Testbed demonstration of optical reconfiguration by make before break approach in cloud computing networks

By

William Abel Orozco Cifuentes

Thesis

Submitted in partial satisfaction of the requirements for the degree of

Master of Science

in

Electrical and Computer Engineering

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

S.J Ben Yoo, Chair

Chen-Ne Chuah

Matthew Bishop

Committee in Charge

2022

To my family and friends.

Contents

List of Figures

List of Tables

Abstract

Testbed demonstration of optical reconfiguration by make before break approach in cloud computing networks

Datacenter and high-performance computing systems demand for interconnections between computing hosts with low latency, low-power consumption and traffic-aware adaptability. Researchers have demonstrated in recent years that all-optical networks can offer these characteristics by leveraging technologies such as microring resonators, Mach-Zehnder switches and wavelength routers interconnecting top-of-rack electronic packet switches (EPSs). One of the challenges in optical switching is the lack of optical buffers, which makes it difficult to perform optical reconfiguration without significant packet loss. In this thesis, we study the benefit of using a reconfigurable optical network with a make before break (MBB) approach to improve network performance and reduce packet loss when we perform a reconfiguration operation. We compare a plain optical reconfiguration (OST) against a MBB approach in terms of packet loss, latency and throughput as we attempt to achieve hitless reconfiguration. Understanding the transport protocol (TCP) is relevant in our experiments because the path reconfiguration triggers congestion control timers that pause traffic between hosts. Thus, a general analysis of the retransmission timeout (RTO) is presented when studying our performance metrics. Our testbed combines the advantages of electronic and optical networks: buffers, SDN integration, flexible topology and optical links. Our design is modular, hence more servers or switches can be added on demand and configured with software for faster deployments in future experiments. On average, when we tested a single stream of data between two servers, the packet loss decreased from 2.8% in OST to 0.93% in MBB. The link unavailability due to RTO events went from 598 ms down to 121 ms (80% reduction). With MBB, the throughput also improved as it dropped only by 0.6 Gbps instead of a 6Gbps hit in the case of OST.

Acknowledgments

Throughout my Master degree, I always felt supported by UC Davis. There are plenty of resources for students and all the staff in the Electrical and Computer Engineering department assisted me whenever I needed help. I am so grateful to my research group, the Next Generation Networking and Computing Systems lab. In particular, I would like to thank Dr. Roberto Proietti and Dr. Sandeep Singh for their help and guidance during the execution of the project. This thesis would not have been possible without the trust of Professor S.J. Ben Yoo, my major advisor who selected me for the project and provided me with access to the labs, equipment and all the necessary devices to deploy the testbed and run the experiments. I appreciate the time and dedication of the other members of my committee in charge, Professor Chen-ne Chuah and Professor Matthew Bishop, for evaluating my thesis. My parents Abel and Elda, my brother Luis Gerardo and Dr. Ricardo Prado have been a fundamental part of my life. They have been always motivating me to pursue my dreams. And finally, my friends and my family deserve an acknowledgement since they never left me alone, but they supported me and made me feel better during challenging circumstances. Thanks to all who have been part of this, I could not have achieved my Master without you.

Chapter 1

Introduction

1.1 Motivation

In recent years, data centers and high-performance computing systems (HPC) have been adopting heterogeneous memory and processor nodes to allow scientists to run large scale experiments and achieve important advances and discoveries in fields like engineering, science, healthcare and astronomy [2]. As of November 2021, supercomputers are closer to achieve Exascale performance (1 Exaflop per second - 10^{18} floating point operations). The first place is for Fugaku (Fujitsu, Japan) with 442 Pflops/s. In second place, Summit (IBM, United States of America) shows a performance of 148.8 Pflops/s [3]. In the USA, the Exascale Computing Project (ECP) is an effort of the US department of Energy, managed by leaders of different national laboratories [4]. ECP's principal mission is to accelerate the delivery of the first nation's exascale system [5]. The ECP also helps the missions of other agencies like the National Institutes of Health (NIH), National Science Foundation (NSF), National Oceanic and Atmospheric Administration (NOAA), and the National Aeronautics and Space Administration (NASA) [6].

Such large-scale systems require interconnections between thousands of computing nodes. Traditional networks offer fixed links where optical fibers connect electronic packet switches (EPS) with high-speed point-to-point optical links. Standard topologies to interconnect computing nodes include 3D and 4D Torus [7] and fat-trees, [8]. where the network architecture is hierarchical. At the bottom in the edge layer, hosts are connected to Top-of-rack (ToR) switches [9].

It is important to highlight that the network traffic in these systems is typically non-uniform. Spots in the network can be barely or heavily loaded, and throughput bottlenecks may occur. [10]. Hence, the necessity of reconfigurable networks for bandwidth steering arises, to allocate higher bandwidth resources in hotspots on demand [11] [12]. To enable these flexible networks, we combine traditional electronic packet switches with optical switching technologies. In that way, we get the benefits of both fabrics to develop a reconfigurable optical network. The topology can be dynamically modified with an optical switch. Finally, routing and forwarding is managed with flow updates, so we can orchestrate the traffic to go through the newly configured physical links. We do not use traditional routing protocols, but a Software Defined Networking (SDN) control plane that allows more flexibility in the definition of the routing schemes and forwarding rules.

In the following subsections we briefly introduce the related work and enabling technologies relevant to different modules of the testbed that was implemented in this research. Finally, we discuss the purpose and contribution of our investigation.

1.2 Enabling technologies

1.2.1 Optical switching technologies

All-optical switches, such as optical micro electro mechanical systems (MEMS)[13], Mach-Zehnder switches [14], microresonator ring [15] and wavelength routing [16] [17] are alternatives to electronic switches. They offer lower power consumption, scalability, fast reconfiguration, low latency and cost savings with silicon photonics integration [18]. However, optical switching technologies have not been commercially deployed in datacenters and HPC systems due to several challenges, including the lack of optical buffers, the scalability limitation posed by physical layer technology (loss, polarization dependence, etc.), and the design of a scalable control plane that can orchestrate fast reconfiguration operations at large scale (between thousands of racks) to follow the bursty nature of the traffic. Therefore, it is necessary to investigate hybrid switching systems that combine the benefits of traditional electronic devices and optical technologies [19] [20].Our testbed combines a

MEMS optical switch and SDN-enabled electronic packet switches, as described later in Chapter 3.

1.2.2 Centralized network management with SDN

In the Software Defined Networking (SDN) approach, the control plane is decoupled from the hardware. It leaves only the data plane in the chassis, while the controller is software-based and uses Application Programming Interfaces (APIs) to send instructions to the hardware to handle network traffic [21]. A centralized control plane is useful to manage complex networks. Google's Software Defined Wide Area Network (SD WAN), B4, which connects their datacenters globally, is an example of a successful deployment of a commercial Software Defined Network (SDN). B4 maximizes the average bandwidth, and gives complete management across all network. It also enables traffic engineering, which derives in other benefits. One is multipath routing, that leverages resources based on application priority, and assigns bandwidth on demand. Google reports an improvement of $3\times$ in link utilization efficiency compared to previous standard practices (that is, networks driven by conventional routing protocols). On average, links run at 70% of their capacity in the long term [22]. In addition, Google adopted SDN in their datacenters when they realized that traditional decentralized routing protocols (like OSPF and BGP) were not the solution to challenges such as creating routes through a broad, fixed and multipath network. Management in such systems is easier to achieve if the network is modeled as a single device with multiple ports instead of a collection of several individual switches running routing protocols [23].

LIGHTNESS is one example of research project that explored and demonstrated the integration of SDN control plane, virtualization of network resources, and an all-optical dynamic data center network. With a modified version of OpenFlow, a customized controller and the abstraction of the Optical Packet Switch node, which was implemented with an FPGA, they achieved optical packet switching [24]. Furthermore, by assigning priority to flows of different applications, Quality of Service (QoS) can be guaranteed. Real-time monitoring is another feature that was implemented by collecting statistics from the network agents through OpenFlow [25]. From these case studies we can conclude that

SDN is the key to control and to use resources efficiently in complex networks.

1.2.3 TCP congestion control

TCP/IP is the name commonly used to refer to the set of network protocols behind the internet. TCP stands for Transmission Control Protocol and Internet Protocol. This is a practical implementation of the theoretical reference model, OSI (Open Systems Interconnection). Due to the complexity of computer networks, organizing the protocols in independent layers with specific purposes makes it easier to implement, provision and debug communication links all around the world. The OSI reference model has seven layers, while TCP/IP condenses all in fewer layers. [26]. So far we have introduced research projects in technologies for reconfigurable optical networks, with applications in datacenters and High Performance Computing (HPC). Specifically in the physical layer. Now we will cover the impact of link reconfiguration in upper layers. We focus on transport layer, following the TCP congestion control mechanism and the reliable data transfer between hosts when there is an event in the optical paths in the middle. A retransmission timer is used to guarantee data delivery when the receiver does not confirm reception of packets. It is defined as Retransmission Timeout, RTO, in internet standards [27]. We discuss this timer in our results in Chapter 4.

Network traffic in our testbed is generated with iPerf [28], a tool that can be configured with TCP or UDP. The first one is known for reliable communication among hosts, thus we choose that protocol in our experiments. Nevertheless, it was originally designed for WANs, where the data Round-Trip Time (RTT) is in the order of hundreds of millisecond due to large geographic distances between nodes, but 2-3 orders of magnitude higher than RTT in datacenters. Such a large RTO in this context impacts in latency and throughput. Reducing the RTO might be a solution, but the challenge is that most systems do not have high resolution timers [29]. Furthermore, varying the RTO helps to act fast against packet loss, but it yields to spurious retransmissions [30] [31]. Researchers have found that optical reconfigurations raise RTO events. RTT in datacenters is generally in the submillisecond scale, while optical switching can reach tens of milliseconds if using MEMS technologies [32]. TCP waits for a certain time (RTO), then it attempts to send data again. The default RTO varies across operating systems. It is set to 200 ms in Linux, 300ms in Windows [33] [34] [35].

A new trend in transport protocols points towards programmable NICs (Network Interface Controller) and FPGAs, to leverage the optimized latency that state of the art distributed datacenter applications require. The High Precision Congestion Control [36] and NanoTransport [37] projects present transport protocols and congestion control schemes in dedicated programmable hardware. This topic is out of the scope of this thesis, and is left as future work to test novel transport protocol schemes in reconfigurable optical networks.

1.2.4 Make before break and hitless network reconfiguration

An optical reconfiguration operation consists of the following steps: decide when to make the reconfiguration, select a new topology based on specific optimization goals, and finally migrate the traffic to the new path [38] [39]. Service halts arise amid topology migrations, in particular when a new lightpath is provisioned dynamically. That is, when the old link is removed and the new one is configured afterwards. To reduce the outage impact, which depends on the link setup time, a make-before-break (MBB) approach is advised. If the optical channels are placed in the network prior to the reconfiguration, they will be ready to carry traffic instantly. However, this technique involves using additional resources, because more links must be operational simultaneously [40]. Quality-of-Service (QoS) and network performance in general are improved if bandwidth is reserved before the traffic transition, since the length of disturbance is minimized [41] [42] [43].

Hitless reconfiguration in optical networks was defined in 1996 as the reconfiguration process where not even a single ATM cell (payload) is lost. ATM stands for Asynchronous Transfer Mode, a standard in telecommunications for data transfer between user-network or network-network nodes, prior to the wide adoption of IP (Internet Protocol) based networks [44]. Bit Error Rate (BER) and Forward Error Correction (FEC) are useful metrics to monitor the quality of a link in an Optical Transport Network (OTN). Specific thresholds must be specified for minimizing or preventing packet loss. As long as these metrics are within the desired range, the link is considered optimal [45]. Research in hitless optical reconfiguration have demonstrated a switching of less than 1μs, without deteriorating the real-time BER, using fast tunable lasers [46] [47]. Other studies implemented a testbed as well, with commercial transceivers and a channel spacing of a minimum frequency step of 0.5GHz to keep BER in a desired range [48]. By adding SDN elements to manage reconfiguration in an AWGR-based optical network via wavelength tuning, the total switching time reached only millisecond scale, and the packet loss was reduced by 50% [49]. For the rest of this thesis, we consider hitless reconfiguration as the update in the topology that generates 0% packet loss, end-to-end.

1.3 Related work

Reconfigurable optical networks have been studied in recent years, in both simulations and testbeds. On the experimental side, we find that SDN-controlled systems are practical to prevent packet loss and throughput drop due to light path updates [50]. With SDN in hybrid networks, we can perform the reconfiguration in different steps: take the traffic from the ports to be updated, reconfigure the topology with the optical switch, synchronizing the transceivers in the new physical routes, and finally route the traffic through the new paths [51]. In terms of reconfiguration latency, we find systems with delays of different orders of magnitude. In [52], researchers integrated silicon photonics Mach-Zehnder Interferometer (MZI) based optical switches with a Ryu SDN controller and electronic packet switches, along with computing nodes, to demonstrate an end-to-end reconfiguration latency of 204 ms. OSA (Optical Switching Architecture) testbed was introduced in [32], which consisted of ToRs, virtual machines (VMs) as hosts, a MEMS Optical switch, a wavelength selective switch (WSS) and the OSA manager which is the server that orchestrates the system. ProjecTor is another testbed with a similar architecture, with a reconfiguration latency of 12μs. In the nanosecond level we find Sirius [16]. These projects are mainly prototypes that integrate several blocks to build a reconfigurable optical network. Our testbed was built with commercial solutions such as an EdgeCore electronic packet switch, DWDM transceivers, a Polatis MEMS optical switch, along with Linux and a python-based SDN controller (Ryu).

An extensive survey on Software-Defined Optical Networks can be found in [53]. If the reader is interested in the state of the art of optical technologies for datacenters and future trends, I recommend the paper ”Prospects and Challenges of Photonic Switching in Data Centers and Computing Systems” [20].

1.4 Our goal and contribution

The purpose of this thesis project is to implement an SDN computing network testbed and investigate techniques to reduce and possibly eliminate packet loss during optical reconfiguration events. The testbed will be used also in future projects of the NGNCS research group at UC Davis. We list below specific tasks that we performed to achieve our goal.

- To design the network architecture of the control plane, application plane and data plane of our testbed.
- To install the hardware and software tools to build the testbed and granting remote access for flexible management of experiments.
- To implement and test a make before break approach to update optical links.
- To achieve hitless reconfiguration and leveraging multipath routing.

In Chapter 2 we explain the testbed architecture in terms of control and data planes. Chapter 3 covers the hardware and software tools that we used to build each block of our design. Next, in Chapter 4, we compare a plain optical switching scenario (OST) with our make before break approach (MBB). We also show the benefit of our design in terms of network performance: link unavailability, throughput, packet loss and RTT. Finally, in Chapter 5 we summarize the results of our project, the benefits and future work.

Chapter 2

Testbed architecture

Simulators are tools that researchers use to demonstrate models, algorithms, devices and other elements of networking and computing systems to improve their performance for large scale and high performance applications. To validate the simulation studies, it becomes essential to execute testbed experiments(like VINI [54], ORBIT [55] [56], Emulab [57] and reconfigurable optical networks discussed in chapter 1), which is the main focus of this thesis.

We describe the testbed in terms of architecture, infrastructure and software tools. First, we will discuss the general design of the system, the purpose of each block and how they interact with others. Second, we will discuss how the system is implemented with servers, switches, wires, fibers, racks and any other physical device. Third, we will explain the software tools we used to orchestrate our experiments. We cover virtualization, network monitoring and traffic generators that we used in our investigation. In this chapter we start with the architecture of our testbed.

Similar to Emulab [58], we connected physical and virtual hosts to different networks on top of a single infrastructure. The first is the management network, which helps to connect remotely to our testbed elements from anywhere. It is practical as it allows researchers from our team to handle experiments through the UC Davis VPN (Virtual Private Network) and an internet access point without being connected with a wire to the hosts nor restricted to UC Davis LAN (Local Area Network). The second is the control network, which orchestrates our experiments and collect statistics from our computing hosts and network nodes. The control and application planes are encompassed here. The third is the experimental network data plane, a reconfigurable optical network composed of electronic and optical switches with 10 Gbps transceivers and optical fibers connecting our computing nodes and carrying data with rates up to 10 Gbps.

Figure 2.1 illustrates the general architecture of our testbed. The data plane incorporates four electronic packet switches (br1, br2, br3, br4), an optical switch (OST, optical switch tray) and optical fibers that interconnect four computing nodes (vm1, vm2, vm3, vm4). The control plane runs on the controller server, with the purpose of sending instructions to the network to update routes. Our orchestrator server set up applications in the computing nodes and collects statistics to measure performance. The northbound (REST API) and southbound (OpenFlow) interfaces enable the communication from the SDN controller to the control plane and the orchestrator. We configured SNMP (Simple Network Management Protocol) and SCPI (Standard Commands for Programmable Instruments) as well, for coarse-grain statistics and to send instructions to the optical switch. To differentiate between host and guest machines for both servers and switches, we use the terms host or physical when not referring to the virtualized elements.

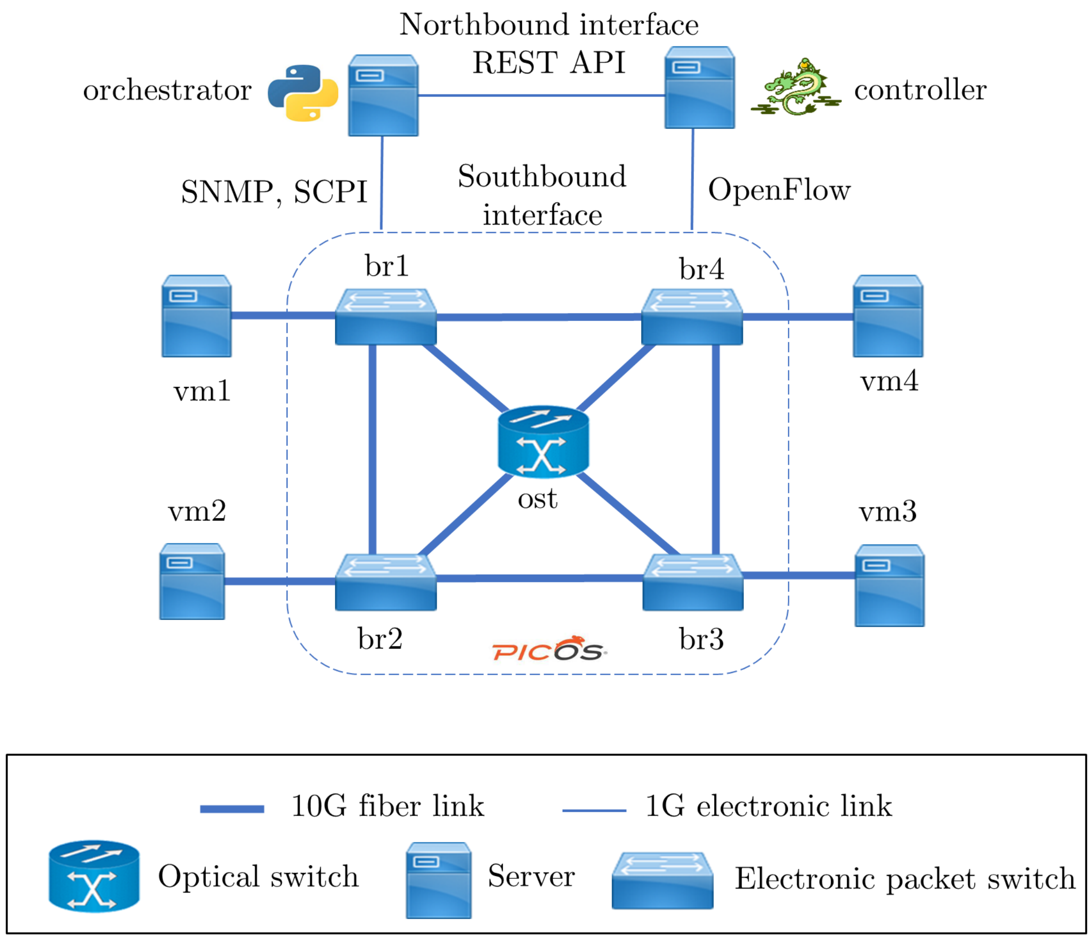


Figure 2.1: Testbed architecture

2.1 Management network

UC Davis network plays an important role to access our servers remotely. Figure 2.2 is a general diagram of our management network. The VPN grants access to our devices via SSH. For security reasons, we are not sharing the public IP addressing. Our computing nodes and electronic packet switches depicted in Figure 2.1 are virtualized in node1, node2 and EPS chassis. Our optical switch is not reachable directly through the VPN, so we do not include it in this diagram.

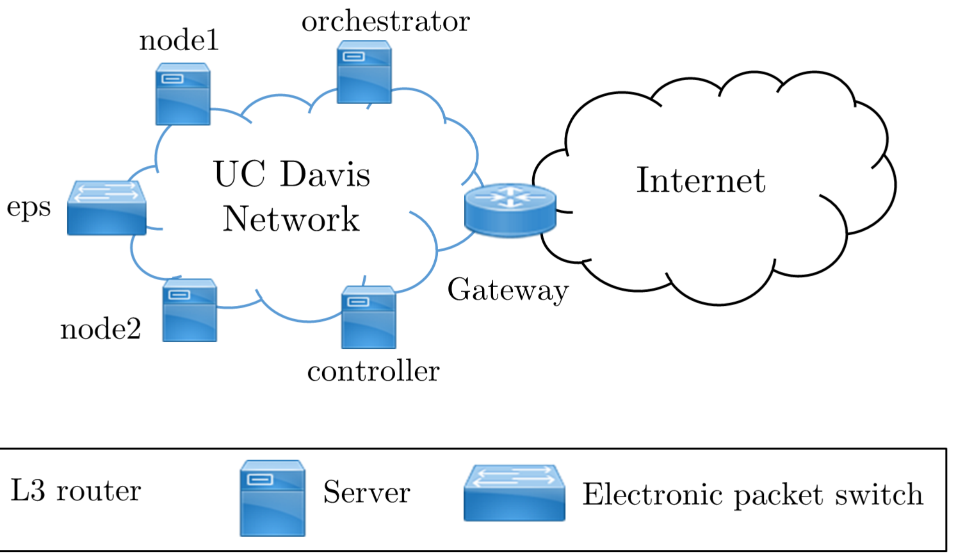


Figure 2.2: Management network

2.2 Control network

The purpose of the control network is to create out-of-band links that facilitate experiment orchestration and collecting data from the testbed, without sending instructions through UC Davis network. In such manner, we have full control over the infrastructure, the network design and we keep dedicated links without worrying for bandwidth availability nor external outages. The individual clouds in Figure 2.3 illustrate different networks, with their own IP addressing. Orchestrator, EPS chassis, controller, node1, and node2 are connected with physical 1 Gbps ethernet links through an electronic switch. The Optical switch (OST) is also connected with a 1 Gbps ethernet cable to the orchestrator. Our computing nodes vm1, vm2, vm3 and vm4 are hosted in node1 and node2, interconnected with virtual links.

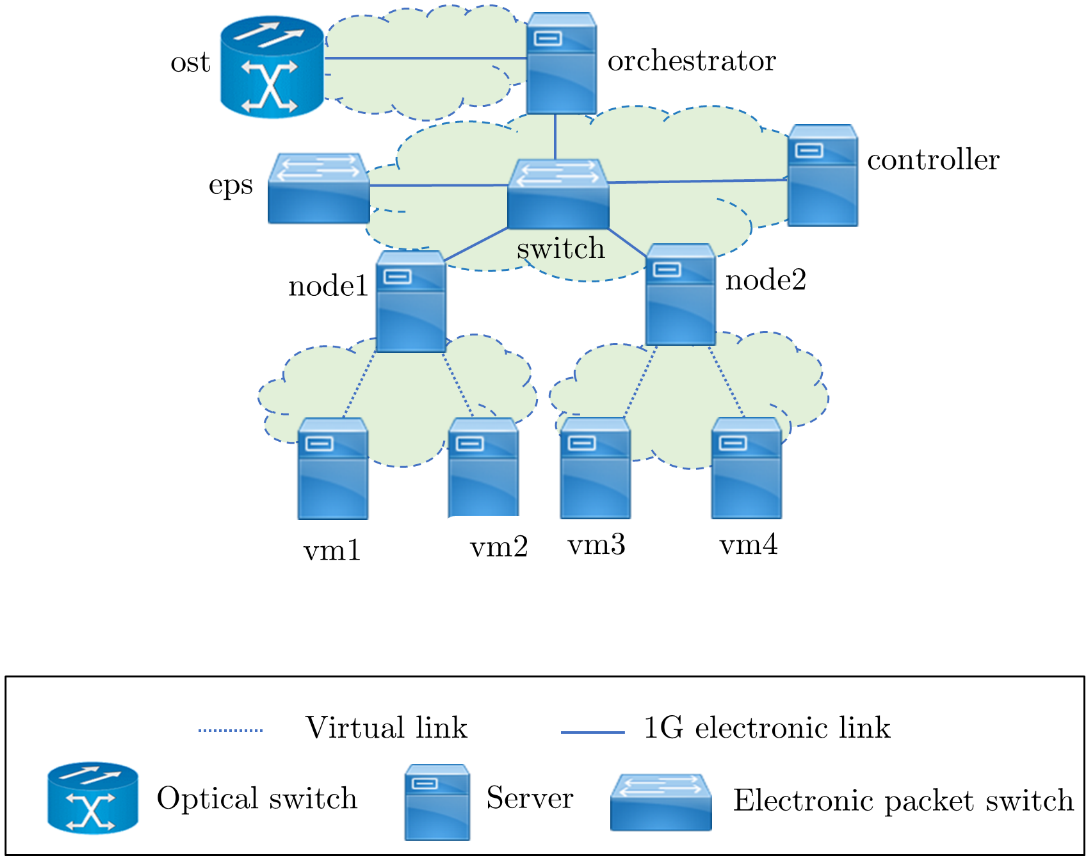


Figure 2.3: Control network

2.3 Data plane

As we mentioned at the beginning of the chapter, our data plane is a reconfigurable optical network driven by an optical switch and four electronic packet switches. Br1, br2, br3 and br4 are hosted in an EPS chassis configured for running in OVS (Open VSwitch) mode, which means that they are optimized for OpenFlow applications and receive instructions from a centralized SDN controller. Each individual link can forward data at rates up to 10 Gbps. Figure 2.4 represent the general topology of our data plane. Nevertheless, the physical links can be attached or removed with software to or from different bridges as a consequence of the flexibility that OVS yields. We use the terms bridge, EPS and ToR as synonyms.

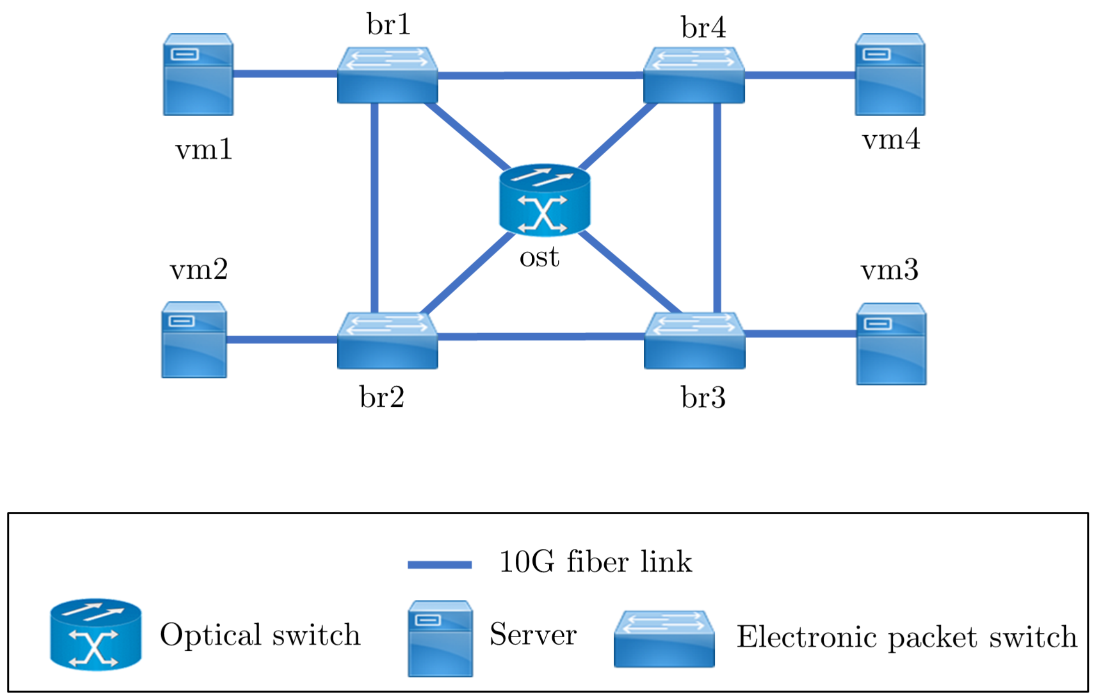


Figure 2.4: Experimental network

2.4 SDN approach

2.4.1 Ryu controller

SDN-enabled devices communicate with a controller through a channel, using OpenFlow protocol [59]. In our testbed, the SDN controller is the built-in application OFCTL

REST from Ryu framework [1] that runs on the controller server. It exposes a REST API (Application Programming Interface) that can be consumed from the orchestrator to send updates or retrieve statistics from the data plane. The URIs (Uniform Resource Identifier) from the API that we employ are *add*, *clear* and *delete strict*. With *add*, flows are installed in the flow table. *Clear* deletes all flows from the specified table. Finally, *delete strict* removes flows that match specific fields like IP address origin, IP address destination and interface.

2.4.2 OVS mode

OpenFlow switches use flow tables to route packets by matching one or more fields. to achieve dynamic routing in a software defined network. The controller can modify flow entries on demand to update the route for each packet [60]. Pica8 PICOS is the Network Operating System (NOS) that runs in our physical EPS. It allows the chassis to run in two modes: Traditional L2/L3 mode (layer 2 / layer 3 from the TCP/IP stack) and OVS. We use the latter, as it fits in our SDN design. In our testbed, we handle the traffic as needed without traditional routing protocols like OSPF. If there is any overlap in the flow tables, the forwarding action will be random and this behavior is not desired. To avoid that, the priority field is varied in our experiments [61]. We use a single flow table for each bridge, and the addressing is done with IPv4. The relevant fields for our testbed are described in Table 2.1.

Field	Description
dpid	Datapath ID (bridge unique identifier)
table_id	Table ID
priority	Priority of the flow
nw_src	IP address, source
nw_dst	IP address, destination
in_port	Input port number
out_port	Output port number

Table 2.1: Relevant fields in the flow table [1]

An alternative to run tests with SDN is to use the L2/L3 mode in PICOS with cross-flow enabled. This hybrid approach allows a combination of both networking paradigms. Network operators interact with the switch with Linux commands (Linux shell) or a CLI (command line interface) similar to Juniper Junos [62] interface (L2/L3 shell) [63]. The flexibility of having two consoles is useful for researchers with different backgrounds. Some may be experienced with Linux servers, others with enterprise or service provider networking. However, throughout our experiments we found issues while performing optical reconfiguration in hybrid operation. We observed that the ports where we connect the transceivers remained in down state after we execute the optical reconfiguration. We had to reboot the Picos service and reinstall the flows to get the ports up, adding at least one minute to the reconfiguration. Hence we decided to run OVS mode.

2.4.3 The concept of flow

So far, we have discussed the pertinence of flows in our testbed and how we use the SDN controller to update routes from the orchestrator using a REST API. Now we will introduce the concept of flow in computer networks. RFCs (Request for Comments) are technical documents published by the IETF (Internet Engineering Task Force) after being written and reviewed by interested parties. They cover foundations for computer networks, namely transport, addressing and routing [64]. From RFCs 2722 and 3697, a flow is defined as the packets sent from a source to a destination that may be unicast, multicast or anycast, with specific attributes (source, destination, port, bytes, addresses, etc.). An alternate definition for flow refers to the packets in a single physical media or stream of data [65] [66]. In the context of our testbed, a flow is an entry in a table (flow table) that is attached to a specific bridge, with different fields that will be matched to follow the desired route towards their destination.

Chapter 3

Testbed infrastructure and software tools

In this chapter we describe the infrastructure of our testbed, hardware and software tools that support our experiments. The hardware is shown in Figure 3.1.

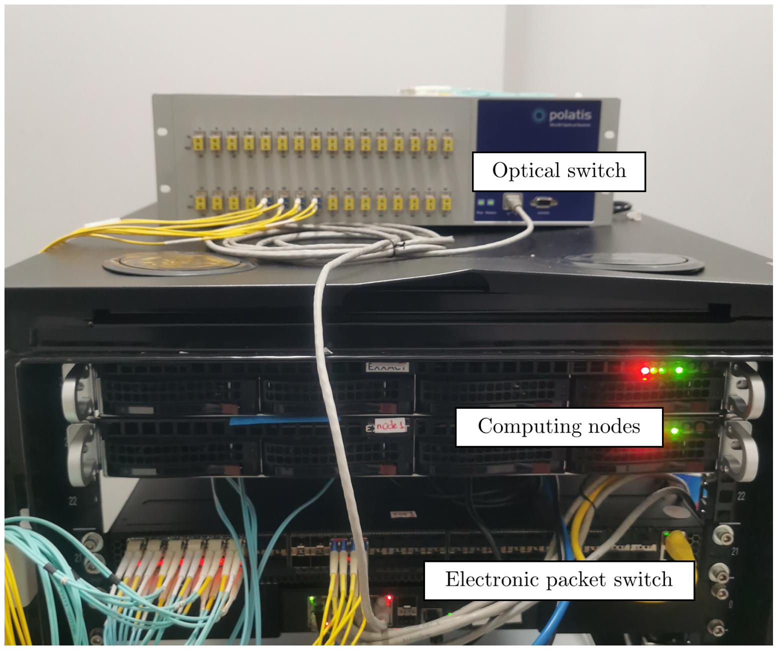


Figure 3.1: Testbed hardware

3.1 Computing servers

Two EXXACT chassis (32 GB RAM, AMD EPYC 7302P 16-Core Processor), node1 and node2, host the virtual machines, vm1, vm2, vm3 and vm4. Each physical server has an Intel X710 dual NIC which supports up to 10 Gbps per port [67]. The main issue with this card was the transceiver compatibility, because it does not read generic devices but only products listed in the Intel compatibility tool [68]. Every virtual server has its own 10 Gbps dedicated port that connects to the data plane, as illustrated in Figure 2.4. Additionally, we connected the virtual machines to the orchestrator with dedicated 1 Gbps NICs apart from the 10 Gbps cards. Figure 3.2 represents the wired and virtual links between the orchestrator, host servers and virtual machines. Virtual bridges inside node1 and node2 map the traffic from the guest servers to the exterior. These interconnections enable the orchestration of applications in our experiments from a principal server. In the rest of the thesis, we use virtual machines, virtual server, computing nodes, computing servers as synonyms.

3.2 Electronic packet switches

Four virtual bridges are configured in the chassis Edgecore AS7312-54XS (tor), which runs the Pica8 operating system in OVS mode optimized for OpenFlow applications. All bridges reach the SDN controller through the physical switch management port, as depicted in Figure 3.3. The bare metal switch has preinstalled the Open Network Install Environment (ONIE) [69], a basic operating system that manages the installation and load of commercial or open source Network Operating Systems (NOS) [70]. If 10-Gbps transceivers are connected in any of the 48 25-Gbps-rated ports of this specific chassis, the speed of the interfaces must be set in groups of 4 consecutive ports. For example, if we require to use port te-1/1/3, all ports te-1/1/1, te-1/1/2, te-1/1/3, te-1/1/4 should have the data rate configured at 10G. We refer to the virtual bridges br1, br2, br3 and br4 as electronic packet switches or bridges.

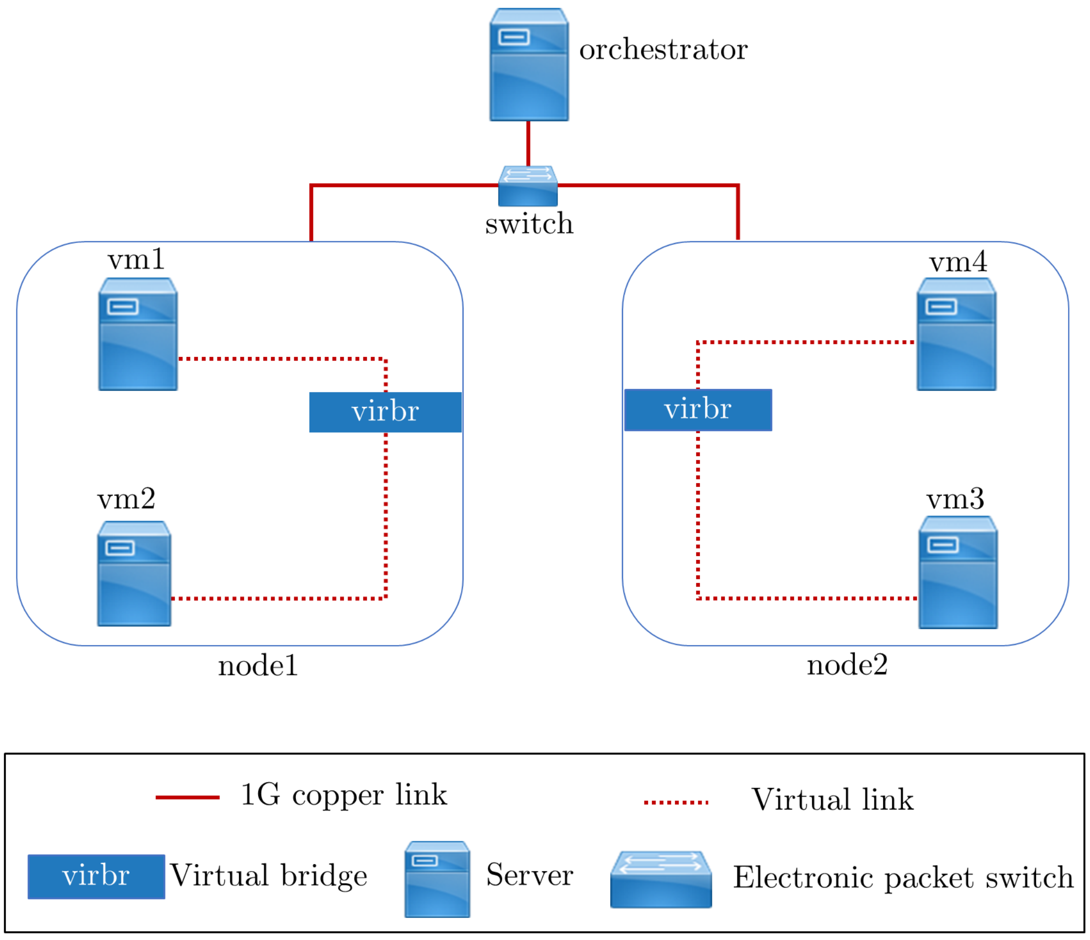


Figure 3.2: Virtual Machines

3.3 Optical switch

Technologies for optical switches were introduced in chapter 1. In our testbed, optical reconfigurations were performed with a Polatis optical switch tray (OST), a single-mode MEMS device that takes up to 25 ms to steer the light beam to the new port [71]. Instructions for switching were sent from the orchestrator with SCPI commands through a TCP socket. The ethernet interface of the switch speeds up the deployment and integration with our testbed control network. The lack of an openflow agent in the OST does not allow to seamlessly integrate the optical switch with an SDN controller. Nevertheless, most recent chassis such as the Polatis 6000 and 7000 series come with openflow agents to enable centralized management with SDN [72]. These products offer flexibility for the

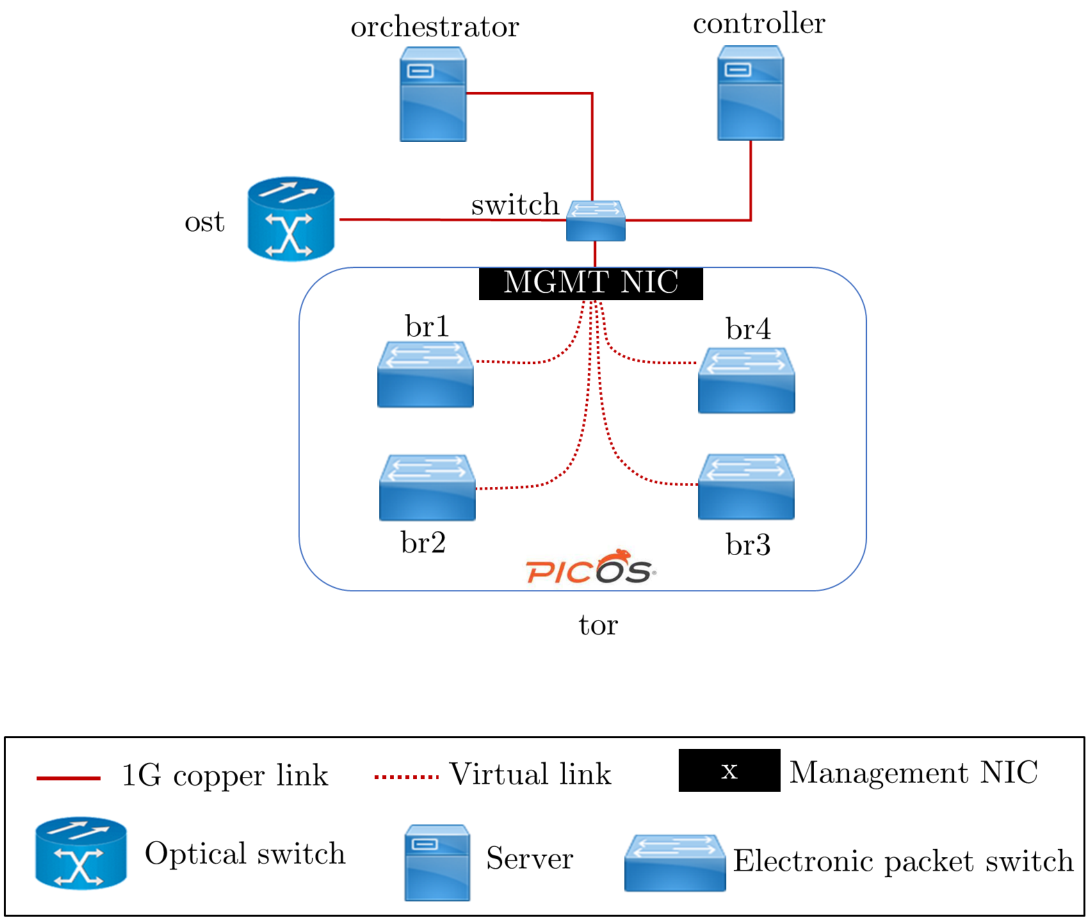


Figure 3.3: Virtual bridges

industry. However, they are built on top of the same MEMS-based technology (Polatis DirectLight [73]) and the switching latency is still in the order of tens of millisecond

3.4 Fibers and transceivers

Small Form-factor Pluggable (SFP) are defined by Cisco as compact optical transceivers [74]. They create an interface between optical and electrical communications, with a transmitter and a receiver in both sides of the link, and support several communication standards like Ethernet, SONET (Synchronous Optical Network) and PON (Passive optical networking). They are manufactured for different purposes: single mode, multimode, for short and long distances, fiber and copper. It is important to choose the appropriate SFP for the application, otherwise the deployments could not work properly or the devices could get damaged. We installed two types of 10 Gbps transceivers. The first works with multi mode fiber (MMF) in the 850 nm band [75], and allow us to connect the computing nodes with the electronic packet switches, as observed in Figure 2.4. The second kind is 10 Gbps DWDM SFP, operating in the C band (four transceivers at 1538, 1540, 1542 and 1543 nm) [76]. By using Single Mode Fiber (SMF) patch cords connecting our electronic packet switches and optical switch, we created a reconfigurable optical network which supports multiple wavelengths.

3.5 Traffic generators

Various tools are available to generate packets between hosts [77]. Some are packeth [78], ostinato [79], D-ITG [80], MGEN [81] iperf [28] and TRex [82]. The latest is developed and maintained by Cisco, and it offers scalability up to 200 Gbps per server. However, this tool is made for a single server, that is, the transmitter and receiver are hosted in the same physical machine, which must have at least two NICs from the supported models [83]. The Device Under Test (DUT) can be an external element or virtualized [84]. Codilime, a company that specializes in software and computer networks, customized TRex to allow testing a network from different start and end points [85] [86]. We tested this traffic generator, but the installation, configuration and integration with our testbed was not straightforward. In contrast, iPerf installation and execution is quick, it does not show issues with drivers and the integration with our experiments was successful. This tool has been widely used in research and generates synthetic packets to emulate traffic between servers[52] [87] [88] [89] [90] [91] [92] [93] [94].

3.6 Packet sampling and network monitoring

Several studies have shown different approaches to gather statistics to analyze network performance. Platforms like Netseer [95], Jetstream [96], Planck [97] have demonstrated improvements in the detection of network performance anomalies, including packet drops, decrease in throughput and increased latency, at different scales like data center [98] and cloud. Zhang *et al.* [99] compare two ways of gathering data from computer networks, fine and coarse sampling. To analyze testbed latencies and network performance metrics, we used tcpdump [100] as we need end-to-end per-packet resolution. On the other hand, SNMP counters work well to confirm that the data stream is going through the desired route when we design the reconfiguration paths. Zabbix [101] is a popular tool in enterprises for monitoring systems, including SNMP statistics, and there are docker versions for agile implementation [102]. With this software we observed how the traffic flows through the EPSs interfaces. To monitor traffic per flow, we deployed an sflow collector and sflow agents in the virtual bridges [103].

Chapter 4

Experiments

Recalling the first chapter, the goals for this research comprehend building a networking and computing testbed with SDN capabilities to demonstrate the benefit of make before break approach combining optical and electronic packet switches to achieve hitless reconfiguration. In Figure 2.1 we show the architecture of the testbed and the meaning of each element in the network diagram. The data plane encompasses an optical switch and four electronic packet switches (br1, br2, br3, br4), connected to an SDN control plane. The topology can be modified as needed with Open vSwitch commands. Additionally, an orchestrator sends the route updates to the controller and optical switch.

In the first subsection we analyze the hardware and software switching latency introduced by our control plane. Next, we compare the throughput, round-trip time (RTT) and packet loss of a single data stream between a pair of servers (sender and receiver). We show the benefit of our make before break approach for updating the route, compared to a plain optical reconfiguration. Finally, similar to the single data stream experiments, we show the metrics of doing a route update with two data streams to demonstrate the benefit of make before break. We refer to the last subsection as bandwidth steering experiments.

4.1 Testbed delays

Five main switching latencies were found in our testbed. A summary is shown in Table 4.1. We discuss each one in the following subsections. Overall, the dominant latency of 605ms is introduced by the transceivers and the operating system of the host electronic packet switch. In later sections, we show how to avoid interrupting the data stream due to a link unavailability caused by a path reconfiguration, by using our MBB approach.

Latency	Source	Value
Orchestrator to controller RTT	Network	0.35ms
EPS per flow insertion delay	Pica8	$30\mu s$
EPS Layer 3 switching delay	Pica8	17.5ms
Optical switch delay	Polatis OST	25ms
SFP locking and EPS polling delay	SFP, Pica8	605ms

Table 4.1: Testbed latencies summary

4.1.1 Optical switching: Transceiver locking, EPS polling and Optical switch delay

We define the optical switching latency as the time it takes from the beginning of the reconfiguration performed by the optical switch, which leads the ports to go down, until all the transceivers report operativity in the ToR chassis. The topology in Figure 4.1 shows a single data stream between servers vm2 and vm3. At the beginning, the data transfer goes through EPS br2, br1, br4, and br3 (continuous orange arrow). Then we perform the path reconfiguration with the optical switch ost1, and the new route (fewer hops) between servers passes through switches br2 and br3 (dotted orange arrow). A summary of steps performed in our experiment is shown in Table 4.2.

Server 2 and server 3 run iperf in server and client mode, respectively. The data rate on the sender side is set at values from 5 Gbps to 10 Gbps, with a duration of 20s. After 485 experiments, we obtained the logs from the physical electronic packet switch, which hosts the virtual switches br1, br2, br3, br4. Then we identify all the interface flapping events of the ports in the topology shown in Figure 4.1. Finally, we calculate the time difference between these events, port down and port up, to obtain the summary of statistics in Table 4.3 and Figure 4.2 as well.

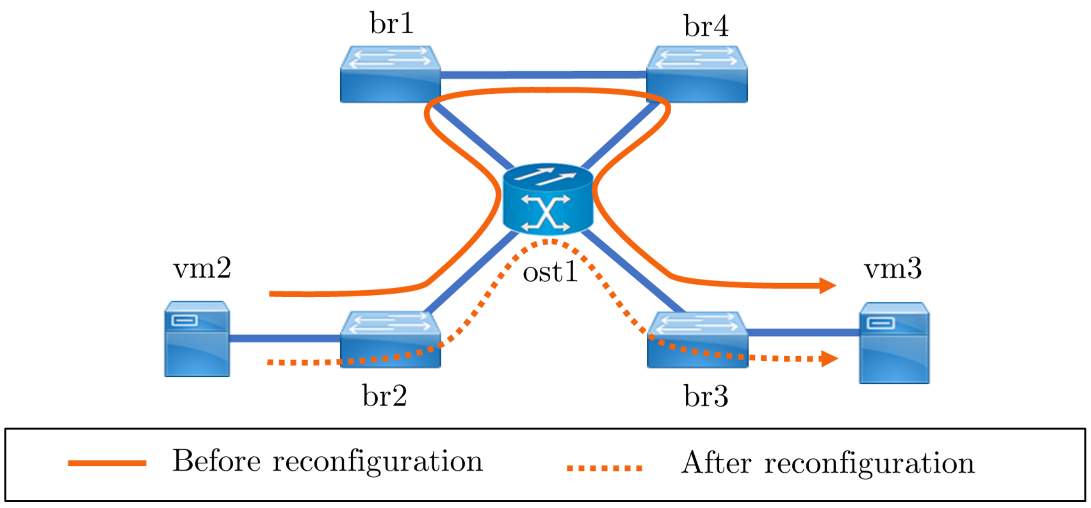


Figure 4.1: Optical reconfiguration - single data stream

Time in seconds	Steps
T=0	1. Set the route between vm2 and vm3 (continuous orange arrow): 1.1. Configure optical paths in ost1 1.2. Install flows in br1, br2, br3, br4 2. Start data transfer
T=10	1. Perform optical reconfiguration in OST (dotted orange arrow): 1.1 Update topology in ost1 1.2 Update flows in br1, br2, br3, br4
T=20	1. Finish data transfer

Table 4.2: Optical reconfiguration steps

The mean latency of our experiments is 0.605s, with a maximum value of 0.834s. This delay has two main components: the operating system polling frequency and the time it takes for transceivers to complete the fault recovery process, before ports come up and data transfer can start or resume. PicOS polls the transceivers every 250ms; furthermore it only supports duplex auto negotiation mode for SFP [104]. On the other hand, the SFP vendor informed that the initialization delay varies across different brands, but it should be consistent with the SFF8431 guidelines for SFP+ DWDM. After a fault is detected in the link, in less than the maximum value of t_start_up (300ms) plus t_reset (10 μs), the optical transmitter reset the laser circuits and disable the tx_fault flag [105, p. 15]. In [52], this delay is defined as transceiver locking and switch polling delay.

Moreover, the Polatis optical switch (ost1 in Figure 4.1) takes 25ms at most to complete the optical reconfiguration, as listed in the technical documentation [71].

Summary	Value [ms]
Mean	605
Std	115
Min	529
25%	531
50%	532
75%	786
Max	834

Table 4.3: SFP delay statistics of 485 experiments

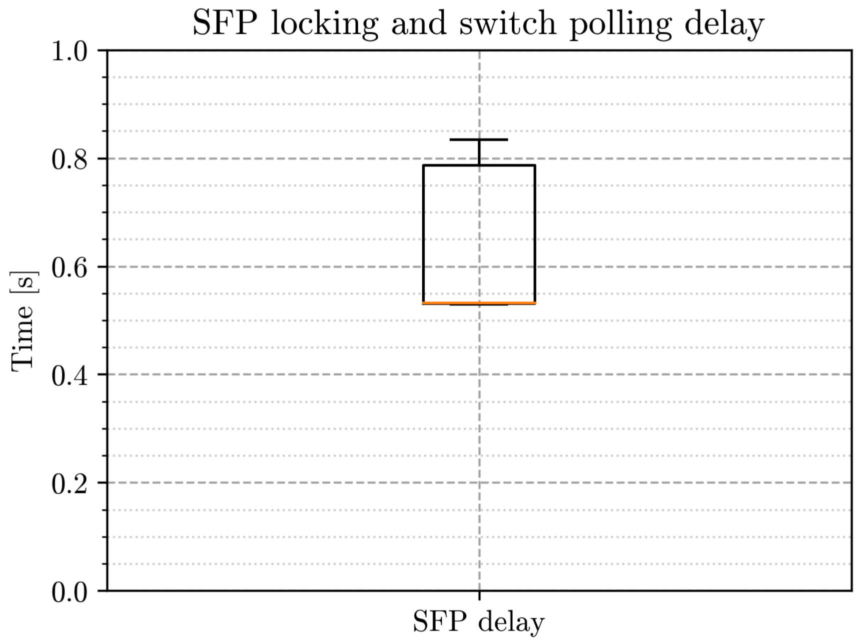


Figure 4.2: SFP delay boxplot

4.1.2 EPS per flow insertion delay

PicOS allows adding flows as bundles, with the *ovs-ofctl bundle* command [106]. We sent 100 flows in parallel, the maximum amount per bundle, to a flow table of an EPS, and it took 3ms in total. We take the ratio of total time divided by number of flows, go calculate that every flow is inserted on a single table after $30\mu s$.

4.1.3 Orchestrator to controller RTT

Other latency in the flow update process is the RTT between our orchestrator and controller servers. The orchestrator sends the path updates through the REST API. The controller runs the OFCTL REST application, which receives the new routes and communicates with the bridges to update the flow tables (See Figure 2.1). We sent several flow update requests to the controller, and the average RTT is 0.35 ms (see Figure 4.3).

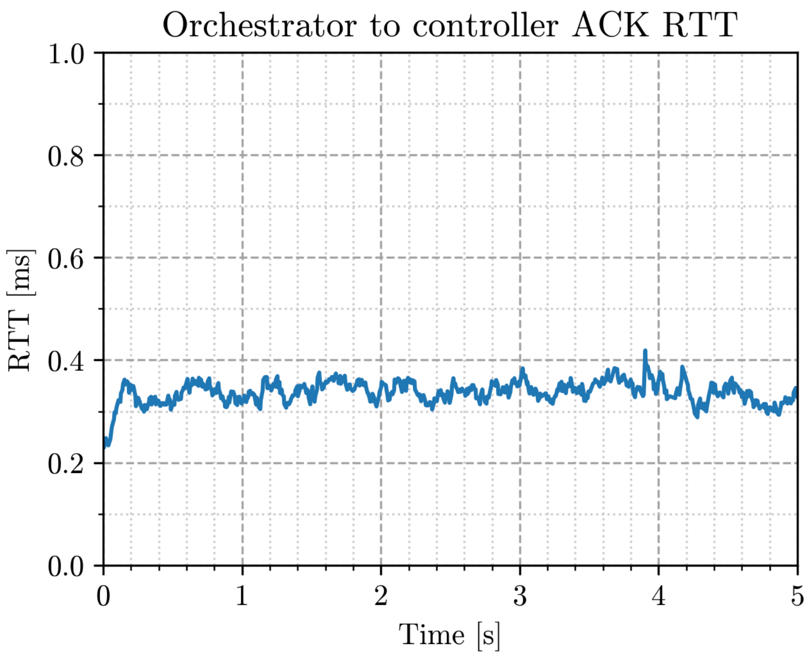


Figure 4.3: Orchestrator to controller RTT

4.1.4 EPS Layer 3 switching delay

After the flows are inserted in the EPS tables and the reconfiguration is completed, traffic takes additional time to start flowing through the new route. We call it layer 3 switching delay of the EPS. Previous research has found a delay of 49.2 ms [52]. To determine the parameter in our testbed, we ran 50 experiments of UDP data transfers between servers vm2 and vm3, switching the route with flow table updates. In other words, TCP congestion control and the optical switching delay were not included in these tests.

Figure 4.4 describes the initial (continuous orange arrow) and final (dotted orange arrow) routes for the data stream. Table 4.4 lists the steps of the experiments. The packet capture was executed in the receiver side to determine for how long the packets are dropped due to changes in the flow tables of all the EPS. In steady state, no packets are lost.

On average, the EPS in our testbed takes 17.5 ms to update the routes after the new paths are stored in the flow tables. See Figure 4.5 and Table 4.5

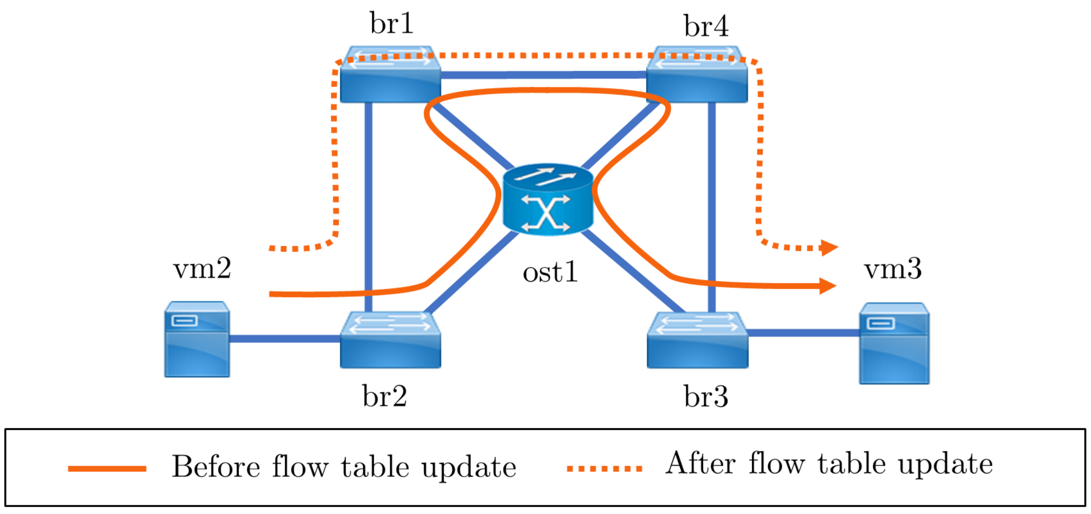


Figure 4.4: Path reconfiguration with flow table updates

Time in seconds	Steps
T=0	1. Set the route between vm2 and vm3 (continuous orange arrow): 1.1. Configure optical paths in ost1 1.2. Install flows in br1, br2, br3, br4 2. Start UDP data transfer
T=10	1. Perform reconfiguration (dotted orange arrow): 1.1 Update flows in br1, br2, br3, br4
T=20	1. Finish data transfer

Table 4.4: Steps for path reconfiguration with flow updates

Summary	Value [ms]
Mean	17.5
Std	10.5
Min	10.6
25%	11.0
50%	12.7
75%	21.9
Max	78.0

Table 4.5: EPS L3 switching delay statistics of 50 experiments

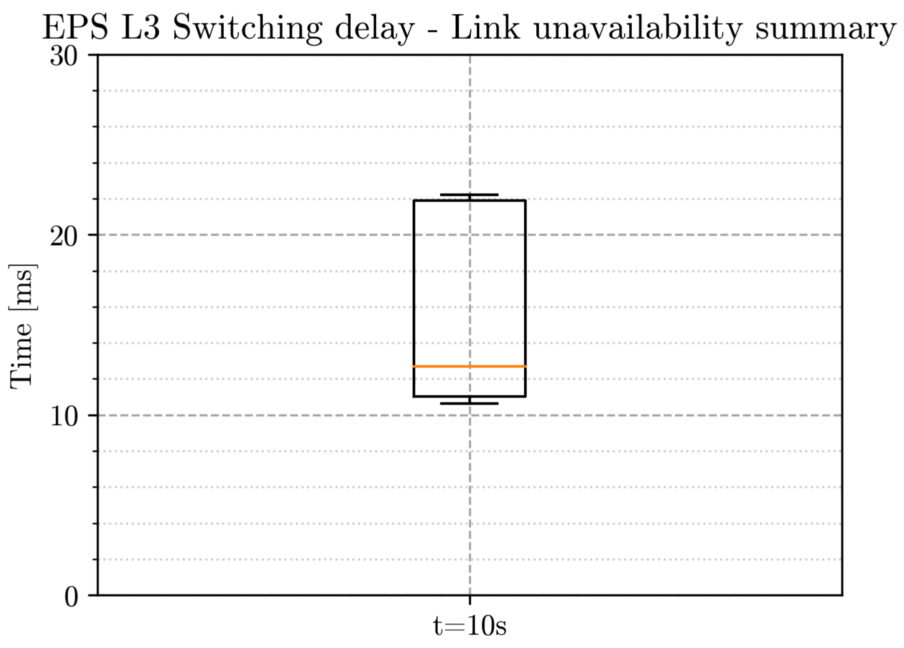


Figure 4.5: EPS L3 switching delay boxplot

4.2 Optical switching and make before break

In this last section, we discussed the different latencies introduced by our control plane, hardware and software, when we perform a route update. Table 4.1 helps as reference for this section, as we are going to present the results of several experiments based on optical switching and the make before break approach, with a single data stream between two servers. In this subsection we are going to compare throughput, packet loss and RTT between a plain optical reconfiguration, and the make before break approach. Furthermore, we are going to show first the effect the TCP retransmission timeout (RTO) on the link unavailability as seen by the end hosts when running the route update, and the throughput at different data rates to validate how the TCP congestion control handles buffered data after the new route is ready.

In Table 4.6 there is a summary of metrics that we use in this chapter, to compare the performance of two path reconfiguration approaches.

Metric	Definition
Throughput	Ratio between TCP payload and packet departure rate
RTT	Round-trip time for the packets the stream
Packet loss	Ratio between packets retransmitted and packets sent
Link unavailability	Time gap during the path reconfiguration where packets were not observed by the sniffer

Table 4.6: Definition of metrics

The data was captured with the sniffer *tcpdump* on the sender side. For all TCP data streams, we obtained the following fields for each packet: TCP payload, RTT, retransmission, and packet departure rate (Δt). All four values are calculated by tshark [107], being retransmission a boolean value which is true if the packet was detected as a retransmission, false otherwise. Packet departure rate (Δt) is obtained by taking the difference between timestamps of all packets.

4.2.1 Data rate sweep

In this section, the experiments were performed with the topology in Figure 4.1, only one stream of data between servers 2 and 3 over TCP using iperf, following similar steps to those in Table 4.2. However, the path reconfiguration was performed after 30 seconds, and the experiment was run for 1 minute instead.

Figure 4.6 shows the throughput of the stream at different rates. When the link is unavailable due to the optical reconfiguration, the sender is unable to continue with the data transfer, thus packets are buffered. Once the link becomes available again, the traffic resumes but the rate reaches the maximum link capacity. That is, near 10Gbps. From the plot we see that is actually 9.4 Gbps. Some streams restore 400 ms after the optical reconfiguration (9Gbps stream), others take up to 800 ms. Throughput is almost steady before path reconfiguration. After the route is modified with the optical switch and the flows are updated in the EPS, we see the hit in throughput, which decreases to almost 1 Gbps. Then it raises, and stabilizes at the original data rate, just after the sender clears the buffer by saturating the link for a few seconds. A factor of 200 ms is implicit in the link unavailability. It is the minimum retransmission timeout, a TCP timer that we discuss in the next subsection.

RTT is also affected by the optical reconfiguration. As shown in Figure 4.7, at T=30 seconds, RTT grows creating a discontinuity in the plots before returning again to the original value. However, before returning to its original value, RTT shows a sawtooth shape when the sender saturates the link to clear its buffer. This is related to TCP congestion control, and we observe this behavior clearly in Figure 4.7f, where data is sent at a fixed rate of 10Gbps, the maximum link capacity, causing link congestion during all the experiment. When the link is not saturated, the average RTT remains between 0.4 and 0.6 ms, but when link is at maximum capacity, RTT oscillates around 2 ms.

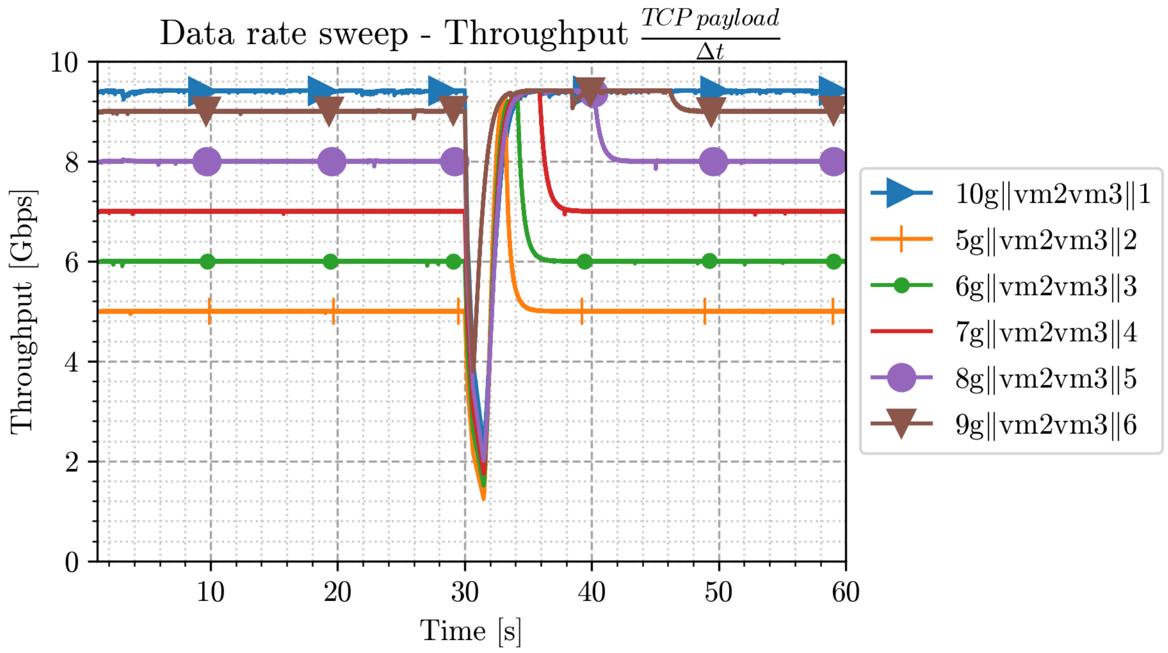
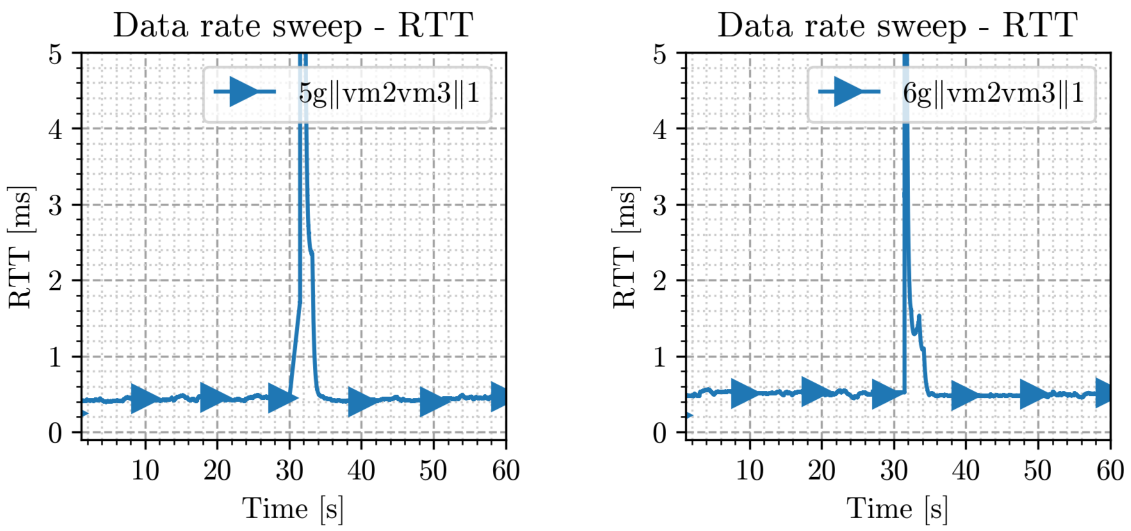
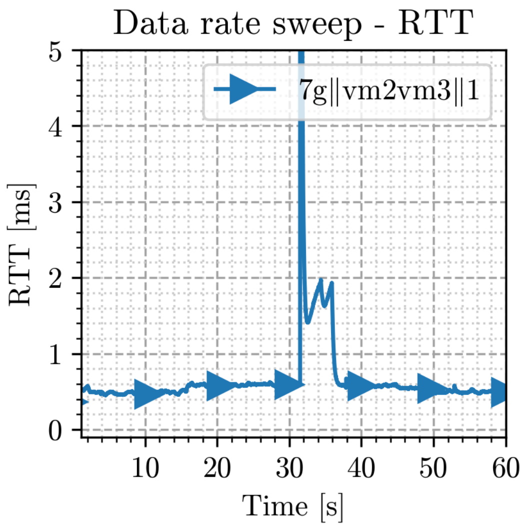


Figure 4.6: Throughput, optical reconfiguration at different data transfer rates

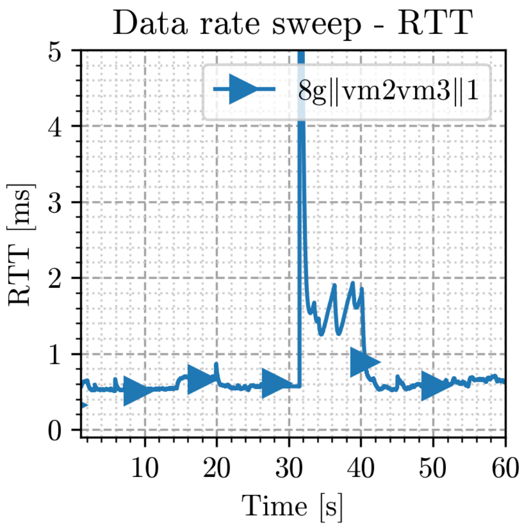


(a) 5 Gbps

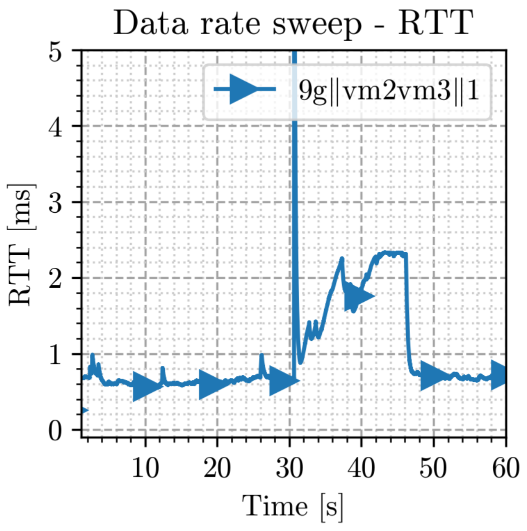
(b) 6 Gbps



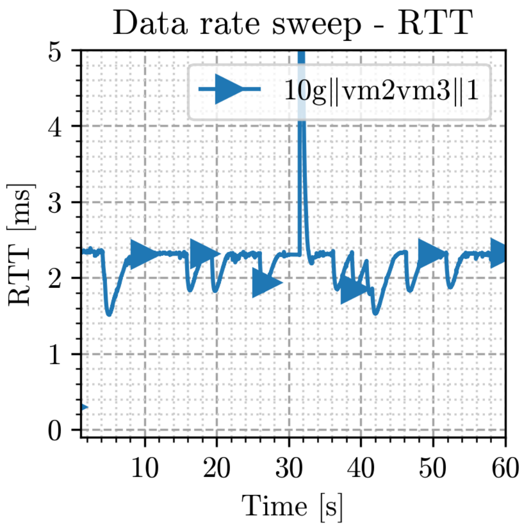
(c) 7 Gbps



(d) 8 Gbps



(e) 9 Gbps



(f) 10 Gbps

Figure 4.7: RTT comparison at different data rates

4.2.2 RTO sweep

Similar to subsection 4.2.1, the experiments were performed with the topology in Figure 4.1, only one stream of data between servers 2 and 3 over TCP using iperf, following steps in Table 4.2. This time we did use a duration of 20 seconds and an optical reconfiguration after 10 seconds. We discuss below the parameters we varied in the TCP session. The goal of sweeping the RTO is to show how the timer is triggered and increased when we perform an optical reconfiguration, at different RTO values. Throughput decreases as the link unavailability dictated by the RTO grows.

In subsection 1.2.3 of chapter 1, we introduced the TCP retransmission timeout. This timer has a minimum value, which is a parameter that varies across implementations of operating systems. Tuning RTO_{min} can lead to unexpected behavior in the way our servers communicate with the rest of the network, in both control and data planes. It enhances the speed response against packet loss, but can derive in spurious retransmissions [31]. Nevertheless, it is possible to vary RTO_{min} per route [108], with the advantage of leaving all other networks without modifications. We found that the TCP session between servers had a default RTO_{min} of 204 milliseconds, and it doubles every time the congestion control mechanism detects a link outage. The default congestion control algorithm in our servers is cubic [109].

Table 4.3 shows an average of 605 ms and a maximum of 834 ms in the dominant latency of our testbed, introduced by the transceivers locking and switch polling delay. Our make before break (MBB) strategy considers this time to avoid the link unavailability seen by the servers 2 and 3. It consists of taking the traffic out of the optical switch to alternate links, then execute the optical reconfiguration, and finally, turn back the traffic to the new path. We show more details in next subsection.

Figure 4.8 shows five experiments where we set $RTO_{min} = 800ms$, and the optical reconfiguration was performed at T=10 seconds. In three of them, traffic raises at $10 + RTO_{min} = 10.8s$. Two streams raise at $10 + RTO_{min} + 2 * RTO_{min} = 12.4s$. In other words, the timer expired once and doubled. This is consistent with our results in Table 4.3. In some experiments, switching time took more than 800 ms, others took less than 800ms.

From these results, if we want the timer to be triggered once only, we should set the RTO to be larger than 800 milliseconds. We tested with 900 ms and 2s.

Figure 4.9 shows ten experiments with $RTO_{min} = 900ms$, and the optical reconfiguration at T=10 seconds. All streams show a throughput raise at $10 + RTO_{min} = 10.9s$. RTO_{min} expired once only, therefore the overall switching time took less than 900ms. Figure 4.10 shows five experiments, with $RTO_{min} = 2s$, and the optical reconfiguration at T=10 seconds. Again, only one RTO_{min} expired and the data transfer continued. In conclusion, we can vary the RTO to be triggered once only if we set it to be larger than 900 milliseconds. However, the larger it is, the longer the link remains unavailable and the larger the throughput decreases. We decided to run our next experiments with the default RTO timer for Linux, 200 ms.

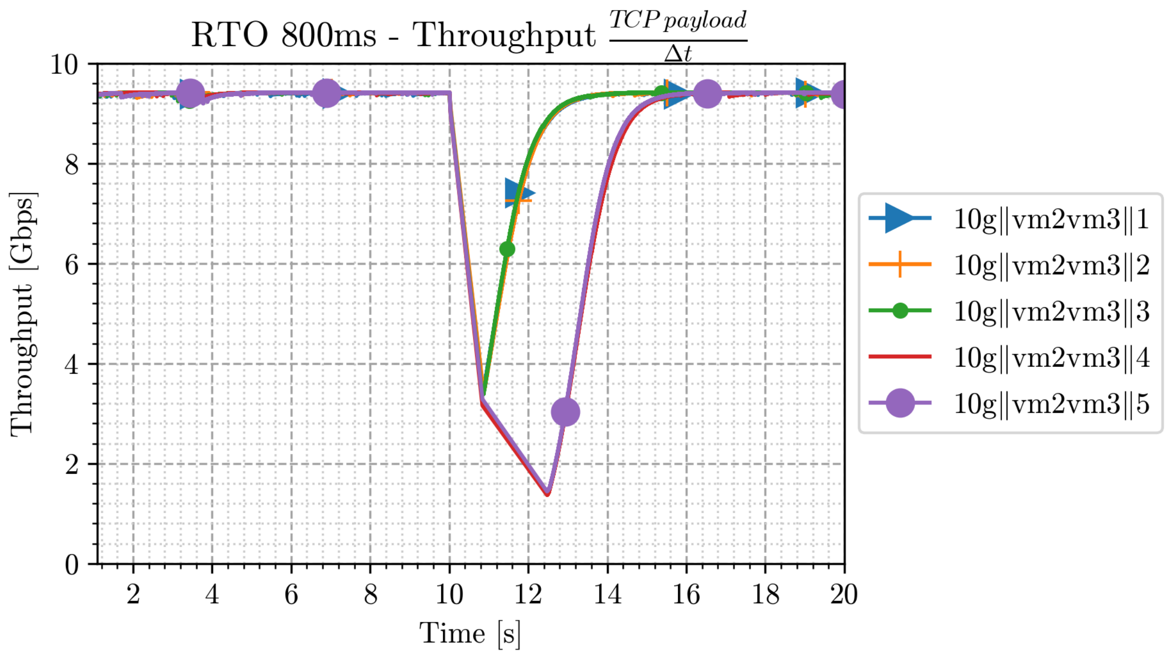


Figure 4.8: Throughput during optical reconfiguration at 10 Gbps traffic rate, $RTO_{min} = 800ms$

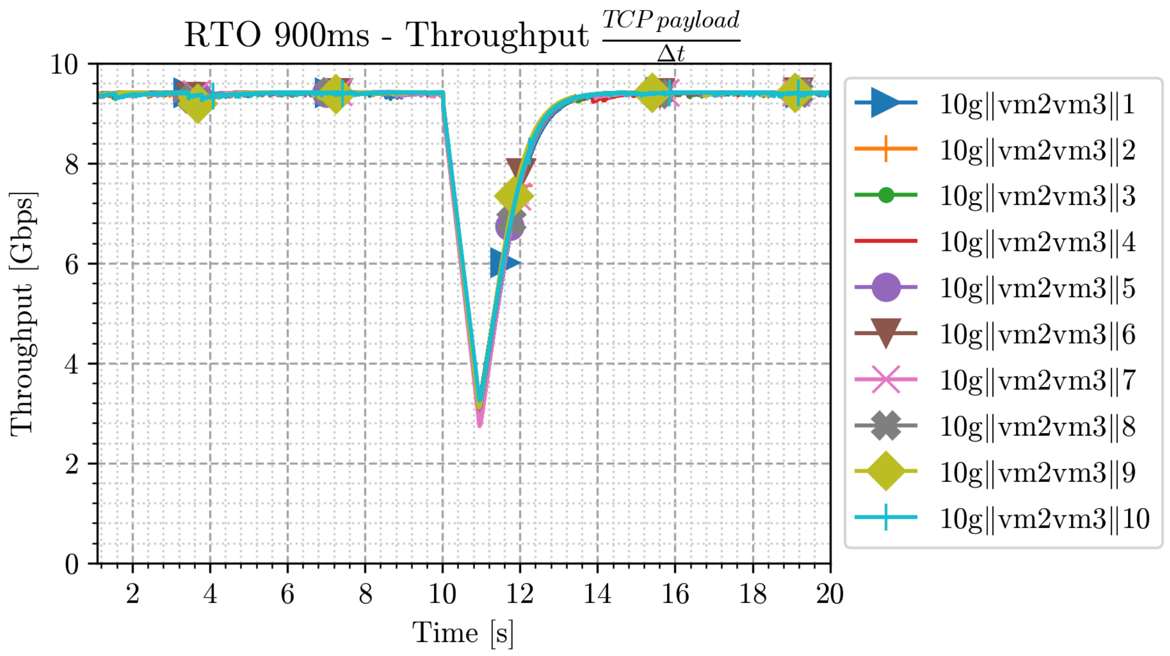


Figure 4.9: Throughput during optical reconfiguration at 10 Gbps traffic rate, $RTO_{min} = 900ms$

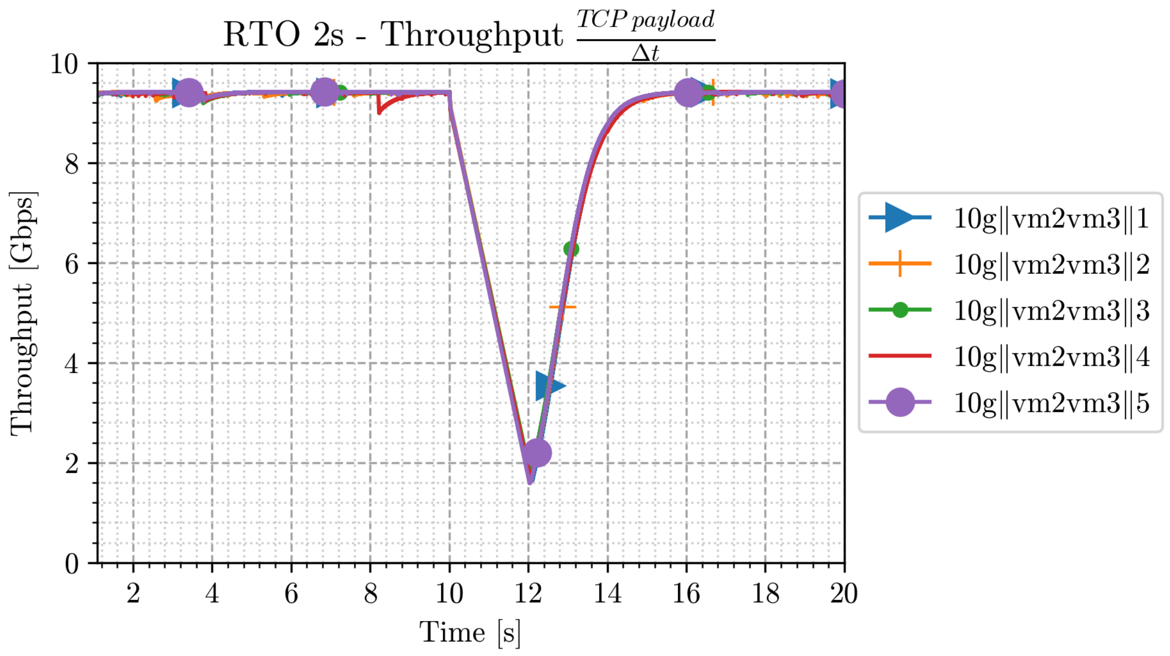


Figure 4.10: Throughput during optical reconfiguration at 10 Gbps traffic rate, $RTO_{min} = 2s$

4.2.3 Make before break approach

Last subsection is helpful to understand why a TCP stream becomes unavailable for a certain period of time when updating the route between endpoints. The testbed latencies in Table 4.1 and RTO_{min} play a main role. The goal of our make before break approach is to achieve hitless reconfiguration when performing an update in the topology and change the route of the traffic between servers. We take the traffic out of the optical switch for 1 second, to allow enough time for all the testbed latencies, before returning the packets back to the new path through the optical switch.

The scenario in Figure 4.11 shows an initial route between servers vm2 and vm3 through optical switch ost1, and four electronic packet switches br2, br1, br4 and br3. Our experiment update the route to a better one, with fewer hops, through EPSs br2 and br3. However, to avoid the reconfiguration time of the testbed introduced mainly by the switch polling and transceiver locking delays, traffic is be switched for 1 second to a temporary route (dotted black arrow) while the optical reconfiguration is performed, between T=10 and T=11 seconds. Finally, we move back the traffic flow to the new rote (dotted orange arrow). A summary of steps is in Table 4.7. One drawback of this approach is that we need two more physical links compared to the plain optical switching approach we introduced in Figure 4.1 and Table 4.2. We show the benefit in terms of link unavailability, packet loss and throughput in the next subsections.

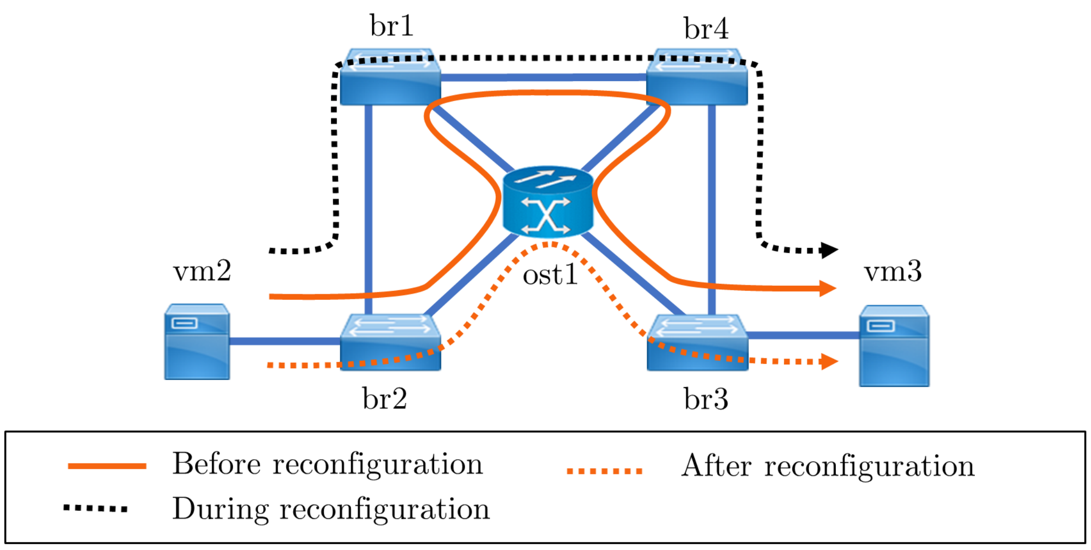


Figure 4.11: Make before break - single data stream

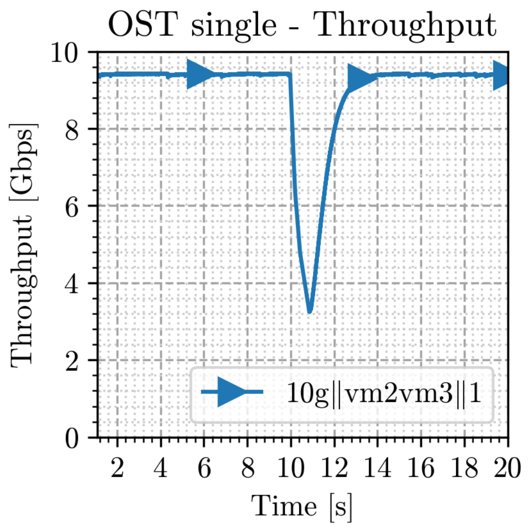
Time in seconds	Steps
T=0	1. Set the route between vm2 and vm3 (continuous orange arrow): 1.1. Configure optical paths in ost1 1.2. Install flows in br1, br2, br3, br4 2. Start data transfer
T=10	1. Take the traffic out of the optical switch (dotted black arrow): 1.1 Update the flows in br1,br2,br3,br4 2. Perform optical reconfiguration in OST (dotted orange arrow): 2.1 Update topology in ost1
T=11	1. Move traffic back to the optical switch (dotted orange arrow) 1.1 Update the flows in br1,br2,br3,br4
T=20	1. Finish data transfer

Table 4.7: Optical reconfiguration steps

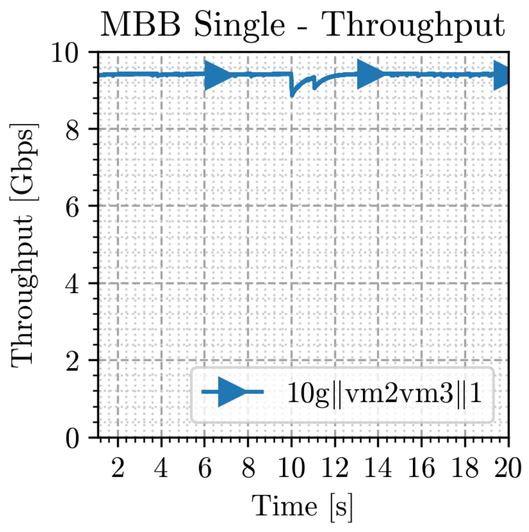
4.2.4 Comparison between Make Before Break (MBB) and Optical switching (OST)

In subsection 4.2.3 we introduced our Make before break strategy, which consists of reconfiguring the network topology with the optical switch, and taking the traffic out to alternate links to avoid the link unavailability during reconfiguration. Fifty experiments for each approach, plain optical switching (OST) and make before break approach (MBB) with a single data stream between servers vm2 and vm3, were performed. Steps for OST and the network diagram are in Table 4.2 and Figure 4.1. For MBB, they are in Table 4.7 and Figure 4.11. Data was generated with iPerf configured at a fixed data rate on the sender side, 10Gbps. Again, the goal was to route the traffic through a better path, with fewer hops.

Figures 4.12 and 4.13 depict a representative experiment of each approach, in terms of throughput and RTT. Throughput improves in MBB, because it drops roughly by 0.6 Gbps, not by 6Gbps as it does in OST. RTT also benefits from MBB approach, because it does not grow up to hundreds of milliseconds due to link unavailability, but only suffers from a smaller hit of 0.2ms. Both plots in Figure 4.15 helps to understand the behavior described before. In 50 experiments, the link unavailability in terms of the default RTO_{min}, 200ms for linux, is between two and three times in OST. It means that the timer expired at most twice. This is consistent with our testbed latencies, where the average value of the dominant time element (switch polling and transceiver locking) was 605ms, as listed in 4.3. On the other hand, the RTO_{min} timer expired at most once in the MBB experiments. 75% of the link unavailability is below 200ms, one RTO_{min}, and one half of the MBB experiments is below 20ms, which is in the order of magnitude of the EPS L3 switching delay that we demonstrated in Table 4.5. Finally, packet loss decreases in MBB compared to OST. In Figure 4.14 we observe that packet loss goes down from an average of 2% and a maximum of 9.5% in OST, to an average of 0.9% and a maximum of 3.4% in MBB. There is some packet loss of 0.01% in steady state, associated to scarce retransmissions due to link congestion.

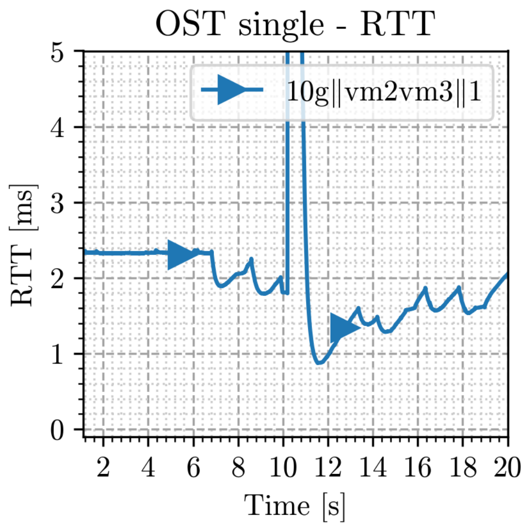


(a) OST, single stream

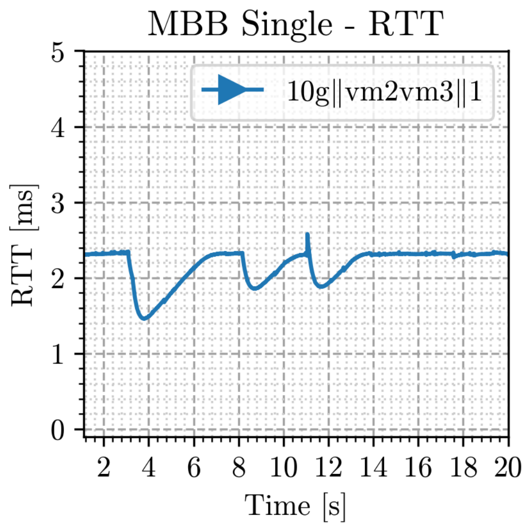


(b) MBB, single stream

Figure 4.12: OST and MBB, throughput comparison

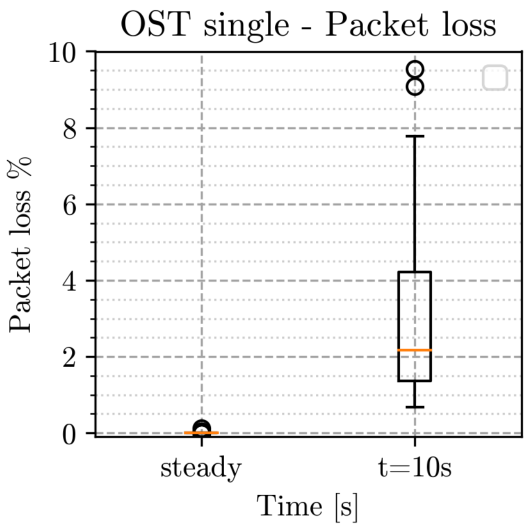


(a) OST, single stream

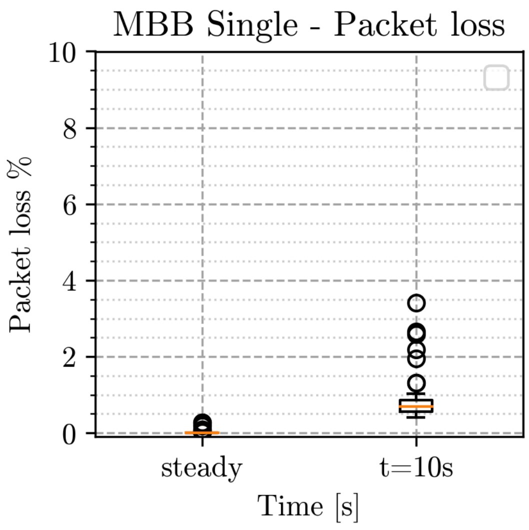


(b) MBB, single stream

Figure 4.13: OST and MBB, RTT comparison

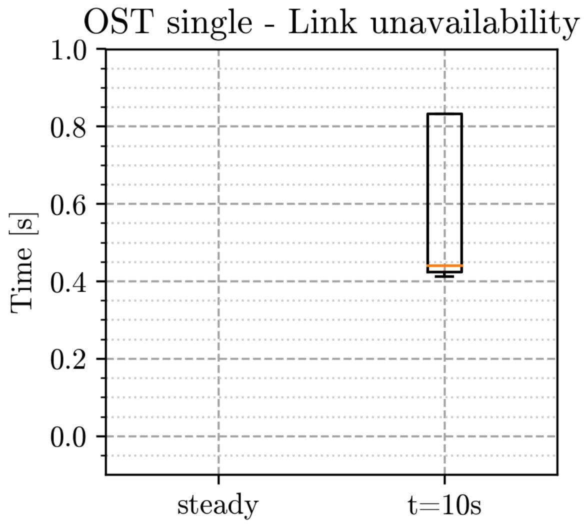


(a) OST, single stream

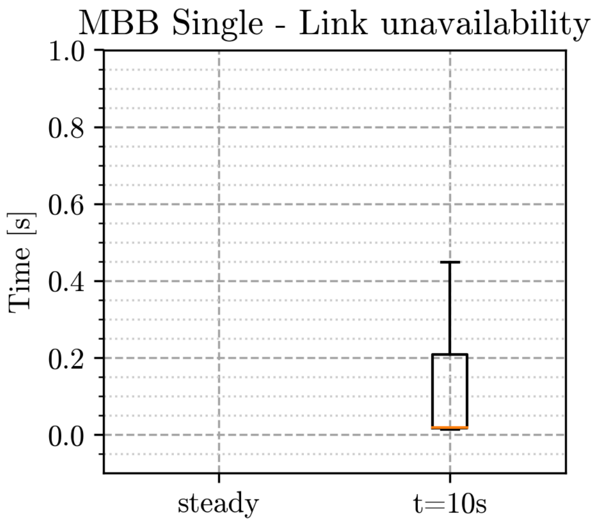


(b) MBB, single stream

Figure 4.14: OST and MBB, packet loss comparison



(a) OST



(b) MBB

Figure 4.15: OST and MBB, link unavailability comparison

4.3 Bandwidth Steering experiments

Data buffering was observed in Figures 4.6 and 4.7, as link utilization at 9.4 Gbps and RTT increased while link was saturated. In the following experiments we configured iPerf with automatic data rate adjustment depending on the link utilization, not at a fixed rate. If there are 10 Gbps of bandwidth available in the route, then sender transmits at 10 Gbps. if there are only 5Gbps available, then sender transmits at 5Gbps. In this way, we reduced buffering on the transmitter side. Our goal with this configuration was to avoid packet loss due to link congestion, leaving only packet loss due to either optical switching or make before break.

We did 50 experiments of each MBB and OST reconfiguration approaches. This time with two data streams going through the testbed. We followed steps in Tables 4.2 and 4.7, but now we add a static flow of data between servers 1 and 4 through EPS br1 and br4, as shown in Figures 4.16 and 4.17. Thus we create link congestion on the link between EPS br1 and br4, because both senders try to adjust the data rate to the maximum possible, but as it is a shared link (bottleneck on purpose), they send at 5Gbps each to achieve the maximum link capacity of 10Gbps with both rates combined. As a result, we notice two data streams at 5Gbps before T=10 seconds in Figure 4.18.

One benefit of reconfigurable optical networks is that we can allocate bandwidth resources on demand. We allow larger data rates by separating flows and removing bottlenecks associated to congested links. Our scenario changes the network topology to assign a better path between servers vm2 and vm3, increasing the available bandwidth to both data streams.

We measured performance metrics of throughput, packet loss, RTT and link unavailability, and our MBB approach shows benefits in all of them. Figures 4.18 and 4.19 include a representative experiment of each approach. Throughput of the dynamic route between vm2 and vm3 falls only 2Gbps in the MBB approach, compared to a hit of almost 4 Gbps in the OST experiment. Moreover, RTT of the dynamic data stream grows to the order of hundreds of milliseconds in OST, compared to MBB where we barely see a hit of 0.2ms. For the fixed data stream between vm1 and vm4 we did not observe a difference in the metrics before, during and after reconfiguration, compared to what we observed in the dynamic data stream between servers vm2 and vm3. Considering this, for the packet loss and link unavailability metrics of Figures 4.20 and 4.21 we only evaluated the stream between vm2 and vm3 of each test. Packet loss decreased from an average of 1.28% in OST to 0.44% in MBB. Similar to our results in previous subsection,there is packet loss of 0.01% in steady state, associated to scarce retransmissions due to link congestion, almost negligible.

An interesting pattern was found in the link unavailability. OST strategy had between two and four times RTO_{min}. That is, the timer expired between two and three times. However, in MBB approach we noticed that the mean value of link unavailability is one RTO_{min} with some outliers above and below. In other words, the link is unavailable for an almost fixed value of 200ms with two data streams. Basically, this metric also improves from OST to MBB when working with multiple data streams in the testbed. It will be of interest to determine if faster optical switching technologies, in the submillisecond latency level, trigger any RTO events in the TCP congestion control mechanism.

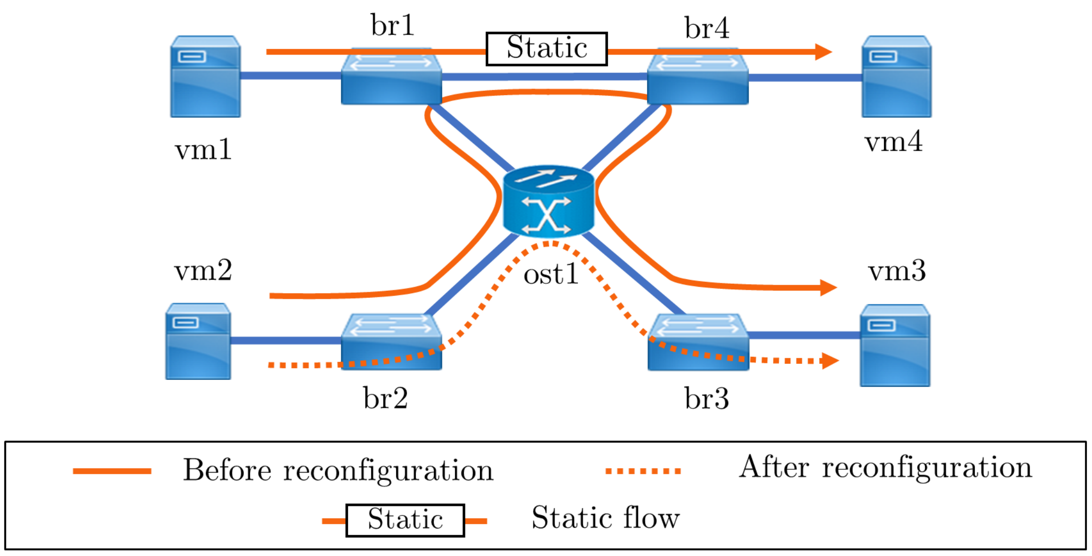


Figure 4.16: Optical reconfiguration - dual data stream

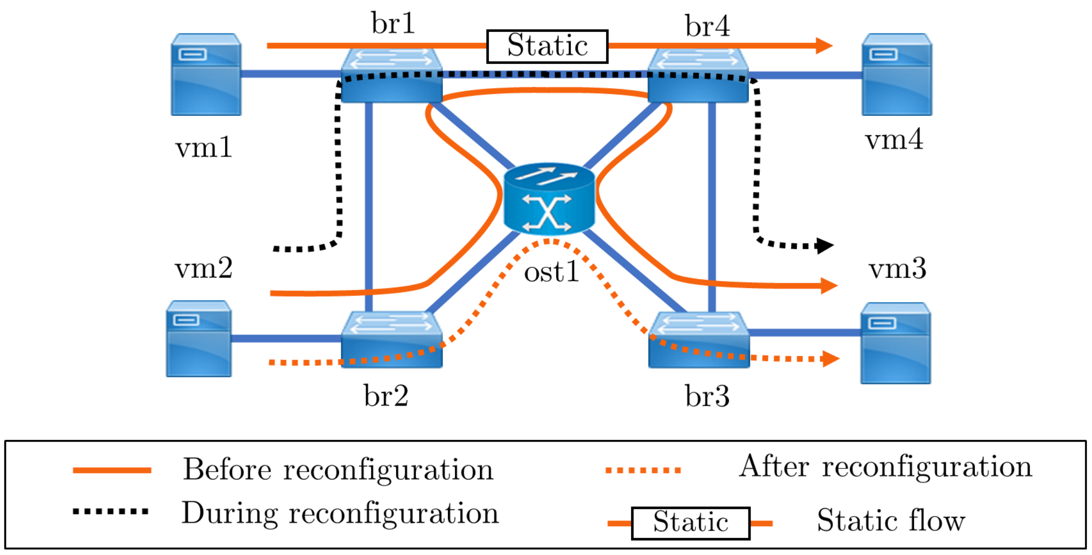
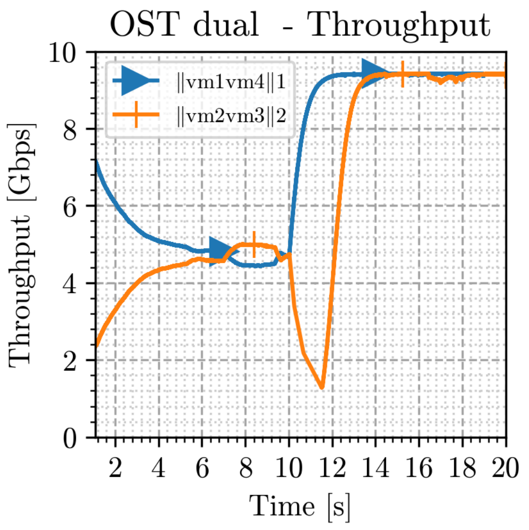
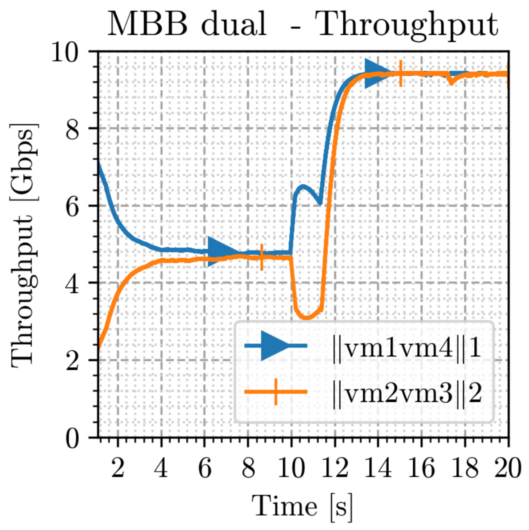


Figure 4.17: Make before break - dual data stream

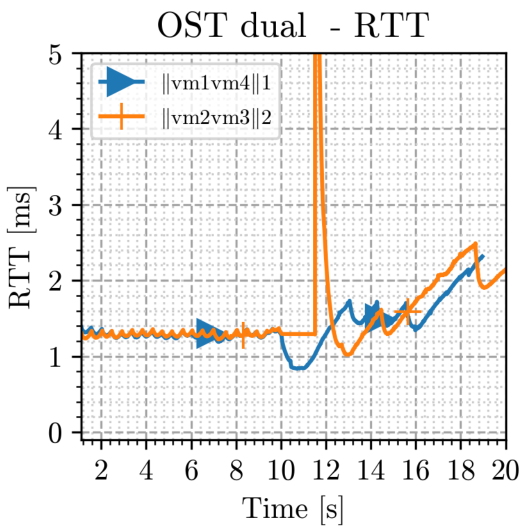


(a) OST, dual stream

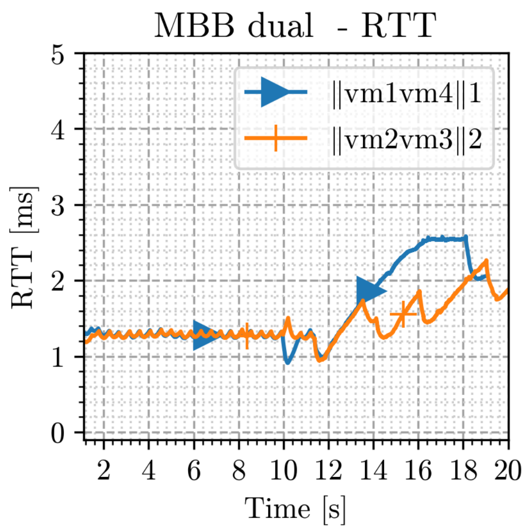


(b) MBB, dual stream

Figure 4.18: OST and MBB dual stream, throughput comparison

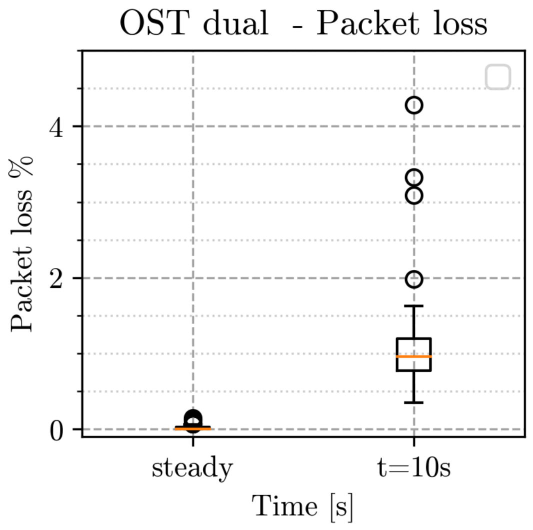


(a) OST, dual stream

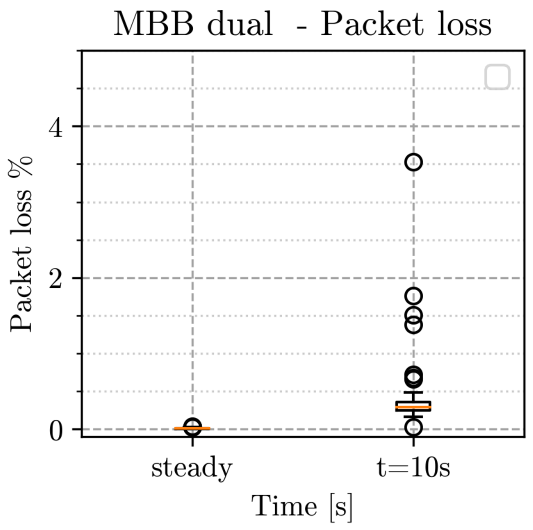


(b) MBB, dual stream

Figure 4.19: OST and MBB dual stream, RTT comparison

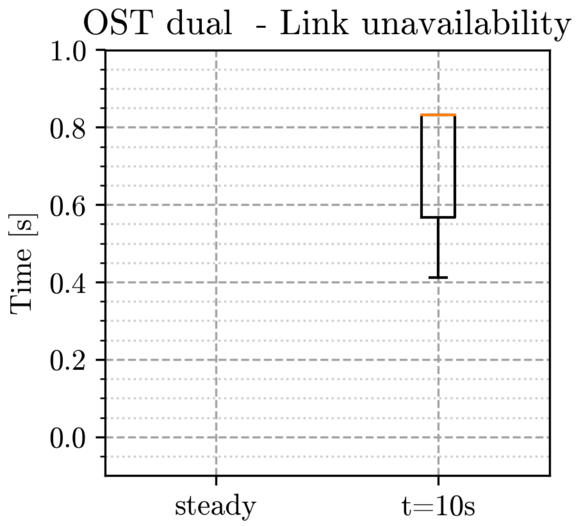


(a) OST, dual stream

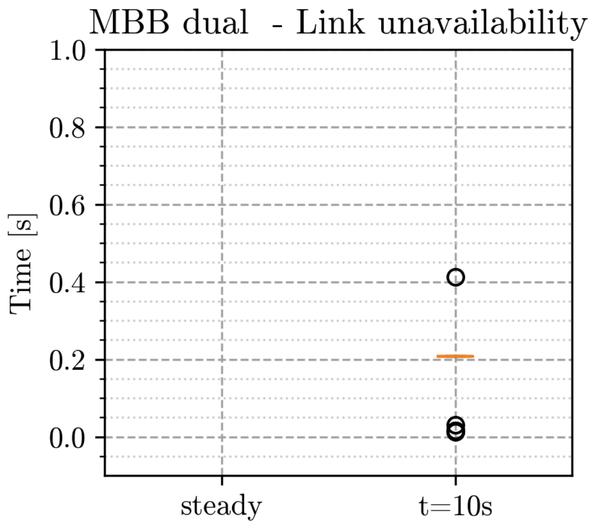


(b) MBB, dual stream

Figure 4.20: OST and MBB dual stream, packet loss comparison



(a) OST, dual stream



(b) MBB, dual stream

Figure 4.21: OST and MBB dual stream, Link unavavilability

Chapter 5

Conclusions and future work

The need for networks with lower power consumption, flexible topology, traffic awareness, lower latency and scalability to thousands of nodes motivates research in reconfigurable optical networks for datacenters and high performance computing. In Chapter 1 we presented an overview of large scale computing systems, current challenges, and related work relevant to our project in topics such as testbeds, technologies for all to all optical networks, centralized management in research and the industry, make before break approach, hitless network reconfiguration and timers in the transport layer congestion control mechanism. Overall we achieved the goals for this thesis, as we deployed a hybrid network with 10 Gbps optical links, optical and electronic packet switches, a centralized management and computing nodes to demonstrate the benefit of a make before break approach for updating routes compared to a plain optical reconfiguration (OST) with a MEMS switch. The bottleneck of the testbed was the switching latency introduced by the transceiver locking and switch polling delay, 605 milliseconds, which triggered the retransmission timeout timer (RTO) of the congestion control mechanism in the transport layer. As a result, the link unavailability observed by the application impacted directly the performance metrics, raising RTT and packet loss, and decreasing throughput as well. To overcome these drawbacks, we presented and tested our make before break approach which combines the advantages of all elements in our testbed: a centralized control plane to update the routes on demand, electronic packet switches with buffers to reduce packet loss and optimized for OpenFlow applications, and an optical switch that enables flexibility in the topology.

With our MBB approach we first updated the flows in the EPS tables sending instructions from the controller, then we provisioned the new link with the optical switch, and finally we updated the flow again in the EPS to force the traffic to pass through the new path. In a single stream of data, RTT grows up to hundreds of millisecond in the OST reconfiguration, but does not grow larger than 2.5 ms with the MBB mode. On average, the packet loss decreases from 2.8% in OST to 0.93% in MBB. Link unavailability due to RTO events went from 598 ms to 121 ms, 80% less. With MBB, throughput drops 0.6 Gbps, compared to the drop by 6 Gbps in OST.

Bandwidth steering was also improved with our MBB approach. This scenario was recreated with two data streams at 20 Gbps in total, forcing a bottleneck of 10 Gbps in the testbed. Comparing performance during reconfiguration, packet loss went from 1.28% with OST to 0.44% MBB. Link unavailability decreased from 727 ms to 185 ms, 74% less. RTT grows up to hundreds of millisecond in OST, but is not larger than 2.5 ms like in the single data stream scenario. We did not achieve hitless reconfiguration, that is, 0% packet loss, because there still was a link unavailability that generated packet drops. However, we decreased packet loss with our MBB approach, getting closer to the ideal goal. The first step to reach zero packet loss would be to decrease the transceiver locking and switch polling latency, so we do not trigger the TCP RTO timer. Other options would be to play with different RTO values, or to replace TCP with other protocols.

In the future, our testbed will be used for deploying optical networks with technologies other than MEMS, and running experiments in different areas such as machine learning and heterogeneous computing. The modular design allows modifying blocks of software or hardware described in chapters of architecture and infrastructure, without affecting other elements of the system.

References

[1] Ryu SDN Framework Community, "Ofctl rest." [Online]. Available: https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html

[2] M. J. Schulte, M. Ignatowski, G. H. Loh, B. M. Beckmann, W. C. Brantley, S. Gurumurthi, N. Jayasena, I. Paul, S. K. Reinhardt, and G. Rodgers, "Achieving exascale capabilities through heterogeneous computing," *IEEE Micro*, vol. 35, no. 4, pp. 26–36, 2015.

[3] Top500, "Top 500 the list november 2021." [Online]. Available: https://top500.org/lists/top500/2021/11/

[4] US Department of Energy, "Exascale computing project." [Online]. Available: https://www.exascaleproject.org/

[5] ——, "Exascale computing project factsheet." [Online]. Available: https://www.exascaleproject.org/wp-content/uploads/2020/01/ECP-Factsheet-Update-1-2020.pdf

[6] T. M. Evans, A. Siegel, E. W. Draeger, J. Deslippe, M. M. Francois, T. C. Germann, W. E. Hart, and D. F. Martin, "A survey of software implementations used by application codes in the exascale computing project," *The International Journal of High Performance Computing Applications*, vol. 36, no. 1, pp. 5–12, 2022. [Online]. Available: https://doi.org/10.1177/10943420211028940

[7] K.-I. Kitayama, Y.-C. Huang, Y. Yoshida, R. Takahashi, T. Segawa, S. Ibrahim, T. Nakahara, Y. Suzaki, M. Hayashitani, Y. Hasegawa, Y. Mizukoshi, and A. Hiramatsu, "Torus-topology data center network based on optical packet/agile circuit switching with intelligent flow management," *Journal of Lightwave Technology*, vol. 33, no. 5, pp. 1063–1071, March 2015.

[8] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, 1985.

[9] Z. Cao, M. Kodialam, and T. V. Lakshman, “Joint static and dynamic traffic scheduling in data center networks,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1908–1918, 2016.

[10] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, “Analyzing network health and congestion in dragonfly-based supercomputers,” in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016, pp. 93–102.

[11] X. Xiao, R. Proietti, G. Liu, H. Lu, P. Fotouhi, S. Werner, Y. Zhang, and S. J. B. Yoo, “Silicon photonic flex-lions for bandwidth-reconfigurable optical interconnects,” *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 26, no. 2, pp. 1–10, 2020.

[12] X. Xiao, R. Proietti, G. Liu, H. Lu, Y. Zhang, and S. J. B. Yoo, “Multi-fsr silicon photonic flex-lions module for bandwidth-reconfigurable all-to-all optical interconnects,” *Journal of Lightwave Technology*, vol. 38, no. 12, pp. 3200–3208, 2020.

[13] D. Bishop, C. Giles, and G. Austin, “The lucent lambdarouter: Mems technology of the future here today,” *IEEE Communications Magazine*, vol. 40, no. 3, pp. 75–79, 2002.

[14] N. Dupuis, B. G. Lee, A. V. Rylyakov, D. M. Kuchta, C. W. Baks, J. S. Orcutt, D. M. Gill, W. M. J. Green, and C. L. Schow, “Modeling and characterization of a nonblocking 4x4 mach–zehnder silicon photonic switch fabric,” *Journal of Lightwave Technology*, vol. 33, no. 20, pp. 4329–4337, 2015.

[15] A. W. Poon, X. Luo, F. Xu, and H. Chen, “Cascaded microresonator-based matrix switch for silicon on-chip optical interconnection,” *Proceedings of the IEEE*, vol. 97, no. 7, pp. 1216–1238, 2009.

[16] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, B. Thomsen, K. Shi, and H. Williams, “Sirius: A flat datacenter network with nanosecond optical switching,” in *SIGCOMM*. ACM, August 2020.

[17] X. Xiao, R. Proietti, K. Zhang, G. Liu, H. Lu, J. Messig, and S. J. B. Yoo, "Experimental demonstration of 64-port thin-clos architecture for all-to-all optical interconnects," in *Conference on Lasers and Electro-Optics.* Optica Publishing Group, 2018, p. SW4C.3. [Online]. Available: http://opg.optica.org/abstract.cfm?URI=CLEO_SI-2018-SW4C.3

[18] B. G. Lee and N. Dupuis, "Silicon photonic switch fabrics: Technology and architecture," *Journal of Lightwave Technology*, vol. 37, no. 1, pp. 6–20, 2019.

[19] J. Wang, S. Basu, C. McArdle, and L. P. Barry, "Large-scale hybrid electronic/optical switching networks for datacenters and hpc systems," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, 2015, pp. 87–93.

[20] S. J. Ben Yoo, "Prospects and challenges of photonic switching in data centers and computing systems," *Journal of Lightwave Technology*, vol. 40, no. 8, pp. 2214–2243, 2022.

[21] VMware, "What is software-defined networking (sdn)?" [Online]. Available: https://www.vmware.com/topics/glossary/content/software-defined-networking.html

[22] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 3–14. [Online]. Available: https://doi.org/10.1145/2486001.2486019

[23] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, H. Liu, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *Commun. ACM*, vol. 59, no. 9, p. 88–97, aug 2016. [Online]. Available: https://doi.org/10.1145/2975159

[24] G. M. Saridis, S. Peng, Y. Yan, A. Aguado, B. Guo, M. Arslan, C. Jackson, W. Miao, N. Calabretta, F. Agraz, S. Spadaro, G. Bernini, N. Ciulli, G. Zervas, R. Nejabati, and D. Simeonidou, "Lightness: A function-virtualizable software defined data center network with all-optical circuit/packet switching," *Journal of Lightwave Technology*, vol. 34, no. 7, pp. 1618–1627, 2016.

[25] W. Miao, F. Agraz, S. Peng, S. Spadaro, G. Bernini, J. Perelló, G. Zervas, R. Nejabati, N. Ciulli, D. Simeonidou, H. Dorren, and N. Calabretta, "Sdn-enabled ops with qos guarantee for reconfigurable virtual data center networks," *Journal of Optical Communications and Networking*, vol. 7, no. 7, pp. 634–643, 2015.

[26] Oracle, "Introducing the internet protocol suite." [Online]. Available: https://docs.oracle.com/cd/E19683-01/806-4075/6jd69oa75/index.html

[27] Internet Engineering Task Force, "Computing tcp's retransmission timer." [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6298

[28] iPerf authors, "iperf - the ultimate speed test tool for tcp, udp and sctp." [Online]. Available: https://iperf.fr/

[29] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding tcp incast throughput collapse in datacenter networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, ser. WREN '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 73–82. [Online]. Available: https://doi.org/10.1145/1592681.1592693

[30] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained tcp retransmissions for datacenter communication," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 303–314. [Online]. Available: https://doi.org/10.1145/1592568.1592604

[31] M. Allman and V. Paxson, "On estimating end-to-end network path properties," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, p. 263–274, aug 1999. [Online]. Available: https://doi.org/10.1145/316194.316230

[32] 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*, vol. 22, no. 2, pp. 498–511, 2014.

[33] A. M. Abdelmoniem, B. Bensaou, and A. J. Abu, "Mitigating incast-tcp congestion in data centers with sdn," *Annals of Telecommunications*, vol. 73, pp. 263–277, 2018.

[34] A. M. Abdelmoniem and B. Bensaou, "T-racks: A faster recovery mechanism for tcp in data center networks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 3, pp. 1074–1087, 2021.

[35] P. Sarolahti and A. Kuznetsov, "Congestion control in linux TCP," in *2002 USENIX Annual Technical Conference (USENIX ATC 02)*. Monterey, CA: USENIX Association, Jun. 2002. [Online]. Available: https://www.usenix.org/conference/2002-usenix-annual-technical-conference/congestion-control-linux-tcp

[36] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "Hpcc: High precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 44–58. [Online]. Available: https://doi.org/10.1145/3341302.3342085

[37] S. Arslan, S. Ibanez, A. Mallery, C. Kim, and N. McKeown, "Nanotransport: A low-latency, programmable transport layer for nics," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, ser. SOSR '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 13–26. [Online]. Available: https://doi.org/10.1145/3482898.3483365

[38] W. Golab and R. Boutaba, "Policy-driven automated reconfiguration for performance management in wdm optical networks," *IEEE Communications Magazine*, vol. 42, no. 1, pp. 44–51, 2004.

[39] B. Jaumard, H. Pouya, and D. Coudert, "Make-before-break wavelength defragmentation," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, 2018, pp. 1–5.

[40] H. Li and J. Wu, "Survey of wdm network reconfiguration: topology migrations and their impact on service disruptions," *Telecommunication Systems*, vol. 60, pp. 349–366, 2015.

[41] B. Jaumard, H. Q. Duong, R. Armolavicius, T. Morris, and P. Djukic, "Efficient real-time scalable make-before-break network re-routing," *Journal of Optical Communications and Networking*, vol. 11, no. 3, pp. 52–66, 2019.

[42] B. Jaumard, H. Pouya, and D. Coudert, "Wavelength defragmentation for seamless migration," *Journal of Lightwave Technology*, vol. 37, no. 17, pp. 4382–4393, 2019.

[43] H. Duong, B. Jaumard, and D. Coudert, "Minimum disturbance rerouting to optimize bandwidth usage," in *2021 International Conference on Optical Network Design and Modeling (ONDM)*, 2021, pp. 1–6.

[44] K. Bala, G. Ellinas, M. Post, C.-C. Shen, J. Wei, and N. Antoniades, "Towards hitless reconfiguration in wdm optical networks for atm transport," in *Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference*, vol. 1, 1996, pp. 316–320 vol.1.

[45] Juniper Networks, "Forward error correction (fec) and bit error rate (ber)." [Online]. Available: https://www.juniper.net/documentation/us/en/software/junos/interfaces-ethernet/topics/topic-map/fec-ber-otn-interfaces.html

[46] R. Proietti, C. Qin, B. Guan, Y. Yin, R. P. Scott, R. Yu, and S. J. B. Yoo, "Rapid and complete hitless defragmentation method using a coherent

rx lo with fast wavelength tracking in elastic optical networks," *Opt. Express*, vol. 20, no. 24, pp. 26 958–26 968, Nov 2012. [Online]. Available: http://opg.optica.org/oe/abstract.cfm?URI=oe-20-24-26958

[47] C. Qin, R. Proietti, B. Guan, Y. Yin, R. P. Scott, R. Yu, and S. J. B. Yoo, "Demonstration of multi-channel hitless defragmentation with fast auto-tracking coherent rx los," in *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2013*. Optica Publishing Group, 2013, p. OW3A.1. [Online]. Available: http://opg.optica.org/abstract.cfm?URI=OFC-2013-OW3A.1

[48] M. Scaffardi, V. Vercesi, A. Sgambelluri, and A. Bogoni, "Hitless reconfiguration of a ppln-based multiwavelength source for elastic optical networks," *J. Opt. Commun. Netw.*, vol. 8, no. 2, pp. 85–92, Feb 2016. [Online]. Available: http://opg.optica.org/jocn/abstract.cfm?URI=jocn-8-2-85

[49] J. Guo, S. Zhang, R. Proietti, Z. Cao, G. Yuan, and S. Yoo, "Fast and hitless topology management of awgr-based optical networking for data centers," in *2017 European Conference on Optical Communication (ECOC)*, 2017, pp. 1–3.

[50] W. Wang, S. Das, and T. S. E. Ng, "Abstractions for reconfigurable hybrid network update and a consistent update approach," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Optical Systems*, ser. OptSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 6–11. [Online]. Available: https://doi.org/10.1145/3473938.3474506

[51] M. Y. Teh, Z. Wu, M. Glick, S. Rumley, M. Ghobadi, and K. Bergman, "Performance trade-offs in reconfigurable networks for hpc," *Journal of Optical Communications and Networking*, vol. 14, no. 6, pp. 454–468, 2022.

[52] Y. Shen, M. H. N. Hattink, P. Samadi, Q. Cheng, Z. Hu, A. Gazman, and K. Bergman, "Software-defined networking control plane for seamless integration of multiple silicon photonic switches in datacom networks," *Opt.*

Express, vol. 26, no. 8, pp. 10 914–10 929, Apr 2018. [Online]. Available: http://opg.optica.org/oe/abstract.cfm?URI=oe-26-8-10914

[53] A. Thyagaturu, A. Mercian, M. McGarry, M. Reisslein, and W. Kellerer, "Software defined optical networks (sdons): A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4, pp. 2738–2786, Oct. 2016, publisher Copyright: © 2016 IEEE.

[54] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In vini veritas: Realistic and controlled network experimentation," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, p. 3–14, aug 2006. [Online]. Available: https://doi.org/10.1145/1151659.1159916

[55] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *IEEE Wireless Communications and Networking Conference, 2005*, vol. 3, 2005, pp. 1664–1669 Vol. 3.

[56] G. C. Hadjichristofi, A. Brender, M. Gruteser, R. Mahindra, and I. Seskar, "A wired-wireless testbed architecture for network layer experimentation based on orbit and vini," in *Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, ser. WinTECH '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 83–90. [Online]. Available: https://doi.org/10.1145/1287767.1287783

[57] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, p. 255–270, dec 2003. [Online]. Available: https://doi.org/10.1145/844128.844152

[58] F. Hermenier and R. Ricci, "How to build a better testbed: Lessons from a decade of network experiments on Emulab," in *Proceedings of the 8th International ICST*

Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom), Jun. 2012.

[59] B. Lokesh and N. Rajagopalan, “Orchestrator for synchronizing network events in sdns,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4365–4375, 2021.

[60] Pica8, “Openflow (open flow) switches in enterprise networks: Glossary definition.” [Online]. Available: https://www.pica8.com/openflow-switch/

[61] ——, “Overlap flow.” [Online]. Available: https://docs.pica8.com/display/PicOS21119sp/Overlap+flow

[62] Juniper Networks, “Junos operating system.” [Online]. Available: https://www.juniper.net/us/en/products/network-operating-system/junos-os.html

[63] Pica8, “Crossflow mode.” [Online]. Available: https://docs.pica8.com/display/PICOS2111cg/Crossflow+Mode+Introduction

[64] IETF, “Rfcs.” [Online]. Available: https://www.ietf.org/standards/rfcs/

[65] ——, “Rfc 3697: Ipv6 flow label specification.” [Online]. Available: https://www.ietf.org/rfc/rfc3697.txt

[66] ——, “Rfc 2722: Traffic flow measurement: Architecture.” [Online]. Available: https://www.ietf.org/rfc/rfc2722.txt

[67] Intel, “Intel ethernet converged network adapters x710 10 gbe.” [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/servers-unified-computing/ucs-c-series-rack-servers/intel-x710-product-brief.pdf

[68] ——, “Intel product compatibility tool find compatibility information for intel products.” [Online]. Available: https://compatibleproducts.intel.com/

[69] Open Compute Project Foundation, “Open network install environment.” [Online]. Available: https://www.opencompute.org/wiki/Networking/ONIE

[70] Edgecore, “As7312-54xs 3.6t data center switch.” [Online]. Available: https://www.edge-core.com/productsInfo.php?cls=1&cls2=5&cls3=155&id=296

[71] . Polatis Inc, “Single mode network optical switch up to 32x32 ports,” Bedford, MA, Datasheet, 2013.

[72] Polatis, “Data center networks - all optical switches.” [Online]. Available: https://www.polatis.com/data-center-colocation-network-optical-switch-solutions-cloud-computing-datacenter-low-loss-swit
asp

[73] ——, “Polatis technology - directlight® beam-steering all-optical switch.” [Online]. Available: https://www.polatis.com/polatis-all-optical-switch-technology-lowest-loss-highest-performance-directlight-beam-steering.asp

[74] Cisco, “Sfp.” [Online]. Available: https://community.cisco.com/t5/networking-documents/sfp/ta-p/3116189

[75] Digikey, “Afbr-709smz 10gb ethernet, 850 nm, 10gbase-sr/sw, sfp+ transceiver.” [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Avago%20PDFs/AFBR-709SMZ.pdf

[76] ——, “Sfp10g-er 10gbase, sfp+, er, smf transceiver 1550nm, 40km reach, duplex lc connector transceiver.” [Online]. Available: https://approvednetworks.com/content/data%20sheets/transceivers/SFP%2010G/SFP10G-ER.pdf

[77] S. Srivastava, S. Anmulwar, A. Sapkal, T. Batra, A. K. Gupta, and V. Kumar, “Comparative study of various traffic generator tools,” in *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*, 2014, pp. 1–6.

[78] “Packeth.” [Online]. Available: http://packeth.sourceforge.net/packeth/Home.html

[79] “Ostinato traffic generator for engineers.” [Online]. Available: https://ostinato.org/

[80] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, “D-itg distributed internet traffic generator,” in *First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings.*, 2004, pp. 316–317.

[81] U.S Navy, “Multi-generator (mgen) network test tool.” [Online]. Available: https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/MGEN/

[82] Cisco, “Trex realistic traffic generator.” [Online]. Available: https://trex-tgn.cisco.com/trex/doc/trex_vm_manual.html

[83] ——, “Trex manual - hardware recommendations.” [Online]. Available: https://trex-tgn.cisco.com/trex/doc/trex_manual.html#_overview_of_trex

[84] Redhat, “Automated open vswitch pvp testing.” [Online]. Available: https://developers.redhat.com/blog/2017/09/28/automated-open-vswitch-pvp-testing#create_the_loopback_virtual_machine

[85] Codilime, “A traffic generator for measuring network performance.” [Online]. Available: https://codilime.com/blog/a-traffic-generator-for-measuring-network-performance/

[86] ——, “Multinode trex wiki.” [Online]. Available: https://github.com/codilime/trex-core/wiki

[87] N. Terzenidis, G. Giamougiannis, A. Tsakyridis, D. Spasopoulos, F. Yan, N. Calabretta, C. Vagionas, and N. Pleros, “Performance analysis of a 1024-port hipoλaos ops in dcn, hpc, and 5g fronthauling ethernet applications,” *Journal of Optical Communications and Networking*, vol. 13, no. 7, pp. 182–192, 2021.

[88] A. A. M. Alraawi and S. A. N. Adam, “Performance evaluation of controller based sdn network over non-controller based network in data center network,” in *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 2021, pp. 1–4.

[89] S. J. Arévalo-Cordero, P. L. Gallegos-Segovia, P. E. Vintimilla-Tapia, J. F. Bravo-Torres, E. J. Cedillo-Elias, and V. M. Larios-Rosillo, “Data traffic management in a hybrid cloud composed of openstack and azure,” in *2019 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2019, pp. 1–6.

[90] B. Dwinanto and A. S. Arifin, “Integrated strategy framework to improve quality of network on the bmkg communication network system,” in *2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, 2021, pp. 244–251.

[91] R. Yunos, N. M. Noor, and S. A. Ahmad, “Performance evaluation between ipv4 and ipv6 on mpls linux platform,” in *2010 International Conference on Information Retrieval Knowledge Management (CAMP)*, 2010, pp. 204–208.

[92] Y. Sharma, M. G. Khan, J. Taheri, and A. Kassler, “Performance benchmarking of virtualized network functions to correlate key performance metrics with system activity,” in *2020 11th International Conference on Network of the Future (NoF)*, 2020, pp. 73–81.

[93] V. J. D. Barayuga and W. E. S. Yu, “Packet level tcp performance of nat44, nat64 and ipv6 using iperf in the context of ipv6 migration,” in *2015 5th International Conference on IT Convergence and Security (ICITCS)*, 2015, pp. 1–3.

[94] V. J. D. Barayuga and W. E. S. Y, “Study of packet level udp performance of nat44, nat64 and ipv6 using iperf in the context of ipv6 migration,” in *2014 International Conference on IT Convergence and Security (ICITCS)*, 2014, pp. 1–6.

[95] Y. Zhou, C. Sun, H. H. Liu, R. Miao, S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi, P. Zhang, D. Cai, M. Zhang, and M. Xu, “Flow event telemetry on programmable data plane,” in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM ’20.

New York, NY, USA: Association for Computing Machinery, 2020, p. 76–89. [Online]. Available: https://doi.org/10.1145/3387514.3406214

[96] O. Michel, J. Sonchack, G. Cusack, M. Nazari, E. Keller, and J. M. Smith, "Software packet-level network analytics at cloud scale," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 597–610, 2021.

[97] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca, "Planck: Millisecond-scale monitoring and control for commodity networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, p. 407–418, aug 2014. [Online]. Available: https://doi.org/10.1145/2740070.2626310

[98] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 267–280. [Online]. Available: https://doi.org/10.1145/1879141.1879175

[99] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, "High-resolution measurement of data center microbursts," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 78–85. [Online]. Available: https://doi.org/10.1145/3131365.3131375

[100] The tcpdump group, "Tcpdump and libpcap documentation." [Online]. Available: https://www.tcpdump.org/

[101] Zabbix LLC, "Zabbix - the enterprise-class open source network monitoring solution." [Online]. Available: https://www.zabbix.com/

[102] ——, "Zabbix docker containers." [Online]. Available: https://blog.zabbix.com/zabbix-docker-containers/7150/

[103] sFlow Org, "sflow." [Online]. Available: https://sflow.org/

[104] Pica8, "Interface configuration commands." [Online]. Available: https://docs.pica8.com/display/PicOS37sp/set+interface+gigabit-ethernet+duplex+auto

[105] S. Committee, "SFF8431: Enhanced Small Form Factor Pluggable Module SFP+," Saratoga, CA, Standard, Jul. 2009.

[106] Pica8, "ovs-ofctl bundle." [Online]. Available: https://docs.pica8.com/pages/viewpage.action?pageId=3084115

[107] Wireshark, "tshark(1) manual page." [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

[108] Linux, "ip-route(8)." [Online]. Available: https://man7.org/linux/man-pages/man8/ip-route.8.html

[109] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, p. 64–74, jul 2008. [Online]. Available: https://doi.org/10.1145/1400097.1400105