance or the range of change. Fig. 6.13 clearly shows how the slot-level algorithm performs better than the epoch-level scheduling algorithm at epoch sizes of 120, 360 and 600 ns. The throughput and wavelength usage of slot-level algorithm is above 90% and tolerant to changes in epoch size compared to epoch-level algorithm 35-62%. The variance in median latency of slot-level scheduling is lowest for a 120ns epoch, compared to epoch-level scheduling. Maintaining a tolerant buffer size for the scheduler and transmitter, the slot-level scheduling has a better overall performance compared to epoch-level scheduling. In addition to this, the slot-level algorithm was shown to achieve a sustained throughput of above 92% for 64-, 128- and 256-blade racks with median latency as low as 120 ns for a 40 ns epoch and tail latency of 6.6-10 μs. Moreover, scaling to the 2-plane architecture by using one NsPU for multiple stars achieves $>$95% throughput for epoch sizes $>$240 ns for P= 2,4 and 8. At 240 ns epoch, the median latency for the 2-plane architecture is found to between 700 ns to 1.4 μs while the tail is found to be below 200 μs. The growth of the buffer within the scheduler at increasing traffic loads are also reviewed showing that the slot-level has a high maximum operable load at 90% for the PULSE NsPU compared to 30-50% for the epoch-level scheduling algorithm.

Chapter 7

Conclusion

7.1 Executive Summary

Data centers are the pillars that hold the global data sphere. To sustain performance in an exponential data growth environment, networks in today's data center must scale, perform robustly while keeping power and cost to a minimum. Moreover, as Moore's law is coming to an end, ASIC limits are also limiting performance boost on network devices and switches. Hence, migrating to optically switched networks can potentially offer performance advantages. However, optical packet switching must deal with complex controllers, latency and buffering (either at the node or using O/E-E/O for electronic buffers), network complexity, packet loss/retransmission and other associated feature replication problems. By comparison, optical circuit switching technology is viewed as a potential solution due to the inherent advantage of simplicity and low power consumption and cost. Traditionally, a major disadvantage in OCS technology is the long time (O(100 μs-10 ms)) they take to (a) **reconfigure** and (b) **compute** a reconfiguration. This work has brought out a wavelength-timeslot switching transceiver architectures that reduces reconfiguration time to 500 ps and a custom ASIC design that reduces the computation time to 40 ns.

Chapter 2 reviews the limitations of EPS and OPS, compares leading OCS technology with PULSE and discusses the requirements of the NsPU. Although powerful and efficient processing units are being designed in ASIC that can perform highly complex computational, I/O bandwidth limitations severely degrade the performance. Hence, de-coupling of TX/RX I/O from the switching engine in order to maximize performance (out-of-band switching) has been proposed. Highlighting the increasing energy consumption of electronic network solutions, their latency and the limits that are being reached due to the saturation of Moore's law,the need for optical networks and switching architectures is justified. The complexity involved in engineering an optical packet switching technology is a major limitation, as it

attempts to keep up with and provide all the features of a state-of-the-art electronic packet switch. By comparison, optical circuit switch network can provide a simpler yet stable solution to tackle the networking problems. Identifying the stand of key research in OCS, a research gap is found when exploring optical circuit switching in the nanosecond regime (reconfiguration and computation). Hence, a flat broadcast-and-select network with fast wavelength-timeslot switched transceivers and parallel hardware scheduler is explored for employment in the nanosecond switching and allocation regime.

The initial proposal that UCL (my previous supervisors) made in collaboration with Microsoft Research [24] focused primarily on one switching element that is built with a flat high-radix passive core, tunable DS-DBRs for transmission and coherent-receivers for reception. The fast reconfiguration is achieved by pushing active switching to the end-nodes (transceiver-based switching) and creating a passive optical switch fabric in the core. With this simple architecture, wavelength was proposed to be tuned within 200 μs and the tuning time was synchronized once every 2 μs (epoch size). Due to the limitation on the number of wavelength channels that the DS-DBR laser can support and an acceptable over-subscription that already exists within current data centers, 80 wavelength channels were proposed for sharing between 1000-ports. Moreover, the control plane worked in phases of collection, computation and distribution, leaving only 1μs for a hardware scheduler to compute optimal matching and resource utilization. Exploring efficiently scalable hardware elements (round-robin arbiter), a parallel scheduler for wavelength-timeslot allocation was designed and the working principle of each block is reviewed in Chapter 3. The reconfiguration cycle is defined by the complexity and scalability of the controller. Traditionally, OCS systems have adopted software-based or FPGA-based solutions in order to deal with circuit or reconfiguration computation; however, they cannot achieve sub-microsecond scheduling. A schedule-less solution remains ignorant and unbothered about the traffic in the network. An ASIC-based parallel hardware scheduler with round-robin arbiters is proposed. It uses three pipelined parallel core modules: Node Contention Resolution (NCR), Wavelength Decision (WD) and Wavelength Contention Resolution (WCR). The individual core modules in the NsPU scheduler are shown to be scalable with a clock speed of 138 MHz (1024-ports) when synthesized on 45 nm CMOS ASIC. At the end of this chapter, the limitations in scheduling performance caused by the network architecture, transceiver architecture and the control architecture has also been brought out.

In chapter 4, the performance of the scheduling algorithm is reviewed by comparing against software wavelength assignment heuristics. The proposed parallel design is able to solve the resource mapping (matching) problem with high effi-

ciency, achieving equivalent performance with maximal matching and software-based heuristics. Variations of the hardware schedulers are designed but to see negligible or minor improvement in performance. Hence, the key bottlenecks that are created by the network and transceivers (and not by the scheduler) are identified.

In this thesis, I propose PULSE, an OCS data center network architecture that achieves sub-nanosecond reconfiguration using optical devices (SOA, DS-DBR, Co-Rx) and a custom ASIC based hardware design that achieves sub-microsecond reconfiguration cycle. The proposed optical transceiver architectures, as shown in Chapter 5, equip ultra-fast tunable DS-DBRs lasers and SOAs at the transmitter and coherent-receivers, DS-DBR (LOs) and SOAs at the receiver for sub-nanosecond wavelength/timeslot selection (500 ps). Sub-nanosecond fast switching speeds allows the OCS system to cater and establish circuits across the proposed any-to-any network at packet-level granularity. Moreover, in order to scale PULSE, a novel 2-plane architecture that supports multiple planes (or clusters) by introducing splitting and re-coupling after/before the transceiver/receiver is introduced. A central controller that schedules and allocates wavelength/timeslot resources at sub-microsecond timescales (to minimize latency) is required. In order to deal with the extremely heavy requests within sub-microsecond timescales, the PULSE NsPU was developed based on the parallel hardware scheduler design. Additional modifications were made to accommodate timeslot level wavelength allocation, improve wavelength allocation management and iteration and buffer management. Two phases of coarse and fine level slot allocation in co-ordination with the above methods to maximize throughput. Chapter 5 explains these changes in detail and characterizes the network in terms of latency overhead, scalability, power and cost.

In Chapter 6, the performance of PULSE is reviewed. A high throughput of 90% is shown at 100% input load that is tolerant to epoch size and traffic. While extending to larger racks (stars), a sustained throughput of above 92% is shown for 64-,128- and 256-blade racks, although more blades would require the transceivers to support a larger wavelength span. By modifying the NsPU to accommodate for the 2-plane PULSE architecture, the scheduler is shown to achieve above 95% throughput for P=8 for above epochs of 240 ns epoch size or longer at 100% input load. I also showcase a wavelength usage of close to 100%, when tuning and scheduling using the slot-level scheduling algorithm. An average latency of 4 μs is showcased with 120 ns epochs for the slot-level scheduling whilst achieving high operable loads, resulting in low (16 kB) transceiver buffer sizes. The CDF of slot-level scheduling shows that a median latency of 270 ns and 4.7 μs is achieved at 100% input loads. When supporting larger racks, the median latency is shown to be tolerant to rack size, traffic type and load, while the tail latency has a slight depen-

dency on load. A minimal impact on latency is also showcased when employing the 2-plane PULSE architecture. Briefly highlighting the limits posed by buffer growth on beyond 90% input load, the network performance from a scheduling perspective has been summarised.

The thesis has shown the feasibility of an ultra-fast OCS data center network architecture, which demonstrates:

- an implementable and scalable ASIC-based hardware algorithm that achieves sub-microsecond computation time.
- a guaranteed performance tolerant to scaling, traffic type and load.
- sustained high throughput and low latency
- low power transceiver architecture, relative to state-of-the-art EPS networks.
- low normalized cost per path, relative to state-of-the-art EPS networks.

These algorithms could be optimized even further to support higher performance, but this research lays the foundation for employing ASIC hardware for future resource-switched slotted optical switching technology.

7.2 Further Work

PULSE is a scalable, flexible and modular architecture. Although novel scheduling, switching and architectural solutions are proposed, there are areas where more research can be done to provide a better insight and stand to establish the practicality of the network. In this section, some of these areas are briefly highlighted.

Starting with the transceiver architecture, the DS-DBR lasers and coherent-receiver module can be investigated to support more wavelengths and faster tuning time. Increasing wavelengths (to 256) can increase rack size to support larger racks and the NsPU is already capable of handling large rack OCS system. On the other faster tuning would minimize the number of DS-DBRs required at the transmitter and thereby, the power and the cost. In terms of the 2-d plane network architecture, broadcast-and-select requires practical evaluation in the optical physical layer to verify how much of a split can be accommodated. With SOAs proposed to amplify signals in the scalable multi-star, a minimum SNR at the receiver with low BER would prove the scalability of PULSE. If required, an additional SOA can be added at the receiver to accommodate more of a split and it would increase network energy. The network architecture, at the moment, has low fault tolerance; if one sub-star fails, the connection to a rack is lost. A multi-routed (multiple flat stars per end point, M) network topology could introduce fault tolerance and reduce contention enabling the scheduler to handle more request loads (resources increase to $W \times T \times$

M). However, this requires an additional route selection module with contention resolution in PULSE NsPU that is shared between all *M* sub-stars globally updated after every iteration.

Another aspect of PULSE that requires more research is timing and synchronization. Source synchronous techniques or a distributed clocking network must be implemented to verify synchronization within sub-nanosecond resolutions across multiple blades and the synchronization scalability must also be evaluated. CDR locking has been demonstrated by using phase caching techniques, causing a minimum overhead [3]; however, the scalability of the technique has to be explored.

A brief overview of a control network architecture with back of the envelope calculations was shown to use 2 Gbps transceivers in Chapter 3. The developed PULSE NsPU also needs to be co-hosted with the source racks with a simple network that collects requests and distributes grants from *N* nodes. A robust development of the control plane network with minimal impact on cost, power and latency is required (with fixed wavelengths or existing electronic networks (NoC)) to interconnect the scheduler.

PULSE NsPU requires further optimization and hardware re-arrangement in terms of buffer and iteration management to support for more of a split (P=16 or 32). A faster CMOS technology (like 7 nm) can be used for improved performance and even smaller epochs as data rates head towards 800 G [69]. The problem with catering for more splits is that it minimizes the number of grants generated per iteration. A controlled coarse and fine management with the aid of multi-star request processing per iteration (instead of 1 star at a time) - flexible re-use of arbiters by multi-star requests to maximize grants per iteration would greatly improve the algorithm throughput and enable a scheduler ASIC to cater multiple star with tolerance to hotspot and clustered traffic. The simulation platform can be extended to emulate traffics that are multi-cast and broad-cast, as this thesis covers only the scope of uni-cast traffic.

An FPGA demonstration of a small-scale sub-microsecond wavelength and timeslot allocation with PULSE NsPU would greatly benefit testing and prototyping before an ASIC design is finalised.

Appendix A

Scheduling Algorithm Pseudocodes

A.1 Iteration/Buffer Management

Require: N Set of nodes, R requests from each node, I scheduler iterations, W wavelengths, B buffer size.

Ensure: Allocate $Q_s < r,d,t >$ queues per source based on iteration number i, where s is source node, r is request number, d is destination node, t is timeslots requested.

1: **procedure** ITERATION/BUFFER MANAGEMENT($Q_s < r,d,t >$)

▷ **Initialization**

2: i_{buf} = ceil($Size_{buf}/W \times R_p$) ▷ $Size_{buf}$ is the total requests in the buffer, R_p (1.6-2.5) is the buffer coefficient.

3: **while** $i \leq I$ **do** ▷ Run until iterations are left

4: $Q < r,d,t > = i \leq i_{buf}$? $Q < r_b,d_b,t_b >$: $Q < r_s,d_s,t_s >$

▷ r_b, d_b, t_b and r_s, d_s, t_s are request number, destination and timeslots requested in buffer and server respectively

5: **end while**

6: **end procedure**

A.2 Epoch-level Scheduler Algorithm

Require: Set of nodes N, requests from each node R, wavelengths W, timeslots per epoch T, scheduler iterations I, N circular queues, one for each node, $Q_s < r,d,t >$.

Ensure: Allocate time-slots and wavelengths resources with assignments subject to switch constraints: limits of W and T.

1: **procedure** SCHEDULE COMPUTATION(W,T)

▷ **Initialization**

2: $slot[w] = 0, \forall w \in W$ ▷ $slot[w]$ = slot availability

3: $\lambda_{full}[w] = 0, \forall w \in W$ ▷ λ_{full} = wavelength availability

4: $Tx[s] = [\,], Rx[d] = [\,], \forall s,d \in N$ ▷ Tx and Rx are wavelengths of source

node s and receiver node d respectively

5: **while** $i \leq I$ **do** ▷ Run until iterations are left

6: $pointer = (i-1)\%R+1$ ▷ Deal with 1 req/node

▷ **1. Node Contention Resolution**

7: **for** each source s of N **do**

8: $Req[s,<d>] \leftarrow Q_s.pop(r = pointer)] = 1$

9: $size[s] = <t> \leftarrow Q_s.pop(r = pointer)$

10: **end for**

11: ncr = **Arbiter**(Req) ▷ Parallel grants

▷ round robin arbiters make winning grants last priority

▷ **2. Wavelength Decision (function WD)**

12: **for** each source s of N **do**

13: $D = Req[s]$

14: **if** ($Tx[s]$=0 & $Rx[D]$=0 & $ncr[s] \neq 0$) **then**

15: $Reqw[s] =$ **random**(W)

16: **else if** ($Tx[s] \neq 0$ & $Rx[D]$=0 & $ncr[s]$=1) **then**

17: $Reqw[s] = Tx[s]$

18: **else if** ($Tx[s]$=0 & $Rx[D] \neq 0$ & $ncr[s]$=1) **then**

19: $Reqw[s] = Rx[D]$

20: **else if** ($Tx[s] \neq 0$ & $Rx[D] \neq 0$ & $ncr[s]$=1) **then**

21: **if** ($Tx[s] = Rx[D]$) **then**

22: $Reqw[s] = Rx[D]$

23: **else**

24: $Reqw[s] = 0$ ▷ Invalidated, buffered

25: **end if**

26: **end if**

27: **end for**

▷ **3. Wavelength, Timeslot Allocation**

28: wcr = **Arbiter**$(Reqw)$ ▷ Resolve wavelength contention

29: **for** each w of W **do** ▷ Slot update (function SU)

30: s = **find**$(wcr == w)$

31: **if** ($size[s] \geq (T - slot[w])$) **then**

32: $slot[w] = T, \lambda_{full}[w] = 1$

33: $size[s] = size[s] - (T - slot[w])$

34: **else**

35: $slot[w] = slot[w] + size[s]$

36: **end if**

37: **end for**

38: **for** each s of N **do** ▷ Wavelength update
39: $D = Req[s]$
40: **if** $(wcr[s] \neq 0)$ **then**
41: $Tx[s] = wcr[s]$
42: $Rx[D] = wcr[s]$
43: **end if**
44: **end for**
45: **end while**
46: **end procedure**

A.3 Slot-level Scheduling Algorithm

1: **procedure** SCHEDULE COMPUTATION(W, T)
▷ **Initialization**
2: $pointer = 0, slot[w] = 0, \forall w \in W$
3: $\lambda_{full}[w] = 0, \forall w \in W$
4: $Tx[s,\tau] = [\,], Rx[d,\tau] = [\,], \forall s,d \in N, \forall \tau \in T$
▷ Tx and Rx are wavelengths of source node s and receiver node d at timeslot τ respectively
5: **while** $i \leq I$ **do** ▷ Run until iterations are left
6: **if** $(pointer \geq T)$ **then** $coarse = 0$
7: **else** $coarse = 1$
8: **end if** ▷ **1. NCR**
9: **for** each source s of N **do**
10: **for** each request r of R **do**
11: $Req[s, < d > \leftarrow Q_s.pop(r)] = 1$
12: $size[s,r] = coarse\ ?\ 1 :< t > \leftarrow Q_s.pop(r)$
13: **end for**
14: **end for**
15: Req_s = **Arbiter**(Req)
16: $[ncr, r_s]$ = **Arbiter**(Req_s) ▷ Two allocators: input/output, where r_s is the array of successful request numbers, 1/node
17: $Reqw$ = **WD**(Req, ncr, Tx, Rx) ▷ **2. WD**
▷ **3. WTA**
18: wcr = **Arbiter**$(Reqw)$ ▷ Resolve wavelength contention
19: $slot$ = **SU**$(size, r_s)$ ▷ Slot update
20: **for** each s of N **do** ▷ Wavelength update
21: $D = Req[s, r_s]$
22: **if** $(wcr[s] \neq 0)$ **then**
23: $Tx[s, \tau : \tau + size - 1] = wcr[s]$

$Rx[D, \tau : \tau + size - 1] = wcr[s]$
end if
end for
$pointer = pointer$ + **Max**($size$)
end while
end procedure

A.4 Software Scheduling Heuristics

Require: Set of nodes N, set of requests from each node R, set of wavelengths W, set of stars T, slots per epoch L, maximum transceivers per wavelength M, circular queue of node pairs with non-zero demand, $Q < i, j, reqsize >$, where i is the source node, j is the destination node and *reqsize* is the number of slots requested.

Ensure: Assign time-slots across wavelengths and stars to obtain (a) Random, (b) Least loaded (LL) or (c) Least used (LU) assignment subject to switch constraints: limits of M, W and L

procedure SERIAL WAVELENGTH ASSIGNMENT(W, L, T)
$m_w[w, \tau] = 0, slot[w, \tau] = 0, \forall w \in W, \forall \tau \in T$ ▷ $m_w[w, \tau]$ and $slot[w, \tau]$ are the number of transmitters and slots assigned to wavelength w in star τ respectively
$\lambda_{full}[w, \tau] = false, \forall w \in W, \forall \tau \in T$ ▷ λ_{full} = wavelength availability
$Tx[i, \tau] = [\,], Rx[j, \tau] = [\,], \forall i, j \in N, \forall \tau \in T$ ▷ where Tx and Rx are the wavelength assigned to node i transmitter and node j receiver in star τ respectively
while $Q \neq [\,]$ **do** ▷ Run until queue is empty
$< i, j, reqsize > \leftarrow Q.pop()$
▷ Only one of the following three algorithms is used at any given time
[λ, star] = **random**(W, T) ▷ Random wavelength w in random star τ
[λ, star] = **leastloaded**(W, T) ▷ Find wavelength w using min slots
[λ, star] = **leastused**(W, T) ▷ Find wavelength w using min transceivers
if $t \in T : (\{Tx[i,t] == Rx[j,t], Tx[i,t] \neq [\,], !\lambda_{full}[Tx[i,t], t]\} \neq \Phi)$ **then**
Let τ in $\{Tx[i,t] == Rx[j,t], Tx[i,t] \neq [\,], !\lambda_{full}[Tx[i,t], t]\}$
Grant = True, $w = \lambda$, $\tau = star$
break
▷ Check for common assigned wavelength across all stars
else if $(Tx[i, star] == [\,] \&\& Rx[j, star] == [\,] \&\& [\lambda_{full}[\lambda, star])$ **then**
Grant = True, $w = \lambda$, $\tau = star$
break
▷ Check if unassigned Tx[i,~], Rx[j,~] are available across all stars

else if $t \in T : (\{Tx[i,t] = [\,] \&\& Rx[j,t] \neq [\,] \&\& !\lambda_{full}[Rx[j,t],t]\} \neq \Phi)$ **then**
    Grant = True, $w = Rx[j,t]$
    break
        ▷ Check if unassigned Tx[i,~] is available across all stars
else if $t \in T : (\{Tx[i,t] \neq [\,] \&\& Rx[j,t] = [\,] \&\& !\lambda_{full}[Tx[i,t],t]\} \neq \Phi)$ **then**
    Grant = True, $w = Tx[i,t]$
    break
        ▷ Check if unassigned Rx[j,~] is available across all stars
else
    Grant = false
end if
if (Grant = true) **then**
        ▷ If request is granted, update time-slot and wavelength registers
    if $(reqsize > L - slot[w,\tau])$ **then**
        $slot[w,\tau] = L, reqsize = requsize - (L - slot[w,\tau])$
        Grant = false
            ▷ Limited or no slots available, creating incomplete grants
    else if $(reqsize \leq L - slot[w,\tau])$ **then**
        $slot[w,\tau] = slot[w,\tau] + req_size$
                ▷ Slots are available, all requested slots are granted
    end if
else
    $Q.push(< i, j, reqsize >)$
                ▷ Enqueue unattended requests
end if
if $(slot[w,\tau] == L || m_w[w,\tau] == M)$ **then**
    $\lambda_{full}[w,\tau]$ = true
                ▷ Update wavelength full register
end if
end while
end procedure

Bibliography

[1] ITU. "Internet Users — forecast", 2017. [Online]. Available: http://stats.areppim.com/stats/stats_internetxfcstx2017.html.

[2] Cisco. "Cisco Visual Networking Index: Forecast and Methodology, 2017-2022", 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[3] K. Clark, H. Ballani, P. Bayvel, D. Cletheroe, T. Gerard, I. Haller, K. Jozwik, K. Shi, B. Thomsen, P. Watts, H. Williams, G. Zervas, P. Costa, and Z. Liu. "Sub-Nanosecond Clock and Data Recovery in an Optically-Switched Data Centre Network". In *2018 European Conference on Optical Communication (ECOC)*, pages 1–3, Sep. 2018.

[4] Marc Taubenblatt Marc. "Optical Interconnects for Data Centers", Sept 2019.

[5] A. C. Funnell, K. Shi, P. Costa, P. Watts, H. Ballani, and B. C. Thomsen. "Hybrid Wavelength Switched-TDMA High Port Count All-Optical Data Centre Switch". *Journal of Lightwave Technology*, 35(20):4438–4444, Oct 2017.

[6] Bernadette Johnson. "How Data Centers Work", Oct 2013. [Online]. Available: https://computer.howstuffworks.com/data-centers.html.

[7] Dave Evans. "Internet of Things: How the next evolution of the internet is changing everything", April 2011. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[8] João Marques Lima. "Exploding the data centre business", Apr 2018. [Online]. Available: https://data-economy.com/exploding-the-data-centre-business/.

[9] Mueen Uddin and Azizah Abdul Rahman. "Server Consolidation: An Approach to make Data Centers Energy Efficient and Green", 2010. [Online]. Available: http://arxiv.org/abs/1010.5037.

[10] 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 Hölzle, Stephen Stuart, and Amin Vahdat. "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network". *SIGCOMM Comput. Commun. Rev.*, 45(4):183–197, August 2015.

[11] Yevgeniy Sverdlik. "Intel: World Will Switch to 'Scale' Data Centers by 2025", 2016. [Online]. Available: http://www.datacenterknowledge.com/archives/2016/04/22/intel-world-will-switch-to-scale-data-centers-by-2025.

[12] Tom Bawden. "Global warming: Data centres to consume three times as much energy in next decade, experts warn", 2016. [Online]. Available: http://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html.

[13] Neil Rasmussen. "Allocating Data Center Energy Costs and Carbon to IT Users", 2009. [Online]. Available: https://uat-s.insight.com/uk01/en-gb/content/media/apc-allocating-costs.pdf.

[14] C E Leiserson. "There's Plenty of Room at the Top: What will drive growth in computer performance after Moore's Law ends?", Jan 2017. [Online]. Available: http://www.neil-t.com/moores-law-and-computer-performance/.

[15] "Scaling Data Center Networks". [Online]. Available: https://www.arista.com/assets/data/pdf/Network_Scalability_AAG.pdf.

[16] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. High-Resolution Measurement of Data Center Microbursts. In *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, page 78–85, New York, NY, USA, 2017. Association for Computing Machinery.

[17] Alison Flood. "Barefoot Networks Unveils Tofino 2, the Next Generation of the World's First Fully P4-Programmable Network Switch ASICs", Dec

2018. [Online]. Available: https://www.globenewswire.com/news-release/2018/12/04/1661637/0/en/Barefoot-Networks-Unveils-Tofino-2-the-Next-Generation-of-the-World-s-First-Fully-P4-Programmable-Network-Switch-ASICs.html.

[18] "Understanding Network Latency in Ethernet Switches", Sep 2018. [Online]. Available: http://www.fiberopticshare.com/network-latency-in-ethernet-switches.html.

[19] Yunjing Xu, Zachary Musgrave, Brian Noble, and Michael Bailey. "Bobtail - Avoiding Long Tails in the Cloud". In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 329–341, Lombard, IL, 2013.

[20] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. "Helios: a hybrid electrical/optical switch architecture for modular data centers". In *SIGCOMM*, 2010.

[21] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. "c-Through: part-time optics in data centers". *SIGCOMM Comput. Commun. Rev.*, 41(4):–, August 2010.

[22] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. "RotorNet: A Scalable, Low-complexity, Optical Datacenter Network". In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 267–280, New York, NY, USA, 2017. ACM.

[23] Joel Webster. "SERIES 7000 - 384x384 port Software-Defined Optical Circuit Switch". [Online]. Available: https://www.polatis.com/series-7000-384x384-port-software-controlled-optical-circuit-switch-sdn-enabled.asp.

[24] Dan Alistarh, Hitesh Ballani, Paolo Costa, Adam Funnell, Joshua Benjamin, Philip Watts, and Benn Thomsen. "A High-Radix, Low-Latency Optical Switch for Data Centers". *SIGCOMM Comput. Commun. Rev.*, 45(4):367–368, August 2015.

[25] A. Funnell, J. Benjamin, H. Ballani, P. Costa, P. Watts, and B. C. Thomsen. "High port count hybrid wavelength switched TDMA (WS-TDMA) optical

switch for data centers". In *2016 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, March 2016.

[26] Noa Zilberman, Gabi Bracha, and Golan Schzukin. "Stardust: Divide and Conquer in the Data Center Network". In *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation*, NSDI'19, pages 141–159, Berkeley, CA, USA, 2019. USENIX Association.

[27] F. Yan, W. Miao, H.J.S. Dorren, and N. Calabretta. "A novel flat DCN architecture adopting low radix optical packet switches". In P. Kockaert, P. Emplit, S.-P. Gorza, and S. Massar, editors, *Proceedings of 20th Annual Symposium of the IEEE Photonics Society Benelux Chapter, November 26-27, 2015, Brussels, Belgium*, pages 265–268. OPERA-photonics, Brussels School of Engineering, 2015. Due to safety threats in Brussels on November 26-27, 2015, the symposium was rescheduled on February 8-9, 2016.

[28] Nathan Binkert, Al Davis, Norman Jouppi, Moray Mclaren, Naveen Muralimanohar, Robert Schreiber, and Jung Ho Ahn. "The role of optics in future high radix switch design". *ACM Sigarch Computer Architecture News*, pages 437–448, 07 2011.

[29] H. J. S. Dorren, Erik H. M. Wittebol, Rob de Kluijver, Gonzalo Guelbenzu de Villota, Pinxiang Duan, and Oded Raz. "Challenges for optically enabled high-radix switches for data center networks". *Journal of Lightwave Technology*, 33(5):1117–1125, 3 2015.

[30] M. C. Wu, S. Han, T. J. Seok, and N. Quack. "Large-port-count MEMS silicon photonics switches". In *2015 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, March 2015.

[31] S Di Lucente, Nicola Calabretta, Jacques Resing, and H J. S. Dorren. "Scaling Low-Latency Optical Packet Switches to a Thousand Ports". *Journal of Optical Communications and Networking*, Vol. 4:A17–A28, 09 2012.

[32] Qixiang Cheng, Adrian Wonfor, Jinlong Wei, R V. Penty, and I.H. White. "Demonstration of the Feasibility of Large Port Count Optical Switching Using a Hybrid MZI SOA Switch Module in a Recirculating Loop. *Optics Letters*, 39, 09 2014.

[33] Paris Andreades, Kari Clark, Philip M. Watts, and Georgios Zervas. "Experimental demonstration of an ultra-low latency control plane for optical packet

switching in data center networks". *Optical Switching and Networking*, 32:51 – 60, 2019.

[34] S. Cheung, T. Su, K. Okamoto, and S. J. B. Yoo. "Ultra-Compact Silicon Photonic 512 × 512 25 GHz Arrayed Waveguide Grating Router". *IEEE Journal of Selected Topics in Quantum Electronics*, 20(4):310–316, July 2014.

[35] "Mellanox Spectrum-2 Ethernet Switch", 2019. [Online]. Available: `http://www.mellanox.com/related-docs/prod_silicon/PB_Spectrum-2.pdf`.

[36] "Barefoot's Tofino 2 chip delivers 12.8 Tbps switching for 32x400GE". [Online]. Available: `https://www.convergedigest.com/2018/12/barefoots-tofino-2-chip-packs-128-tbps.html`.

[37] "At a Glance: Tomahawk 3 is the first 12.8 Tb/s chip to achieve mass production". [Online]. Available: `https://www.broadcom.com/blog/at-a-glance-tomahawk-3-is-the-first-12-8-tb-s-chip-to-achieve-mass-production`.

[38] Broadcom Inc. "Broadcom Trident 4 Delivers Disruptive Economics for Enterprise Data Center and Campus Networks". *GlobeNewswire News Room*, Jun 2019.

[39] Broadcom Inc. "Broadcom Ships Jericho2, Industry's Highest Bandwidth Ethernet Switch-Router at 10 Terabits per Second". *GlobeNewswire News Room*, Mar 2018.

[40] C. Gray, R. Ayre, K. Hinton, and R. S. Tucker. "Power consumption of IoT access network technologies". In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 2818–2823, June 2015.

[41] "Cisco ME 3800X Series Carrier Ethernet Switch Router Data Sheet", Feb 2014. [Online]. Available: `https://www.cisco.com/c/en/us/products/collateral/switches/me-3800x-series-carrier-ethernet-switch-routers/data_sheet_c78-601950.html`.

[42] R. S. Tucker. "Scalability and Energy Consumption of Optical and Electronic Packet Switching". *Journal of Lightwave Technology*, 29(16):2410–2421, Aug 2011.

[43] Y. Audzevich, P. M. Watts, A. West, A. Mujumdar, J. Crowcroft, and A. W. Moore. "Low power optical transceivers for switched interconnect networks". In *2013 International Conference on Advanced Technologies for Communications (ATC 2013)*, pages 301–306, Oct 2013.

[44] J. Bowers. "Low power 3D MEMS optical switches". In *2009 IEEE/LEOS International Conference on Optical MEMS and Nanophotonics*, pages 152–153, Aug 2009.

[45] N. Farrington, A. Forencich, G. Porter, P. . Sun, J. E. Ford, Y. Fainman, G. C. Papen, and A. Vahdat. "A Multiport Microsecond Optical Circuit Switch for Data Center Networking". *IEEE Photonics Technology Letters*, 25(16):1589–1592, Aug 2013.

[46] He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter. "Circuit Switching Under the Radar with REACToR". In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 1–15, Berkeley, CA, USA, 2014. USENIX Association.

[47] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. "FireFly: A Reconfigurable Wireless Data Center Fabric Using Free-space Optics". *SIGCOMM Comput. Commun. Rev.*, 44(4):319–330, August 2014.

[48] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen. "OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility". *IEEE/ACM Transactions on Networking*, 22(2):498–511, April 2014.

[49] Sangjin Han, Norbert Egi, Aurojit Panda, Sylvia Ratnasamy, Guangyu Shi, and Scott Shenker. "Network Support for Resource Disaggregation in Next-generation Datacenters". In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, pages 10:1–10:7, New York, NY, USA, 2013. ACM.

[50] Theophilus Benson, Aditya Akella, and David A. Maltz. "Network Traffic Characteristics of Data Centers in the Wild". In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 267–280, New York, NY, USA, 2010. ACM.

[51] R. R. Grzybowski, B. R. Hemenway, M. Sauer, C. Minkenberg, F. Abel, P. Muller, and R. Luijten. "The OSMOSIS Optical Packet Switch for Supercomputers: Enabling Technologies and Measured Performance". In *2007 Photonics in Switching*, pages 21–22, Aug 2007.

[52] R. Luijten, W. E. Denzel, R. R. Grzybowski, and R. Hemenway. Optical interconnection networks: The osmosis project. In *The 17th Annual Meeting of the IEEELasers and Electro-Optics Society, 2004. LEOS 2004.*, volume 2, pages 563–564 Vol.2, Nov 2004.

[53] C. Minkenberg, F. Abel, P. Muller, Raj Krishnamurthy, M. Gusat, and B. R. Hemenway. "Control path implementation for a low-latency optical HPC switch". In *13th Symposium on High Performance Interconnects (HOTI'05)*, pages 29–35, Aug 2005.

[54] Kang Xi, Yu-Hsiang Kao, Ming Chien Yang, and H. Jonathan Chao. "Petabit Optical Switch for Data Center Networks", 2010. [Online]. Available: `https://pdfs.semanticscholar.org/577a/708b00fccf2e55de3f2f1c7f210e05a34bff.pdf`.

[55] Aaron Stillmaker and Bevan Baas. "Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm". *Integration*, 58:74 – 81, 2017.

[56] Y. Cheng, M. Fiorani, R. Lin, L. Wosinska, and J. Chen. "POTORI: a passive optical top-of-rack interconnect architecture for data centers". *IEEE/OSA Journal of Optical Communications and Networking*, 9(5):401–411, May 2017.

[57] H. J. Chao, Kung-Li Deng, and Zhigang Jing. "PetaStar: a petabit photonic packet switch". *IEEE Journal on Selected Areas in Communications*, 21(7):1096–1112, Sep. 2003.

[58] N. Terzenidis, M. Moralis-Pegios, G. Mourgias-Alexandris, T. Alexoudi, K. Vyrsokinos, and N. Pleros. "High-Port and Low-Latency Optical Switches for Disaggregated Data Centers: The Hipoλaos Switch Architecture". *IEEE/OSA Journal of Optical Communications and Networking*, 10(7):102–116, July 2018.

[59] M. Moralis-Pegios, N. Terzenidis, G. Mourgias-Alexandris, K. Vyrsokinos, and N. Pleros. "A 1024-Port Optical Uni- and Multicast Packet Switch Fabric". *Journal of Lightwave Technology*, 37(4):1415–1423, Feb 2019.

[60] R. Proietti, Y. Yin, R. Yu, C. J. Nitta, V. Akella, C. Mineo, and S. J. B. Yoo. "Scalable Optical Interconnect Architecture Using AWGR-Based TONAK LION Switch With Limited Number of Wavelengths". *Journal of Lightwave Technology*, 31(24):4087–4097, Dec 2013.

[61] Odile Liboiron-Ladouceur, Assaf Shacham, Benjamin A. Small, Benjamin G. Lee, Howard Wang, Caroline P. Lai, Aleksandr Biberman, and Keren Bergman. "The Data Vortex Optical Packet Switched Interconnection Network". *J. Lightwave Technol.*, 26(13):1777–1789, Jul 2008.

[62] A. Shacham, B. A. Small, O. Liboiron-Ladouceur, and K. Bergman. "A fully implemented 12 /spl times/ 12 data vortex optical packet switching interconnection network". *Journal of Lightwave Technology*, 23(10):3066–3075, Oct 2005.

[63] F. Yan, W. Miao, O. Raz, and N. Calabretta. "OPSquare: A flat DCN architecture based on flow-controlled optical packet switches". *IEEE/OSA Journal of Optical Communications and Networking*, 9(4):291–303, April 2017.

[64] W. Miao, F. Yan, O. Raz, and N. Calabretta. "OPSquare: Assessment of a novel flat optical data center network architecture under realistic data center traffic". In *2016 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, March 2016.

[65] Xiaohui Ye, P. Mejia, Yawei Yin, R. Proietti, S. J. B. Yoo, and V. Akella. Dos - a scalable optical switch for datacenters. In *2010 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 1–12, Oct 2010.

[66] C. Kachris and I. Tomkos. "A Survey on Optical Interconnects for Data Centers". *IEEE Communications Surveys Tutorials*, 14(4):1021–1036, Fourth 2012.

[67] A. Shacham, H. Wang, and K. Bergman. "Experimental Demonstration of a Complete SPINet Optical Packet Switched Interconnection Network". In *OFC/NFOEC 2007 - 2007 Conference on Optical Fiber Communication and the National Fiber Optic Engineers Conference*, pages 1–3, March 2007.

[68] Qi Li et. al. Scaling Star-Coupler-Based Optical Networks for Avionics Applications. *J. Opt. Commun. Netw.*, 5(9):945–956, Sep 2013.

[69] "800G: Coherent versus PAM4 Optical Transceivers Inside Data Centers", Sep 2019. [Online]. Available: https://www.neophotonics.com/800g-coherent-versus-pam4-optical-transceivers-data-centers/.

[70] W. M. Mellette *et al.* "A Scalable, Partially Configurable Optical Switch for Data Center Networks". *JLT*, 35(2):136–144, Jan 2017.

[71] Y. Birk, W. M. Mellette, and E. Zahavi. "Switch Radix Reduction and Support for Concurrent Bidirectional Traffic in RotorNets@. In *2018 Photonics in Switching and Computing (PSC)*, pages 1–3, Sep. 2018.

[72] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. "Integrating Microsecond Circuit Switching into the Data Center". *SIGCOMM Comput. Commun. Rev.*, 43(4):447–458, August 2013.

[73] Yong Cui, Shihan Xiao, Xin Wang, Zhenjie Yang, Chao Zhu, Xiangyang Li, Liu Yang, and Ning Ge. "Diamond: Nesting the Data Center Network with Wireless Rings in 3D Space. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 657–669, Santa Clara, CA, March 2016. USENIX Association.

[74] A. Sugita, A. Kaneko, K. Okamoto, M. Itoh, A. Himeno, and Y. Ohmori. "Very low insertion loss arrayed-waveguide grating with vertically tapered waveguides". *IEEE Photonics Technology Letters*, 12(9):1180–1182, Sep. 2000.

[75] J. Tippinit and W. Asawamethapant. "N×N cyclic AWG with low and uniform insertion losses achieved by introducing array of tapered waveguides". In *2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–6, June 2015.

[76] J. Chen, Y. Gong, M. Fiorani, and S. Aleksic. "Optical interconnects at the top of the rack for energy-efficient data centers". *IEEE Communications Magazine*, 53(8):140–148, August 2015.

[77] A. Azad, M. Halappanavar, S. Rajamanickam, E. G. Boman, A. Khan, and A. Pothen. "Multithreaded Algorithms for Maximum Matching in Bipartite Graphs". In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, pages 860–872, May 2012.

[78] R. R. Hoare, Z. Ding, and A. K. Jones. “A Near-optimal Real-time Hardware Scheduler for Large Cardinality Crossbar Switches”. In *SC ’06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, pages 8–8, Nov 2006.

[79] Hui Zang and Jason P. Jue. “A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks”. *Optical Networks Magazine*, 1:47–60, 2000.

[80] G. S. Zervas, J. Triay, N. Amaya, Y. Qin, C. Cervelló-Pastor, and D. Simeonidou. “Time shared optical network (TSON): A novel metro architecture for flexible multi-granular services”. In *2011 37th European Conference and Exhibition on Optical Communication*, pages 1–3, Sep. 2011.

[81] J. L. Benjamin, A. Funnell, P. M. Watts, and B. Thomsen. “A High Speed Hardware Scheduler for 1000-Port Optical Packet Switches to Enable Scalable Data Centers”. In *2017 IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI)*, pages 41–48, Aug 2017.

[82] Nick McKeown. “The iSLIP Scheduling Algorithm for Input-queued Switches”. *IEEE/ACM Trans. Netw.*, 7(2), April 1999.

[83] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. “Inside the Social Network’s (Datacenter) Network”. *SIGCOMM Comput. Commun. Rev.*, 45(4):123–137, August 2015.

[84] “Low Noise Tunable Laser Designed for high SNR (Signal-to-Noise-Ratio) applications at 1550nm”, 2015. [Online]. Available: `http://www.amstechnologies.com/fileadmin/amsmedia/downloads/5140_purephotonicsdatasheetcollection.pdf`.

[85] William James Dally and Brian Patrick Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[86] Hui Zang and Jason P. Jue. “A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks”. *Optical Networks Magazine*, 1:47–60, 2000.

[87] M. Demange and T. Ekim. “Minimum Maximal Matching is NP-hard in Regular Bipartite Graphs”. In *Proceedings of the 5th International Conference*

on Theory and Applications of Models of Computation, TAMC'08, pages 364–374, Berlin, Heidelberg, 2008. Springer-Verlag.

[88] E. Solomonik and L. V. Kalé. "Highly scalable parallel sorting". In *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pages 1–12, April 2010.

[89] Christian Siebert and Felix Gerd Eugen Wolf. "A scalable parallel sorting algorithm using exact splitting". Technical report, Publikationsserver der RWTH Aachen University, Aachen, 2011.

[90] W. Song, D. Koch, M. Luján, and J. Garside. "Parallel Hardware Merge Sorter". In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 95–102, May 2016.

[91] J. L. Benjamin and G. Zervas. "Parallel Star-coupler OCS Architectures using Distributed Hardware Schedulers". In *2018 Photonics in Switching and Computing (PSC)*, pages 1–3, Sep. 2018.

[92] Kasper Van Gasse, Ruijun Wang, and Gunther Roelkens. "27 dB gain III-V-on-silicon semiconductor optical amplifier with>17 dBm output power". *Opt. Express*, 27(1):293–302, Jan 2019.

[93] R. C. Figueiredo, N. S. Ribeiro, A. M. O. Ribeiro, C. M. Gallep, and E. Conforti. "Hundred-Picoseconds Electro-Optical Switching With Semiconductor Optical Amplifiers Using Multi-Impulse Step Injection Current". *Journal of Lightwave Technology*, 33(1):69–77, Jan 2015.

[94] K. Szczerba, P. Westbergh, J. S. Gustavsson, M. Karlsson, P. A. Andrekson, and A. Larsson. "Energy Efficiency of VCSELs in the Context of Short-Range Optical Links". *IEEE Photonics Technology Letters*, 27(16):1749–1752, Aug 2015.

[95] Alexander Gaeta, Michal Lipson, and Tobias J. Kippenberg. "Photonic-chip-based frequency combs". *Nature Photonics*, 13:158–169, 03 2019.

[96] Davis H. Hartman, Peter J. Delfyett, and S. Zuber Ahmed. "Optical clock distribution using a mode-locked semiconductor laser diode system". In *Optical Fiber Communication*, page FC3. Optical Society of America, 1991.

[97] M. R. Inggs, J. S. Sandenbergh, and S. A. C. Lewis. “Investigation of white rabbit for synchronization and timing of netted radar”. In *2015 IEEE Radar Conference*, pages 214–217, Oct 2015.

[98] V. Natoli *et al.* “A Decade of Accelerated Computing Augurs Well For GPUs”, Jul 2019. [Online]. Available: https://www.nextplatform.com/2019/07/10/a-decade-of-accelerated-computing-augurs-well-for-gpus/.

[99] Tiernan Ray. “Seeking Big A.I. Advances, a Startup Turns to a Huge Computer Chip”, Aug 2019. [Online]. Available: https://fortune-com.cdn.ampproject.org/c/s/fortune.com/2019/08/19/ai-artificial-intelligence-cerebras-wafer-scale-chip/amp/.

[100] T. Baji. “Evolution of the GPU Device widely used in AI and Massive Parallel Processing”. In *EDTM*, pages 7–9, March 2018.

[101] “DARPA FastNICs Program Looks to Accelerate Application Performance by 100x”, Sep 2019. [Online]. Available: https://insidehpc.com/2019/09/darpa-fastnics-program-looks-to-accelerate-application-performance-by-100x/.

[102] G. Zervas *et al.* “Optically Disaggregated Data Centers with minimal remote memory latency: Technologies, architectures, and resource allocation” [Invited]. *JOCN*, 10(2):A270–A285, Feb 2018.

[103] “Photonic ASIC directly integrates 100G optical ports”, Jan 2019. [Online]. Available: https://www.eenewsanalog.com/news/photonic-asic-directly-integrates-100g-optical-ports.

[104] R. Meade *et al.* “TeraPHY: A High-Density Electronic-Photonic Chiplet for Optical I/O from a Multi-Chip Module”. In *OFC*, pages 1–3, 2019.

[105] Tiffany Trader. “Ayar Labs to Demo Photonics Chiplet in FPGA Package at Hot Chips”, Aug 2019. [Online]. Available: https://www.hpcwire.com/2019/08/19/ayar-labs-to-demo-photonics-chiplet-in-fpga-package-at-hot-chips/.

[106] W. Kobayashi, N. Fujiwara, T. Shindo, S. Kanazawa, K. Hasebe, H. Ishii, and M. Itoh. “Ultra low power consumption operation of SOA assisted extended reach EADFB laser (AXEL)”. In *2016 21st OptoElectronics and Communications Conference (OECC) held jointly with 2016 International Conference on Photonics in Switching (PS)*, pages 1–3, July 2016.

[107] M. Nagatani *et al.* "A 3-Vppd 730-mW Linear Driver IC Using InP HBTs for Advanced Optical Modulations". *CSICS*, pages 1–4, 2013.

[108] Klaus Grobe and Michael Eiselt. *Wavelength Division Multiplexing: A Practical Engineering Guide*. Wiley Publishing, 1st edition, 2013.

[109] P. Minzioni *et al.* "Roadmap on all-optical processing". *Journal of Optics*, 21(6), 5 2019.

[110] Toshihide Yoshimatsu, Masahiro Nada, Manabu Oguma, Haruki Yokoyama, Tetsuichiro Ohno, Yoshiyuki Doi, Ikuo Ogawa, Hiroshi Takahashi, and Eiji Yoshida. "Compact and high-sensitivity 100-Gb/s (4× 25 Gb/s) APD-ROSA with a LAN-WDM PLC demultiplexer". *Opt. Express*, 20(26):B393–B398, Dec 2012.

[111] Arista Networks. "7050X3 Series Network Switch - Quick Look", Aug 2019. [Online]. Available: `https://www.arista.com/en/products/7050x3-series-network-switch-datasheet`.

[112] Benjamin Lee. "Platforms for Integrated Photonic Switching Modules, OFC Workshop", 2019.

[113] "High Speed Ethernet Optics Report", 2018. [Online]. Available: `https://www.lightcounting.com/DataCenter.cfm`.

[114] "Arista Networks 7170-32C Layer 3 Switch". [Online]. Available: `https://www.pcnation.com/web/details/5JV204/Arista-Networks-Hardware-7170-Programmable-32-X-100GBE-Qsfp-Switch-Exp-Mem-Ssd-Front-To-DCS-7170-32C-M-F`.

[115] "HPE Arista 7328X Switch Chassis". [Online]. Available: `https://www.pcnation.com/web/details/8V6585/HPE-Arista-7328X-Switch-Chassis-JH822A`.

[116] CompSource Inc. "HPE Arista 7516R Switch Chassis". [Online]. Available: `https://www.compsource.com/buy/JQ125A/Hp-195`.

[117] Thanasis Theocharidis. "Public executive summary of the final Project Periodic Report", Mar 2016. [Online]. Available: `https://cordis.europa.eu/docs/projects/cnect/4/318714/080/reports/001-ASTRONPublicexecutivesummaryofthefinalProjectPeriodicReport.pdf`.

[118] G. Arevalo *et al.* “Optimization of multiple PON deployment costs and comparison between GPON, XGPON, NGPON2 and UDWDM PON”. *OSN*, 25, 03 2017.

[119] Z. Jia *et al.* “Digital Coherent Transmission for Next Generation Cable Operators’ Optical Access Networks”, 2017. [Online]. Available: https://www.nctatechnicalpapers.com/Paper/2017/2017-digital-coherent-transmission-for-nextgeneration-cable-operators-optical-access-networks.