Low latency and high throughput dynamic network infrastructures for high performance datacentre interconnects

Small or medium-scale focused research project (STREP)
Co-funded by the European Commission within the Seventh Framework Programme
Project no. 318606
Strategic objective: Future Networks (ICT-2011.1.1)

Start date of project: November 1st, 2012 (36 months duration)

SEVENTH FRAMEWORK PROGRAMME

Deliverable D3.3

Evaluation of OPS, OCS and TOR switches' prototypes and programmable optical NICs including electronics control and interfaces

Due date:	28/02/2015
Submission date:	08/04/2015
Deliverable leader:	TUE
Author list:	Wang Miao (TUE), Nicola Calabretta (TUE), Harm Dorren (TUE), Yi Shu (UNIVBRIS), Yan Yan (UNIVBRIS), Bingli Guo (UNIVBRIS), Chris Jackson (UNIVBRIS), Georgios Zervas (UNIVBRIS), Shuping Peng (UNIVBRIS), Reza Nejabati (UNIVBRIS), Dimitra Simeonidou (UNIVBRIS), Salvatore Spadaro (UPC), Fernando Agraz (UPC), Jose Carlos Sancho (BSC), Hugo Meyer (BSC), Montse Farreras (BSC)

Dissemination Level

☒	**PU:**	Public
☐	**PP:**	Restricted to other programme participants (including the Commission Services)
☐	**RE:**	Restricted to a group specified by the consortium (including the Commission Services)
☐	**CO:**	Confidential, only for members of the consortium (including the Commission Services)

Abstract

This document presents the implementation and experimental evaluation of the hybrid data plane prototypes developed in LIGHTNESS.

LIGHTNESS data plane integrates innovative optical switching technologies including programmable optical Network Interface Card (NIC), optical Top of Rack (ToR) switch, optical packet switching (OPS) and optical circuit switching (OCS)to provide scalable, high bandwidth and low latency optical interconnectivity within the data centre network (DCN) infrastructure. The architecture-on-demand (AoD) switch nodes bound all the switching elements and enable the flexible configuration of the intra- and inter-cluster communication for the flat DCN.

The results of the prototypes implementation for OPS, OCS and TOR nodes as well as programmable optical NICs are presented in this deliverable. It also provides a final report about the overall data plane architecture for LIGHTNESS. Experimental assessments of the prototypes, interfaces between switching nodes and interfaces with control plane are reported in detail. The benefits of synthetic hybrid data plane have been analyzed and evaluated with scientific applications. An all-optical solution for intra- DCN to inter-DCN interface is also introduced. As a comprehensive summary of the achievements in WP3, the results descript in this document will be employed as guideline for the experimental evaluation of the DCN integration in the WP5.

Table of Contents

Figure Summary

Table Summary

0. Executive Summary

The exponential growth of the Internet traffic is boosting the requirements of DCN for higher capacity, lower latency and more flexible interconnectivities. To overcome the hardware limitations of multi-tier electrical DCNs, optical switching technologies exploiting space, time, and wavelength multiplexing have been investigated in several projects for effectively accommodating data centre (DC) applications [1]. A novel flat and scalable data plane architecture which supports high-bandwidth and agile network connectivity has been defined in LIGHTNESS. In cooperation with a unified network control plane, the data plane elements can be dynamically configured to adapt to the on-demand requirements.

The objectives of WP3 are the design and implementation of the switching technologies involved in the data plane. The high-end NIC implemented with Field-Programmable Gate Array (FPGA)-based platform aggregates the traffic from the dedicated server and transmits the data to the pure optical ToR for intra- and inter-rack communication. The AoD-based OCS backplane and the OPS interconnect all the ToRs and specified to handle long-lived and short-lived inter-rack traffic, respectively. This deliverable mainly focuses on the evaluations of the prototypes including optical ToR, OCS, OPS and programmable NIC. Based on the design scheme reported in D3.1 [2] and the preliminary evaluation results descript in D3.2 [3], the prototypes of the switching nodes are implemented and experimentally assessed.

As the final deliverable of the WP3, not only the system evaluation of each switching element, but also the evaluation of the overall data plane architecture is reported. The performances of the programmable OCS/OPS DCN architecture allowing for dynamic inter- and intra- cluster network topology and technology synthesis are analysed. The performance of the synthetic hybrid data plane is analysed and evaluated with scientific applications. The results of the data plane integration in this document will serve as a complementary content for experimental validation in WP5.

To enable the dynamic control and operation from the Software Defined Networking (SDN)-based control plane, the hardware and software control interfaces have been developed for the different switching nodes (ToR, AoD and OPS). As OpenFlow (OF) has been chosen as the communication protocol for the southbound interface, dedicated OF-agents are implemented to facilitate the messages delivering between the SDN controller and switching elements. The implementation schemes of the OF-agents are explained in detail and the performances of the control interfaces are experimentally evaluated. The investigation results will be further utilized and finalized in WP5 during the integration of the data plane and the control plane developed in WP4.

1. Introduction

1.1. Motivation and scope

The bandwidth of the data centre networks (DCNs) scales with roughly a factor of 1000 every 10 years [4]. Nowadays the DC traffic is handled by the multi-tier electronic switches. Limited by the scaling issues including latency and bandwidth bottleneck of the electronic switches, the high capacity and flexibility required by next generation DCN applications justifies the investigation of high-end optical switching technologies. Besides the switching nodes, a more efficient topology is also required to improve the performance in terms of bandwidth and latency. To this end, the LIGHTNESS proposes a flat data plane architecture integrating both OCS and the OPS switching technologies to deal with the inconsistent application requirements. Programmable NIC is directly plugged into each server and is given the intelligence to diverge the traffic to OCS/OPS according to the specific requirements. Benefiting from this, the ToR could be implemented optically eliminating the extra processing efforts. Control interfaces are developed for the switching elements which enables the control plane to flexibly and dynamically provision of the data plane resources.

The switching nodes have been properly designed and experimentally implemented [3,4]. The final implementation and assessments of the prototypes are presented in this document. This includes the opto-electronic interfaces and the controlling interfaces to SDN controller which have been tested and evaluated in detail. To enable the overall operation of the data plane, different switching technologies are integrated and the synthesized hybrid OCS/OPS data plane has been assessed. The switching performance of the individual sub-systems and as a whole system has been examined which makes this deliverable a summary of the results achieved in WP3 along its whole duration. The prototypes will be employed in the data plane integration and data plane/control plane integration defined in WP5.

1.2. Structure of the document

This document is structured as follows.

Chapter 2 provides the detailed implementation schemes of the LIGHTNESS data plane including both intra-cluster and inter-cluster scenarios. Two different design schemes for ToR switching node are investigated. The OCS/OPS test bed used for evaluating the overall data plane performance is introduced. A summary of the interfaces exposed towards the control plane is also provided.

Chapter 3 describes the implementations of the novel optical ToR and the programmable NIC. The advantages over the traditional solutions are also presented. The characteristics of the optical ToR and the capabilities of the optical NIC are reported. It also presents the evaluation results of the NIC in terms of throughput and latency and the interfaces with control plane for both nodes.

Chapter 4 presents the implementation and system evaluation of the OCS node. In particular, the intra-cluster network is investigated and the opto-electronic interface used to provide bandwidth adjustable transmission is explained.

Chapter 5 focuses on the modular OPS node and reports the related implementation and assessment results. The preparation of the compact printed circuit board (PCB) designs for the prototype is described and the performance evaluation of the burst mode operation is given. Capabilities of network reconfiguration benefited from SDN have been evaluated. The interface with the control plane is also discussed at the end of this chapter.

Chapter 6 presents the benefits of synthetic hybrid DCN architectures. It introduces the performance evaluation of programmable synthetic hybrid OCS/OPS DCN. The case of multi-tenant scenario where each tenant requests several virtual slices according to their needs is investigated. The benefits of the hybrid DCN are analyzed considering dynamic on-demand DCN configuration. An experimental implementation of the synthetic data plane is also evaluated. Finally, the evaluation of the performance with real world scientific applications for the LIGHTNESS solution is reported.

Chapter 7 provides an all-optical multi-dimensional and programmable solution for inter-DCN communications. High speed interface for inter-DCN communication is described and the experimental evaluation results are given.

2. LIGHTNESS data plane implementation and evaluation

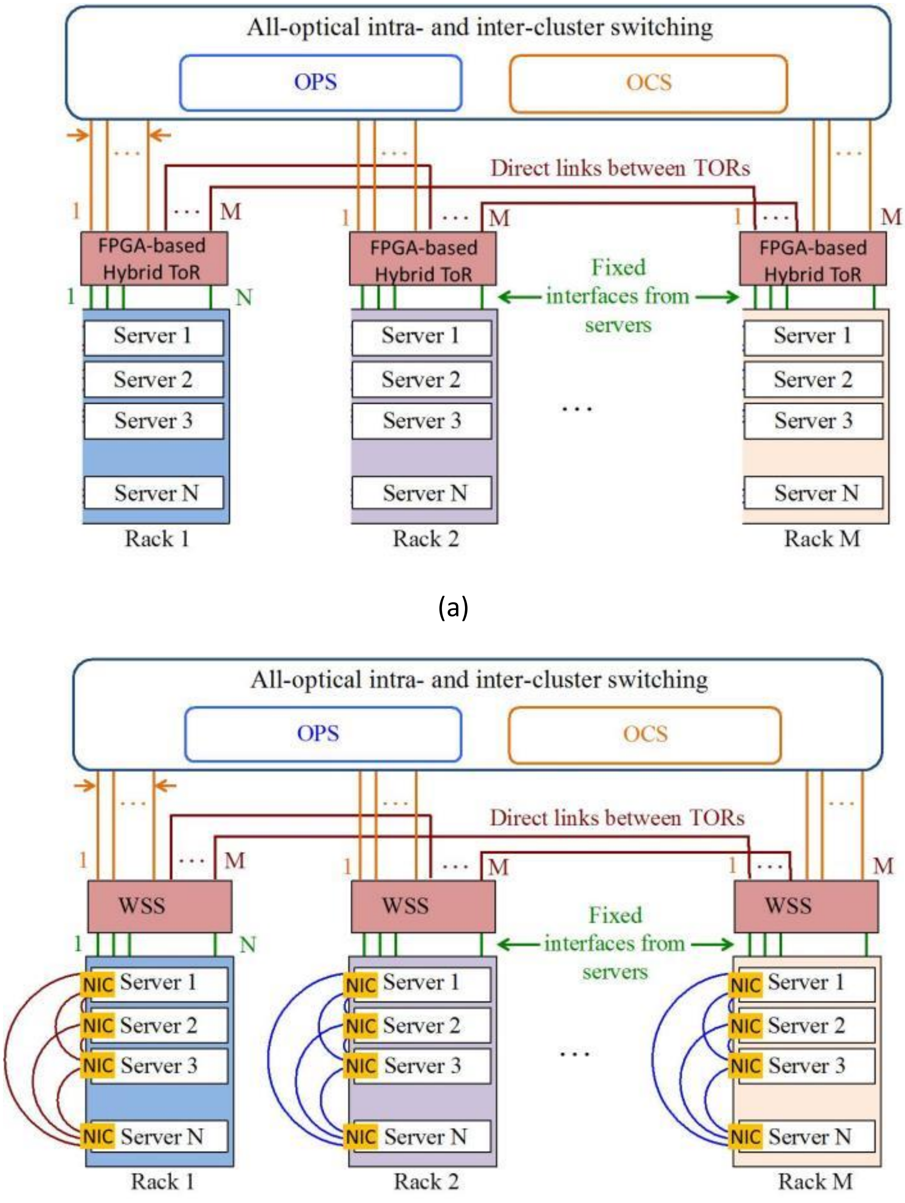
2.1. Implementation of the hybrid OCS/OPS data plane testbed

As presented in D3.1 [2], the intra-DC network architecture is displayed in Figure 2.1. Two different schemes for ToR switch design are investigated.

In Figure 2.1(a), a hybrid ToR switch is designed and implemented based on high-speed FPGA platforms. It can interface the servers to the hybrid OPS/OCS interconnects network by efficient aggregation and mapping of the traffic in long-lived data flows (to be transported by OCS network) and in short-lived packet flows (to be transported by OPS network). It includes several interfaces:

- The interface with control plane communicates with the controller by 10Gbps link, and makes the FPGA-based hybrid ToR switch controllable from control plane;
- The interface with server is 10Gbps enhanced small form-factor pluggable (SFP+), through that the hybrid ToR switch receives/sends Ethernet frames from/to the server
- To realize OPS transmission, a label interface is employed to send/receive labels and ACK signals to/from Switch Module FPGA
- The inter-rack communication is through 10x10Gbps Dense Wavelength Division Multiplexing (DWDM) C Form-factor Pluggable (CFP) links.

In Figure 2.1(b), the Hybrid NIC is deployed directly on the server in place of the commercial NIC. Thus, the hybrid OPS/OCS switchover functions can be implemented in the Hybrid NIC on the server. The ToR switch does not need to employ any electronic platform but contains only pure optic components such as wavelength selective switches (WSS) or spectrum selective switches (SSS). Furthermore, the servers in the rack can directly communicate with each other. More discussion about optical ToR and hybrid NIC can be found in Chapter 3.


All-optical intra- and inter-cluster switching
OPS
OCS
Direct links between TORs
1
M
FPGA-based Hybrid ToR
N
Fixed interfaces from servers
Server 1
Server 2
Server 3
Server N
Rack 1
Rack 2
Rack M
(a)
All-optical intra- and inter-cluster switching
OPS
OCS
Direct links between TORs
WSS
NIC
Server 1
Server 2
Server 3
Server N
Rack 1
Rack 2
Rack M
Fixed interfaces from servers

(a)

(b)

Figure 2.1: LIGHTNESS data plane architecture with (a) FPGA-based electrical ToR and (b) hybrid NIC and WSS-based optical ToR

The OPS and OCS schemes are accommodated seamlessly by network function programmable (NFP) node deploying Architecture-on-Demand (AoD) concept [5]. The AoD configuration utilizes a large-port-count fibre switch (LPFS)[6] as the optical backplane. The group of optical modules, acting as the plug-in function pool in the AoD architecture, include Wavelength Division Multiplexing (WDM) modules (e.g. Arrayed Waveguide Grating (AWG), SSS, etc.), Time Division Multiplexing (TDM) modules (e.g. fast switch, OPS system, etc.), and other modules such as splitters, Erbium doped fibre amplifiers (EDFAs) or optical delay lines. With this architecture, different arrangements of inputs, modules and outputs can be constructed by setting up appropriate cross-connections in the optical backplane. Thus, synthetic node architectures can be dynamically created involving only the required transmission and functionality. Utilizing the programmability provided by AoD, the NFP node is capable to deliver various network functions on demand. The implemented functions in the node can operate in aggregation as well as in isolated groups, which enables the node to fully support hardware virtualization: creating slices of the node and associating them with arbitrary traffic types.

Figure 2.2 illustrates an example for synthetic OPS/OCS intra-cluster connection. OPS/OCS programmable channels provided by hybrid NIC on each server are connected to the NFP node. Then, if OPS function is required, the corresponding link can be connected to an OPS module directly (blue dash line) or connected to SSS, where link capacity can be readjusted or reassembled with other OPS channels from other ToRs (e.g. for multicasting), before the OPS module (red dash line). Likewise, OCS links can be established by connecting OCS-enable channels, after reassembling or directly, between two ToRs through the NFP node (blue solid line). In addition, different OPS modules can be cascaded on demand by setting OCS links between them (red solid line). All ToRs and NICs are connected with control plane with Ethernet raw sockets, and so does the NFP node. Thus, the control layer can configure and enable OPS/OCS function for each intra-cluster connection.

Such intra-cluster architecture design can be easily extended to the inter-cluster DCN architecture, as shown in Figure 2.3. With all clusters utilizing the same intra-cluster infrastructure, another NFP node is used as the inter-cluster NFP node for the interconnection between clusters. Each cluster is connected to the inter-cluster NFP node with a bunch of fibre from the intra-cluster NFP node. A group of Inter-cluster OPS modules are connected to the inter-cluster NFP node as well. ToRs in different clusters can communicate with each other through relayed OCS links or OPS modules provided by inter- and intra-cluster NFP nodes.

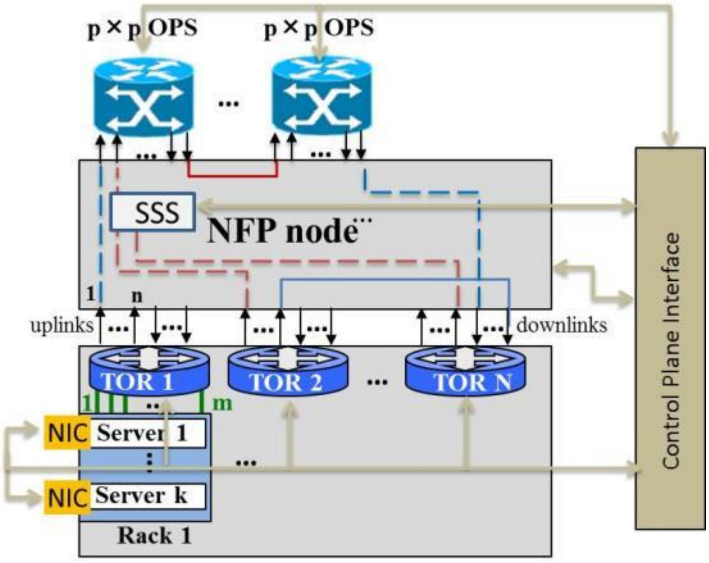


Figure 2.2: LIGHTNESS intra-cluster DCN architecture

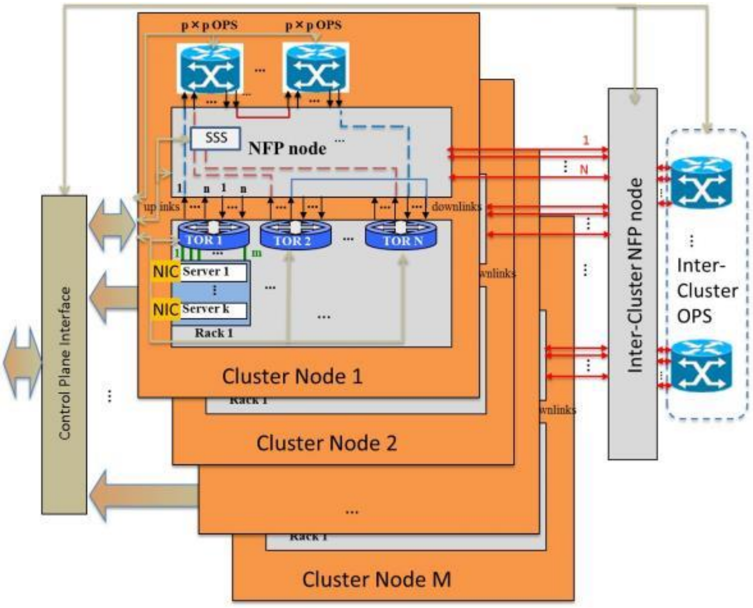


Figure 2.3: LIGHTNESS inter-cluster DCN architecture

The implementation of the hybrid OCS/OPS data plane test bed is illustrated in Figure 2.4. All the transmitters and receivers from hybrid NICs are connected to a 192×192 fibre switch (Polatis) through 4X16 WSS (FinisarWaveShaper) [7]. The OPS node, additional WSSs (1X4 or 4X16), EDFAs, couplers and splitters are already connected to the fibre switch for network function programmability. At the transmitter side, the FPGA-based NIC parses the input traffic from servers. The inter-function switch sends the traffic either to 10Gbps SFP+links or the optical bandwidth variable transmitters (BVT) according to its destination and capacity. For the SFP+ link, the traffic is aggregated and then encapsulated to OPS or OCS signals. Programmable OPS/OCS over Space Division Multiplexing (SDM) signals are provided through two SFP+ transceivers for intra-cluster communication. The OPS signals are synchronized with the OPS node in the same cluster. For the optical BVT link, the traffic is also aggregated and sent to the optical BVT to generate optical signals with variable capacities for inter-DCN or inter–cluster communication. In the setup of the optical BVT, an electrical multiplexer multiplexes the four 10 Gbaud input signals to a 40 Gbaud electrical signal, which is used to drive an IQ modulator to achieve 40 Gbaud QPSK signals. An optical emulator is followed to generate 40 Gbaud 16QAM signals. A bank of tunable external cavity lasers (ECLs) is feeding the modulators.

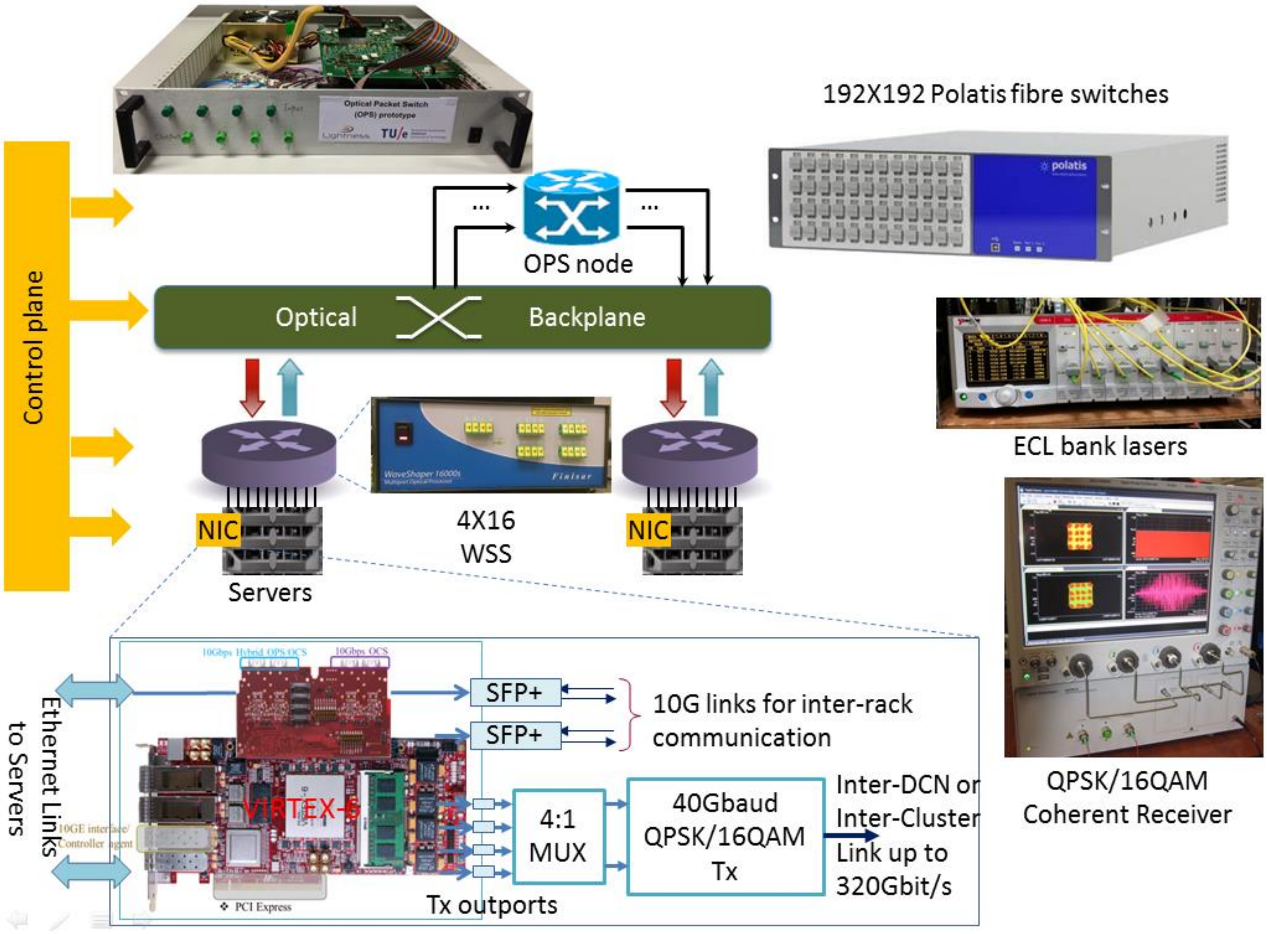


Figure 2.4: Implementation of the hybrid OCS/OPS data plane test bed

2.2. Implementation of the interface to control plane

The interface between the control plane and the data plane, i.e. the SDN southbound interface, is the responsible for enabling the SDN-based control and operation of the DCN. In LIGHTNESS, the southbound interface has been implemented by means of the OpenFlow (OF) protocol, which was extended to support the particular features of the hybrid all-optical data plane proposed in the project [8]. Furthermore, to enable the OF-based configuration in the network elements (NEs) of the data plane, a set of OF-Agents was developed. These OF-Agents reside on top of each NE (i.e. OCS, OPS, ToR and NIC) and their role is two-fold. On the one hand, the agents translate the extended OF protocol messages coming from the SDN-controller into a set of operations that perform the actual configuration of the device. On the other hand, the agents collect monitoring information from the devices and send it to the SDN controller formatted as extended OF protocol messages.

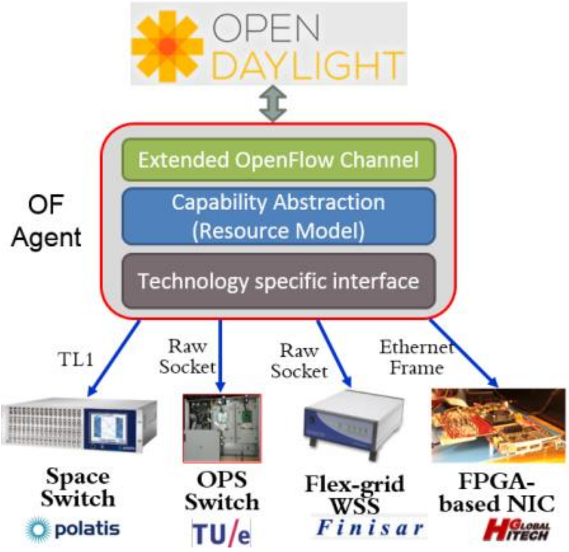


Figure 2.5: Interface between control plane and data plane

Figure 2.5 provides the general view of the communication interface between the SDN controller and optical devices developed/employed in LIGHTNESS. The architecture of the OF agents that have been developed for each data plane element is depicted as well (more details can be found in D4.3). In brief, agents utilize the NE's specific management interface (like Simple Network Management Protocol (SNMP), Transaction Language 1 (TL1), Vendor Application Programming Interface (API), etc.) to communicate with the data plane. A generic resource model is used to maintain NE's configuration (i.e., port capabilities and switching constraints). The resource model and specification deals with the complexity of the NEs capabilities and represents them to the controller in a generic way. Finally, the OF agent includes an OF API, which is responsible for the communication with the SDN controller through an extended version of the OF protocol.

Table 2.1 summarizes the control plane interface developed for each LIGHTNESS device and more details will be elaborated in the following section.

	Devices	**Agent development**		
		Language	Running environment	Device interface
OCS switch	Polatis N-VST-16x16	C	Linux	Ethernet port
Optical ToR	FinisarWaveshaper 4000	C	Linux and windows	USB
OPS	4x4 OPS	Java and C	windows	USB
NIC	FPGA	C	Linux	Ethernet port (SFP+)

Table 2.1: Summary of the interface between control plane and devices

3. Optical ToR and programmable NIC evaluation

3.1. Implementation

3.1.1. Optical ToR implementation

The proposed optical ToR based on NxM WSS is shown in Figure 3.1. While most of the DCN architectures in the state-of-the-art propose electronic ToRs, our DCN design introduces all-optical links from server-to-server without any O/E/O conversion in the middle stages, thus offering lower latency, higher energy and cost efficiency. In addition, another considerable advantage of having NxM WSS as switching elements instead of static AWGs, router-AWG or fibre switches, is the dynamicity and the flexibility in routing a range of frequency channels on any physical output port, combining both frequency and fibre switching. One more benefit of this WSS-based DCN design stems from the ability of the WSS to operate in Flex-grid, fine filtering desired spectrum slices. Then considering also that links inside DCs are shorter compared to metro/core networks and can tolerate lower Bit Error Rate (BER)-Optical Signal Noise Ratio (OSNR), the channels, i.e. 10Gb/s OOK, can be spaced much closer (20GHz instead of 50GHz that AWGs support). That results in increasing the spectral efficiency of the system by at least 150% while using low-cost technology. Furthermore, the WSS employed in our test bed supports bidirectional communication and can be reconfigured arbitrary (e.g. the 4X16 WSS can be used as 8X12 or 10X10 WSS as well). Thus, the proposed NxM WSS-based ToR design provides intra-rack communication between servers, as well as low-latency inter-rack connectivity with on-demand flexible bandwidth. Grand-bandwidth data exchanges can be served by the high-capacity DP-QPSK channels and/or a combination of the 10G channels.

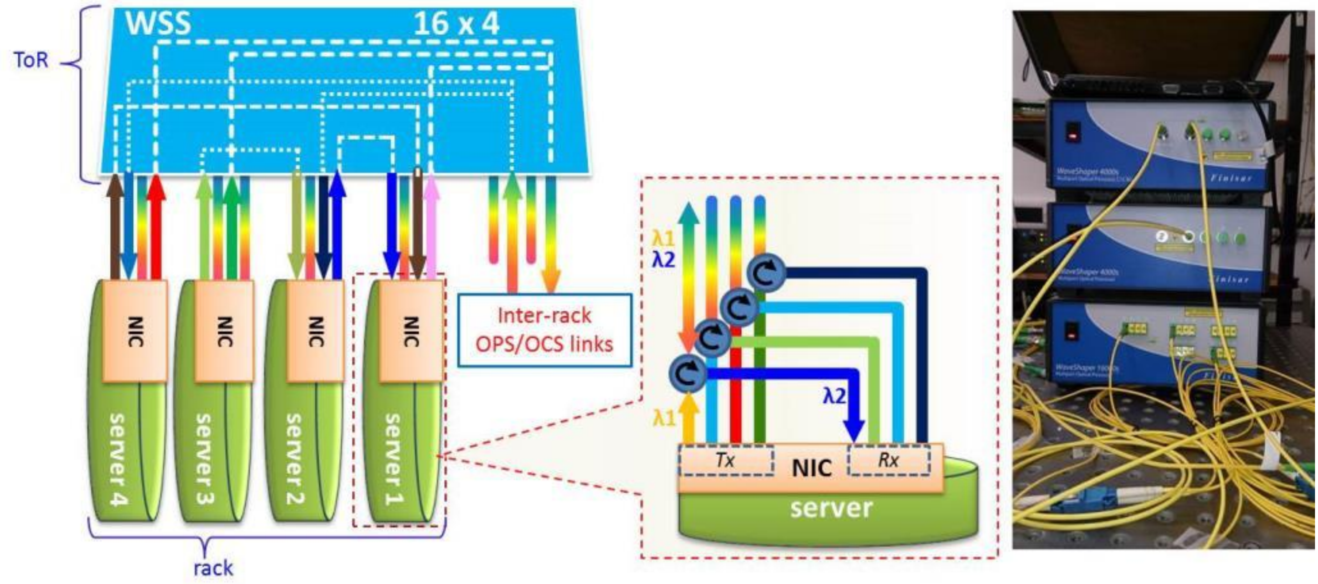


Figure 3.1: WSS-based Optical ToR implementation

3.1.2. Programmable NIC Implementation

The programmable NIC is proposed to replace the traditional NIC and plugged into each server directly. This flattens the DC infrastructure, enables all intra-rack optical server-to-server communications and eliminates the electronics in the ToR. The infrastructure with programmable NICs and pure optical ToR switch enables achieving high performance intra-rack evolving to inter-rack communication.

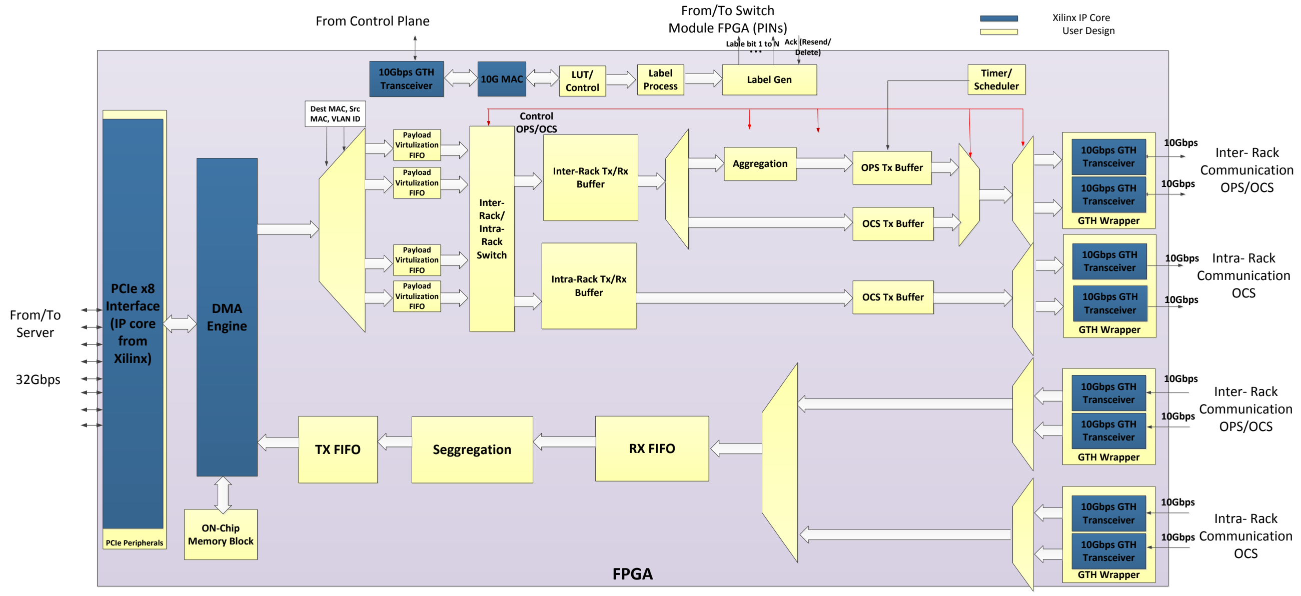


Figure 3.2: FPGA-based NIC Block Diagram

The programmable optical NIC is FPGA-based, designed and implemented in Hitech Global's HTG-V6HXT-X16PCIE-380. It features with Xilinx HX380T FPGA, SFP+ interfaces and Gen2 PCIe interface. The FPGA-based programmable NIC design and implementation block diagram is shown in Figure 3.2.

In Figure 3.2, the blocks represent variable functions. The blue ones are IP cores from Xilinx or third party and the yellow ones are designed and implemented by ourselves. The data flow follows the arrow in the block diagram. The FPGA-based programmable NIC fetches the data from the server RAM and can support 2 hybrid OPS/OCS ports for inter-rack communication and 2 OCS ports for intra-rack communication.

The functions of the programmable NIC are as below:

a) 10GE interface with OF agent for communication with SDN control plane

The OF-agent, which acts as a mediation entity with the SDN control plane, talks with the FPGA-based programmable NIC through 10GE interface. The FPGA-based programmable NIC receives the pre-defined Ethernet frame, parses the frame, get the useful information and stores the commands in the Look-Up Table(LUT). The FPGA-based block diagram is shown in Figure 3.3.

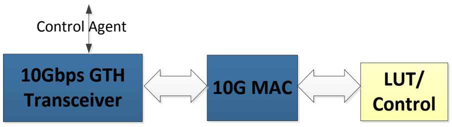
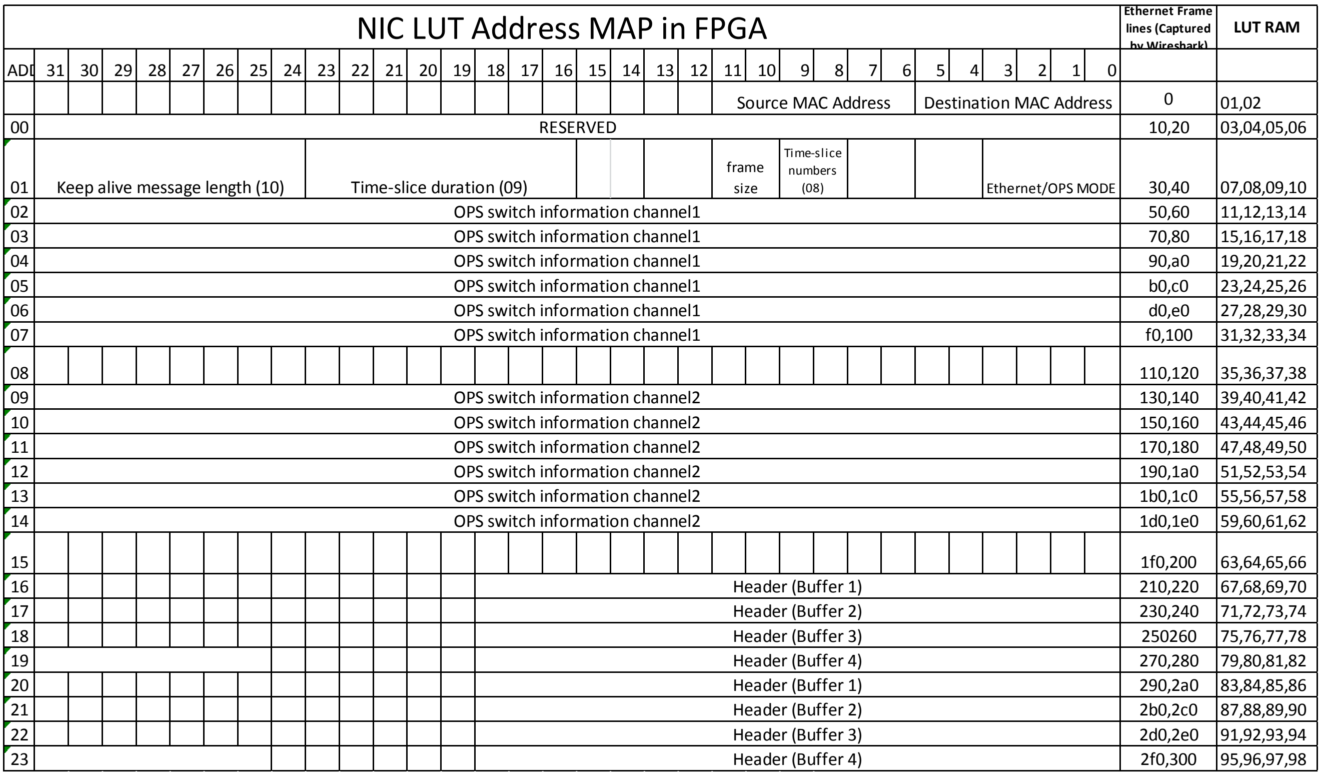


Figure 3.3: FPGA-based controller agent interface block diagram

The size of pre-defined Ethernet frame (with VLAN ID) is 1504 Byte, the node ID is defined as the first byte of destination MAC address. The programmable NIC supports push, pull, add, delete and modify functions. These functions are defined in the last byte of destination MAC address of the Ethernet frame. The details of the mapping information of the LUT and the Ethernet frame are shown in Figure 3.4.



NIC LUT Address MAP in FPGA																																	Ethernet Frame lines (Captured by Wireshark)	LUT RAM
ADI	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
																					Source MAC Address						Destination MAC Address						0	01,02
00	RESERVED																																10,20	03,04,05,06
01	Keep alive message length (10)								Time-slice duration (09)												frame size		Time-slice numbers (08)						Ethernet/OPS MODE				30,40	07,08,09,10
02	OPS switch information channel1																																50,60	11,12,13,14
03	OPS switch information channel1																																70,80	15,16,17,18
04	OPS switch information channel1																																90,a0	19,20,21,22
05	OPS switch information channel1																																b0,c0	23,24,25,26
06	OPS switch information channel1																																d0,e0	27,28,29,30
07	OPS switch information channel1																																f0,100	31,32,33,34
08																																	110,120	35,36,37,38
09	OPS switch information channel2																																130,140	39,40,41,42
10	OPS switch information channel2																																150,160	43,44,45,46
11	OPS switch information channel2																																170,180	47,48,49,50
12	OPS switch information channel2																																190,1a0	51,52,53,54
13	OPS switch information channel2																																1b0,1c0	55,56,57,58
14	OPS switch information channel2																																1d0,1e0	59,60,61,62
15																																	1f0,200	63,64,65,66
16														Header (Buffer 1)																			210,220	67,68,69,70
17														Header (Buffer 2)																			230,240	71,72,73,74
18														Header (Buffer 3)																			250260	75,76,77,78
19														Header (Buffer 4)																			270,280	79,80,81,82
20														Header (Buffer 1)																			290,2a0	83,84,85,86
21														Header (Buffer 2)																			2b0,2c0	87,88,89,90
22														Header (Buffer 3)																			2d0,2e0	91,92,93,94
23														Header (Buffer 4)																			2f0,300	95,96,97,98

Figure 3.4: NIC LUT address MAP in the FPGA

The LUT contains every single function that FPGA-based programmable NIC can support. The functions include fetching/storing data from/to the server RAM, OCS/OPS hitless switchover and the hitless switchover of the programmable parameters, such as time-slice duration and keep-alive message duration. These functions will be described later in this section.

b) PCIe Gen2 fetching/storing data from/to the server RAM

The PCI Express is based on point-to-point topology, with separate serial links connecting every device to the host. To get the maximum throughput, we used a Direct Memory Access (DMA) for copying large amounts of data efficiently between two devices with minimal host processor involvement. The advantage of using the DMA with PCIe is DMA Engine can issue large burst transactions and a repetitive task required only simple decisions, well suited for hardware acceleration. The FPGA-based block diagram of the PCIe with DMA is shown in Figure 3.5.

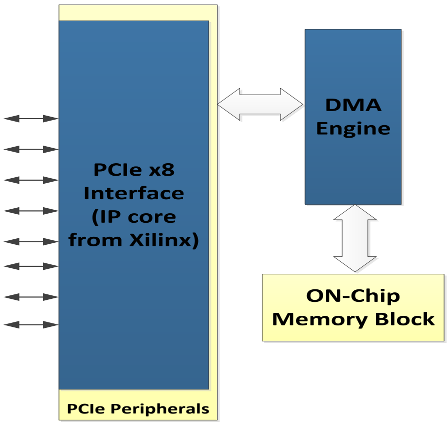


Figure 3.5: FPGA-based PCIe and DMA engine block diagram

Here the PCIe core is a hard IP core directly embedded in the FPGA board. The underlying communications mechanism consists of three layers: the Transaction Layer, the Data Link Layer, and the Physical Layer. The transmission of the packet which is interested for us is defined as a Transaction Layer Packet (TLP), which relates to the PCIe's uppermost layer. A flow control mechanism makes sure that a packet is sent only when the link partner is ready to receive it.

The DMA engine is the EZDMA2 IP core from PLDA. The PLDA EZDMA2 core can support up to 8 built-in DMA channels and up to 4GB DMA data transfer. With the 8 independent DMA channels that may be used simultaneously, it could manage up to 8 separate data flows.

The main module in the EZDMA2 core is the master request module and the master completion module. The master request module receives and transmits transfer requests to the transmit module. When one or more channel requests are detected, the Master Request Module masks requests for which there are no completion resources or not enough credits. It then selects the requests according to a round-robin priority scheme. Data is read from the local interface without any DMA channel interaction. Master Completion module receives Completions from the Receive module. It identifies the appropriate completion resource, DMA channel, and local address for the Completion and informs the requesting channel. Data is transmitted to the local interface without any DMA channel interaction. The Master Completion Module receives Completions from the Receive module and informs the requesting channel.

c) Destination MAC address, source MAC address and VLAN ID based traffic classification

The programmable FPGA-based NIC is capable of classify the traffic based on the destination MAC address, source MAC address and VLAN ID. The income traffic is filtered and stored in separate isolated FIFO, and will be aggregated or processed based on each FIFO. The FPGA-based traffic classification block diagram is shown in Figure 3.6.

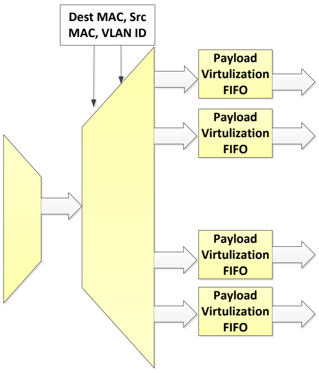

Dest MAC, Src MAC, VLAN ID
Payload Virtulization FIFO
Payload Virtulization FIFO
Payload Virtulization FIFO
Payload Virtulization FIFO

Figure 3.6: FPGA-based traffic classification block diagram

The implementation includes 4 FIFOs. The matching of the header includes destination MAC address (6 bytes), source MAC address (6 bytes) and VLAN ID (4 bytes). There is a separate mask for each bit of the header. The mask works on bit-based map, select the bits of header to match with the traffic. In the FPGA implementation, we defined the bit-based '1' means select, '0' means not. All the matching information is sent through the controller agent and the LUT updates the matching information accordingly. Therefore, the FPGA-based programmable NIC is capable of classify the traffic based on one single, or a set of header information, i.e., the last byte of destination MAC address 01 can be picked up and stored in FIFO1, the last byte of destination MAC address 02-07 can be groomed in FIFO2 and so on. This function supplies the most flexibility and availability for classifying the traffic.

d) OPS/OCS hitless switchover

A very important function of the FPGA-based programmable NIC is the OPS/OCS hitless switchover. There are two ports in the implementation supporting the hybrid OPS/OCS switchover. To get the hitless feature, when the FPGA-based programmable NIC receives the OPS/OCS switchover command from the controller agent, the OPS/OCS switchover functional block does not switch immediately; instead, it carries on with the current mode and clears the data in the transmitter buffer. After the transmitter buffer is empty, the mode is switched, and the NIC starts to transmit the traffic in the new mode.

e) Configurable parameters

For the OPS, the packet size and overhead is an important consideration in the experiment. These two parameters can be programmed on-the-fly in the FPGA-based programmable NIC by the controller agent.

When the data goes through the OPS switches, the packets are cut and forwarded to the path according to the label which causes the FPGA-based programmable NIC receives the traffic in burst mode. When there is no traffic received or the receiver receives the adjacent packets from different sources, the FPGA-based receiver lost the clock. Then, in the start of the packet, the FPGA-based receiver has to recover the clock based on the incoming traffic. Therefore, the overhead before the data is vital to avoid any data loss for the FPGA receiver recovering the clock. The duration of the overhead needed, just enough for recovering the clock, depends on the transmission environment, i.e. the number of the switches used, the loss of the optical devices and so on. When setting up the optical network with the FPGA-based programmable NIC, we can configure the overhead parameter on-demand.

After fixed the minimum overhead, the maximum bandwidth and the latency are relevant to the real data size in the packet. Based on the requirements of the different scenario, such as latency sensitive or bandwidth crucial, we can program the data size accordingly.

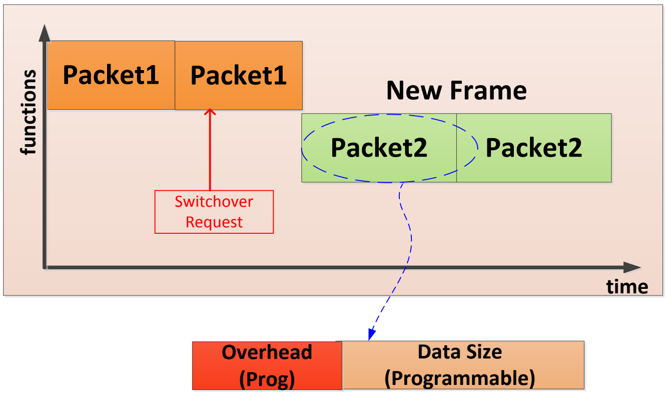


Figure 3.7: Packet size switchover when receives the request

Figure 3.7 shows that when the request of changing the overhead size or data size is received, the FPGA-based programmable NIC finishes the current packet and starts to transmit the packet with the new size parameters.

3.2. System evaluation

3.2.1. WSS-based optical ToR evaluation

The characteristics of WSS-based optical ToR by deploying Finisar Waveshaper 16000 are listed in Table 3.1. Furthermore, we measured the pass-through latency of the WSS (24ns) with the traffic analyzer.

Insertion Loss (incl. connectors)	6.5 dB (typ. 4.5 dB)
Polarization Dependent Loss (PDL)	0.7 dB (typ. 0.2 dB)
Filter Shape	Arbitrary
Filter Bandwidth	10 GHz to 5 THz
Center Frequency Setting Accuracy	± 2.5 GHz
Bandwidth Setting Accuracy	± 5 GHz

Settling Time	500 ms
Maximum Total Input Optical Power	+ 27 dBm
Max Optical Power per 50 GHz channel	+ 13 dBm
Latency pass-through (measured)	24ns

Table 3.1: Characteristics of the WSS-based ToR

3.2.2. FPGA-based programmable NIC evaluation

The FPGA-based programmable NIC is fully tested and evaluated based on the functions described in the previous section. The board we used for testing is Hitech Global Xilinx Virtex 6 PCIe board. The picture of the board is shown in Figure 3.8.

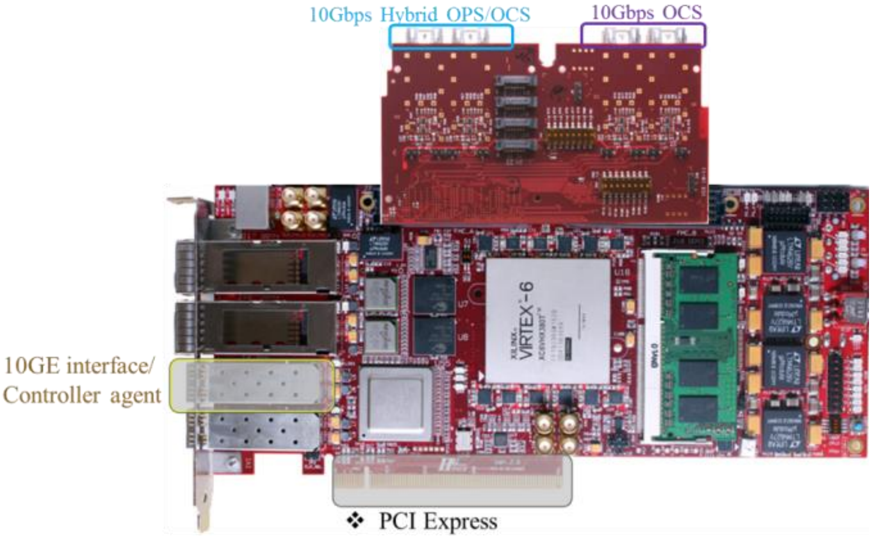


Figure 3.8: HiTech Global Xilinx Virtex 6 PCIE board

The PCIe interface on the board is Gen2 x16 lanes, we used x8 lanes for the implementation and testing because the DMA core we currently used only supports x8 lanes. We used 2 ports on the FPGA Mezzanine Card (FMC)-SFP+ daughter card as the hybrid OPS/OCS transceivers (inter-rack communication), and 2 ports on the FMC-SFP+ daughter card for the OCS only transceivers (intra-rack communication). Furthermore, we used one port on the board for communicating with the controller agent.

We did several experiments based on the PCIe/DMA only, PCIe/DMA with MAC and FIFO, and the programmable NIC. The experiment setup and test results are shown as below:

a) PCIe/DMA-only test

The first experiment is only based on the PCIe/DMA. The test setup is shown in the left side of Figure 3.9. The FPGA-based board is plugged into the server PCIe socket. In the FPGA, the design and implementation is shown in the right side of Figure 3.9. In this experiment, only dummy data are used for transmitting and receiving.

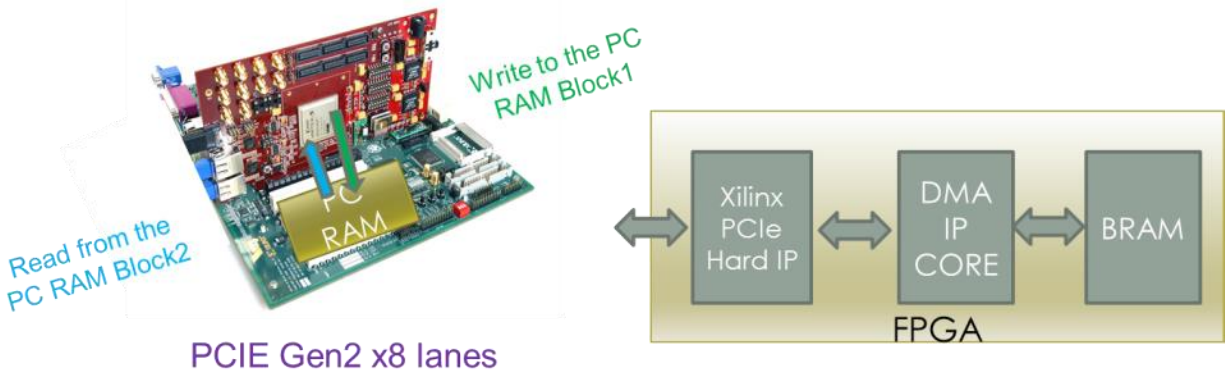
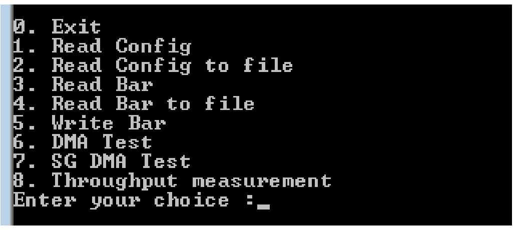


Figure 3.9: PCIe/DMA-only test setup and FPGA-based implementation

When write, the dummy data is sent through the DMA, PCIe and stored in the server RAM block 1.

When read, the dummy data is read from the server RAM block2, through the PCIe, DMA and stored in the BRAM of the FPGA.

The screen of the 8 functions the driver support is shown in Figure 3.10. The driver of the DMA engine allows to preset the size of the block data that to be transferred in one go.



```
0. Exit
1. Read Config
2. Read Config to file
3. Read Bar
4. Read Bar to file
5. Write Bar
6. DMA Test
7. SG DMA Test
8. Throughput measurement
Enter your choice :_
```

Figure 3.10: DMA engine driver script

We did the experiment based on 128KB block data.

The Maximum Throughput Measurement (128KB block data):

Read only (PC to NIC): Read Rate 12.8Gbps

Write only (NIC to PC): Write Rate 23.36Gbps

Read and Write: Read Rate 10.46Gbps Write Rate 21.36Gbps

When read/write through PCIe, the protocol is set to wait for the host (server) to reply. Therefore, the DMA performance is dependent on the latency of the platform, which is the time that the host takes to reply. For writing to the PC RAM, it only needs to wait the response and completion from the host once before writing the block of RAM. For reading from the PC RAM, it needs to wait the response from the host every TLP. So the read rate is lower than write rate.

b) Basic NIC, PCIe/DMA with MAC and FIFO, real Ethernet traffic

The second experiment setup is shown in the top of Figure 3.11. The FPGA-based implementation is shown in the bottom of Figure 3.11, which includes GTH (transceiver in the FPGA), 10GE MAC, FIFOs, DMA and PCIe.

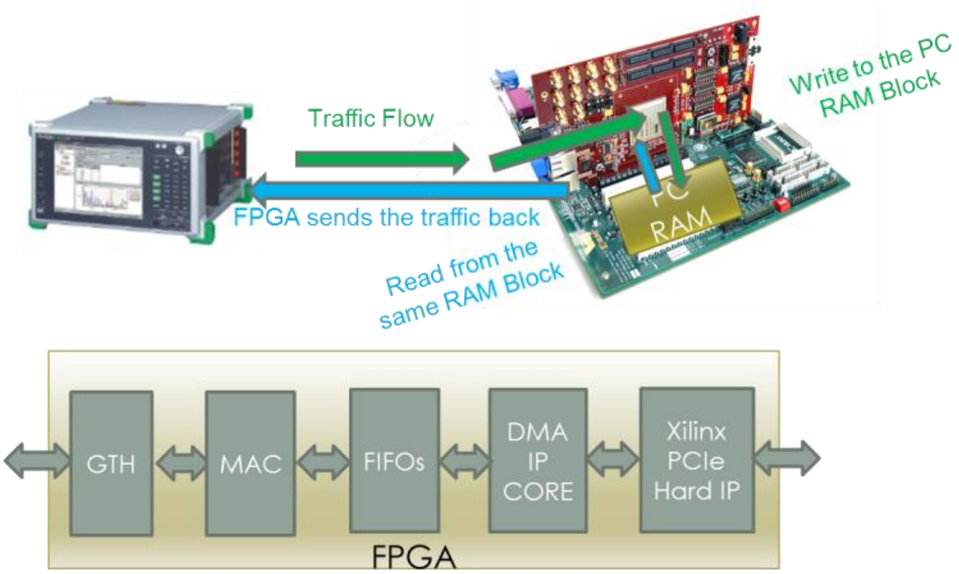

Traffic Flow
Write to the PC RAM Block
FPGA sends the traffic back
PC RAM
Read from the same RAM Block
GTH
MAC
FIFOs
DMA IP CORE
Xilinx PCIe Hard IP
FPGA

Figure 3.11: Basic NIC, PCIe/DMA with MC and FIFO setup and FPGA-based implementation

The traffic is generated by the traffic generator and sent to the FPGA through 10Gbps link, the 10GE MAC processed the data, buffered them, and stored in the server RAM through DMA/PCIe. Then these data are read back from the same RAM block and moved back to the FPGA FIFO by DMA/PCIe. After processed by MAC, these data are sent out as Ethernet frames and received by traffic analyzer.

By configuring the DMA transferring data size (assigned RAM block size), we can measure the maximum throughput (1500B Ethernet Frame) that the basic NIC can deliver without any data loss. The result chart is shown in Figure 3.12**Error! Reference source not found.**.

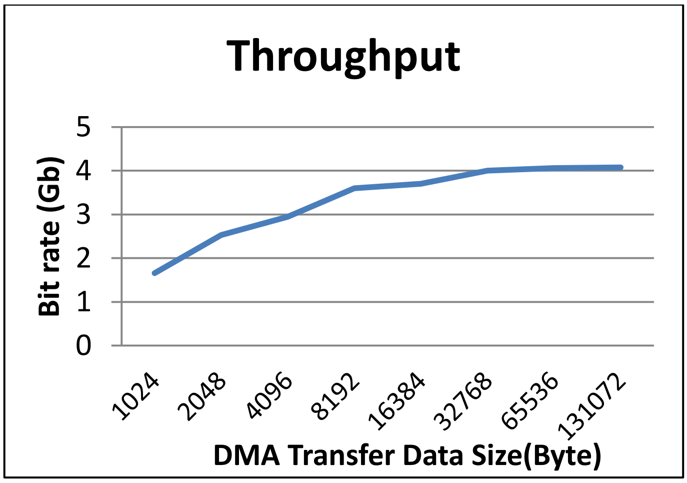


Figure 3.12: Basic NIC throughput by RAM block

When the RAM block size goes larger, the more data is transferred in one-go. The maximum throughput in this experiment is around 4Gbps. There are two factors that affect the limited throughput here. One is the Read and Write process works on the same RAM block in the PC, the read only happens after write, not simultaneously, so the throughput is limited. The other one is because the implementation in the FPGA caused around 30% data overhead when moving the traffic to the server RAM.

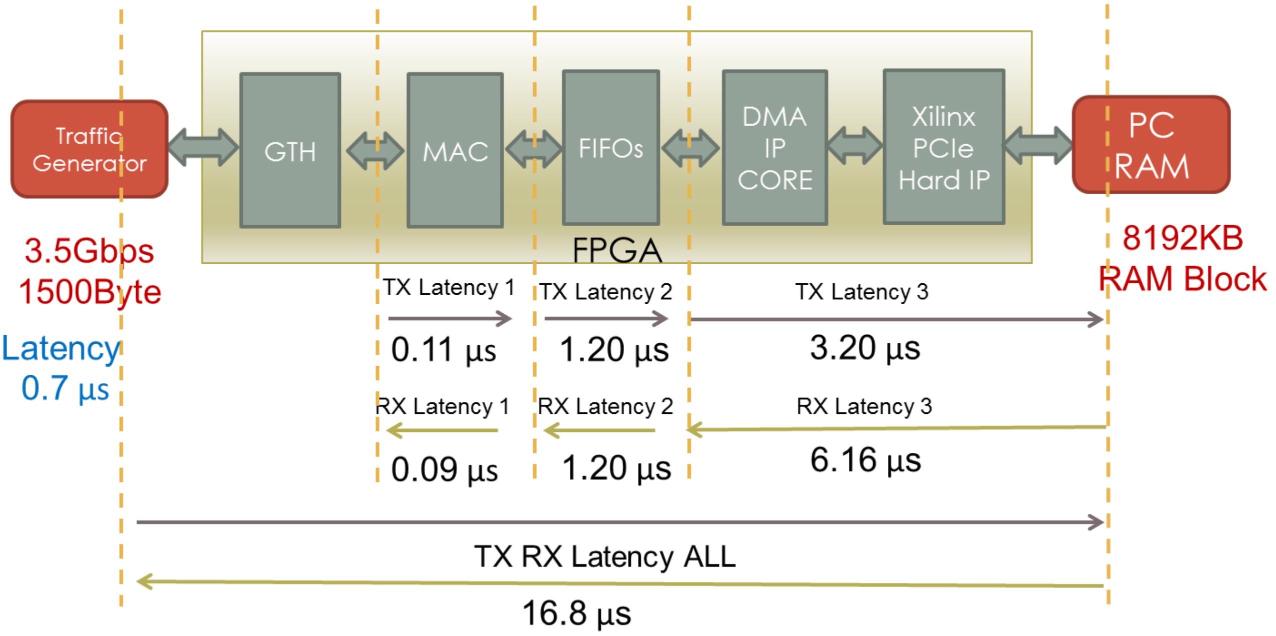


Figure 3.13: Basic NIC latency result

To get the latency result, we compromise the traffic and RAM block size by configuring 8192KB RAM block size, and set the traffic as 3.5Gbps. The detailed latency is shown in Figure 3.13.

From Figure 3.13, we can see the latency is mainly caused by the DMA/PCIe moving data between FPGA and PC RAM. The TX latency is 3.2µs and the RX latency is 6.16µs. The FIFO latency is because the implementation of the basic NIC needs to wait for a complete Ethernet frame then send out. The total latency for the basic NIC to receive a 1500B Ethernet Frame, put it in the PC RAM, read it out, processed in the FPGA, and then sent back to the traffic analyzer is 16.8µs.

c) FPGA-based programmable NIC test

The FPGA-based programmable NIC is experimentally tested. The experiment setup is shown in Figure 3.14. The FPGA-based programmable NIC is connected to the server socket through PCIe extension cable. The controller agent interface of the FPGA-based programmable NIC is connected with the server through SFP+ and fibre. For testing, the traffic generator generates the Ethernet traffic and feeds directly to the hybrid OPS/OCS port. The hybrid port 1 is set to OCS mode. Therefore, after receiving the Ethernet traffic, the FPGA-based NIC processed the data, and sent them through DMA/PCIe to the server RAM. After the transmission, the DMA engine initiates the transmission from the server RAM back to the FPGA. The FPGA-based NIC then do the processing in the OCS mode and sends the traffic out to the traffic analyzer which measures the latency.

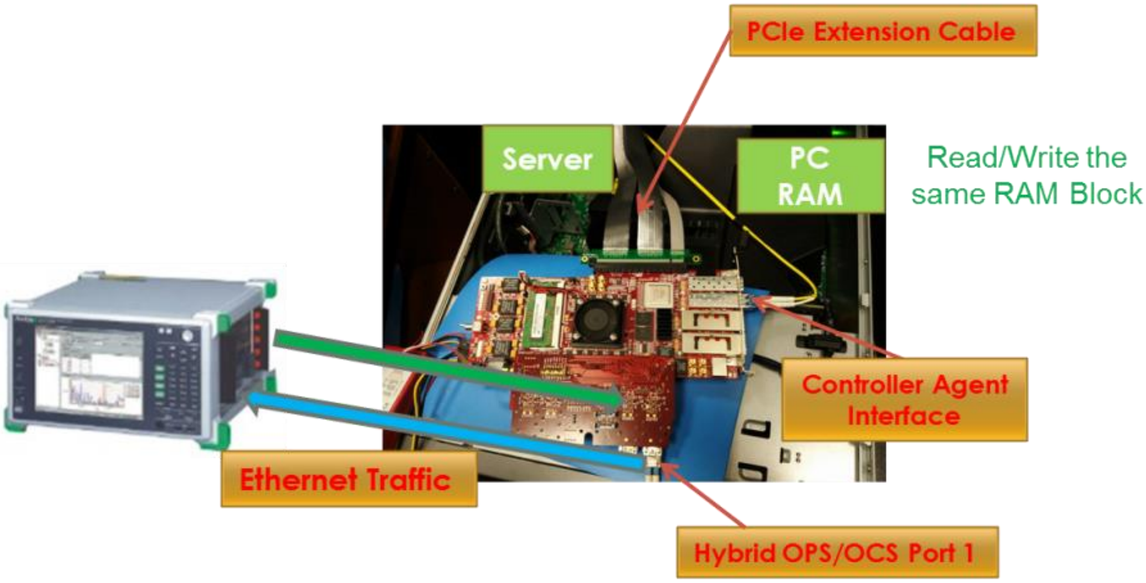


Figure 3.14: FPGA-based programmable NIC experiment setup

In the experiment we measured the maximum throughput and the latency. The maximum throughput measurement result is shown in Figure 3.15. For the throughput measurement, the data is written to the RAM, and after the block of RAM is filled, the data is read back. The maximum throughput is limited because of this non-duplex transmission.

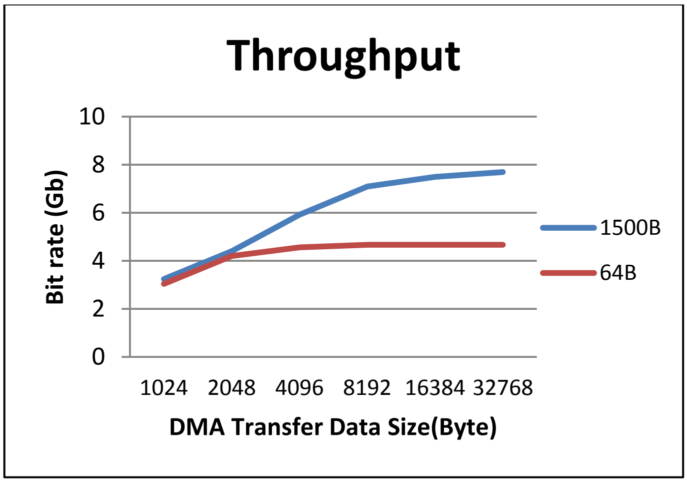


Figure 3.15: FPGA-based programmable NIC maximum throughput measurement result

The maximum throughput for 1500B Ethernet frame is around 8Gbps and for 64B Ethernet frame is around 4.7Gbps. The difference of the maximum throughput that the FPGA-based programmable NIC can achieve between 1500B and 64B is because the header and tailor of the frames need to be written into the RAM to get the correct data out, and for 64B, the ratio of the real data and the whole frame with header and tailor is much lower than 1500B. Therefore, the throughput performance for 64B Ethernet frame is worse than 1500B Ethernet frame. This throughput could be improved by sending the data directly from the server.

We also measured the latency with different DMA transfer data size and Ethernet frame size. The measurement results are shown in Figure 3.16 and Figure 3.17.

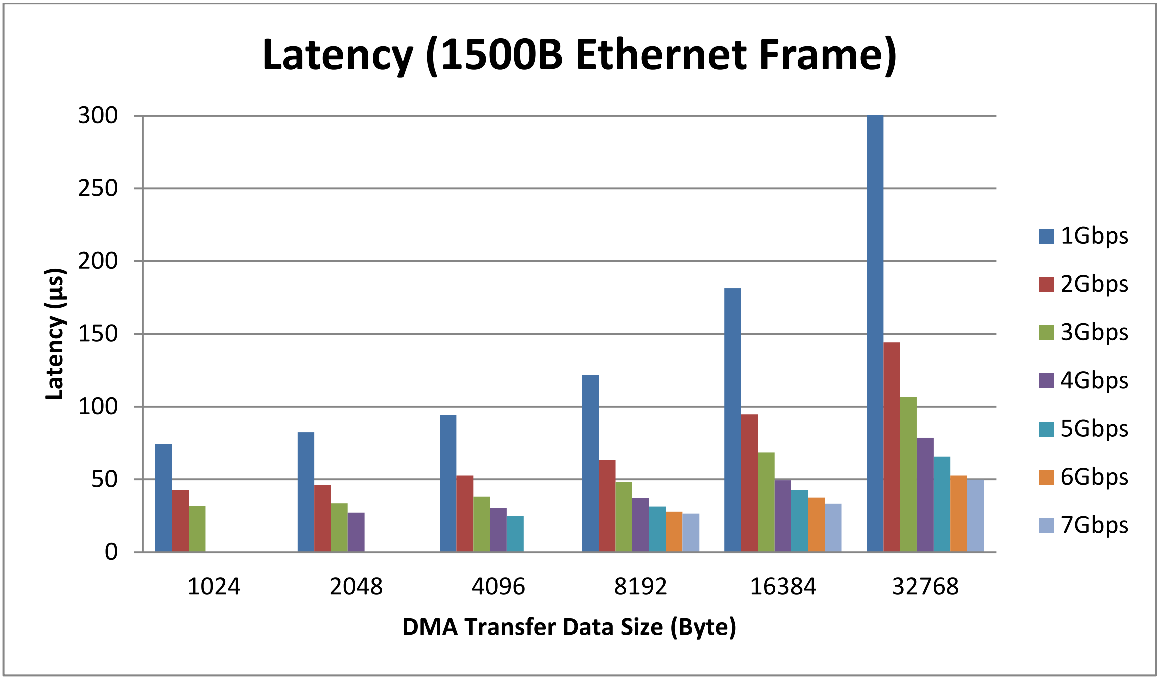


Figure 3.16: FPGA-based programmable NIC maximum throughput measurement result (1500B)

Figure 3.16 shows the measured latency result of the traffic of 1500B Ethernet frame. The latency increases when the DMA transfer data size increases. The minimum latency we can get is when transmitting 3Gbps traffic with 1024Byte DMA transfer data size. By increasing the DMA transfer data size, the throughput increases but the latency gets longer.

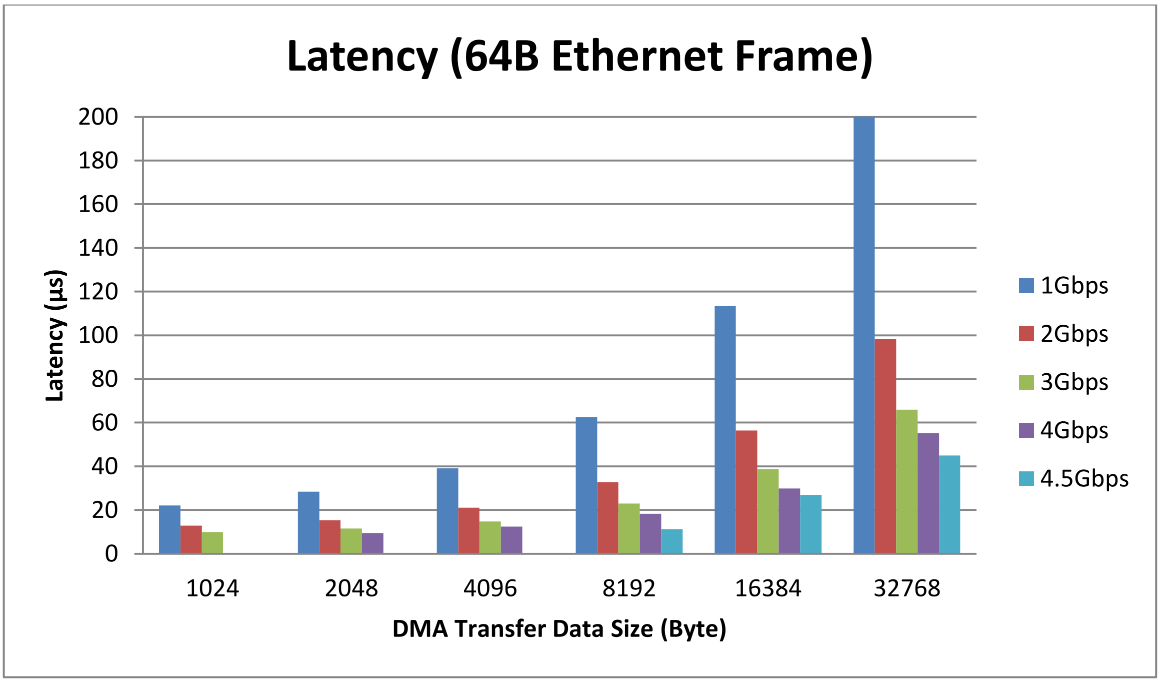


Figure 3.17: FPGA-based programmable NIC maximum throughput measurement result (1500B)

Figure 3.17 shows the measured latency result of the traffic of 64B Ethernet frame. Similar to the measured results of 1500B Ethernet frame, the measured result shows the latency increases when the DMA transfer data size increases. The minimum latency measured is 9.417µs when DMA transfer data size is 2048 Bytes, and the bit rate of the traffic is 4Gbps.

Comparing Figure 3.16 and Figure 3.17, the traffic of 64B Ethernet frame has better latency performance than the traffic of 1500B Ethernet frame. In the FPGA-based traffic processing, 1500B Ethernet frame takes longer time than 64B Ethernet frame.

Lastly, we measured the FPGA-based programmable NIC performance when switchover among the programmable parameters and OCS/OPS switchover.

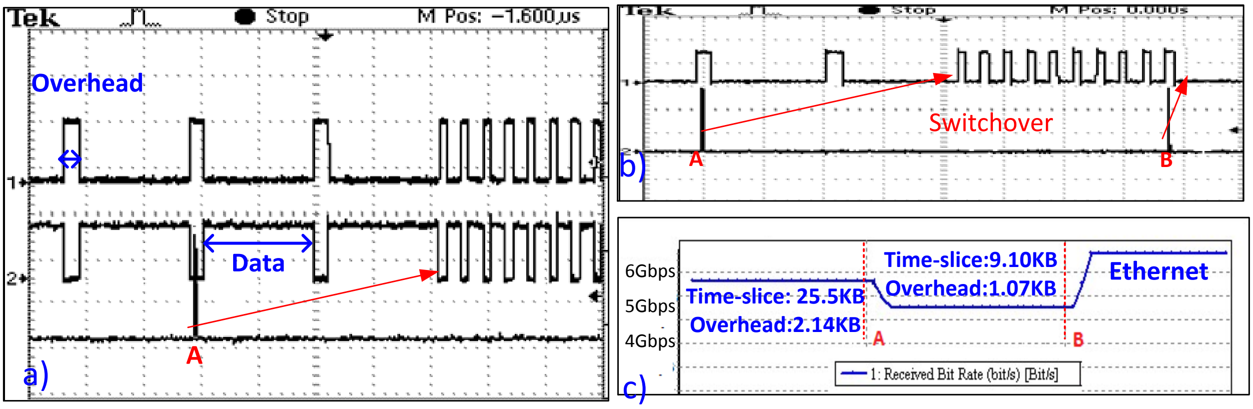
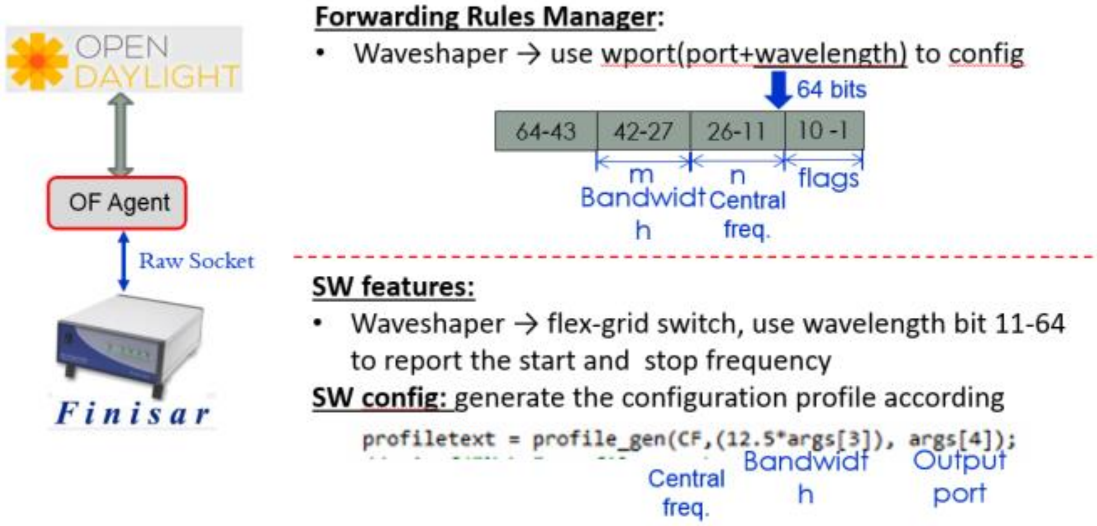


Figure 3.18: Programmable parameters test results

Figure 3.18(a) shows the hitless transmission after switchover, when switch from OPS to OCS, the inter-function switch blocks wait for the current aggregated time-slice finished, then starts to transmit the Ethernet. In opposite, while switching from OCS to OPS, the FPGA waits finishing processing current Ethernet Frame, and then starts to transmit as packet-based. The programmable parameters switchovers are demonstrated in Figure 3.18 (b)and (c), such as overhead (programmable from 0 to 39KB) and packet data size (programmable from 80B to 31.25KB) parameters changed from one to the other which shows in Figure 3.18(c) that the maximum bandwidth changed correspondingly.

3.3. Interface with control plane

3.3.1. Interface with control plane for optical ToR



Forwarding Rules Manager:

- Waveshaper → use wport(port+wavelength) to config

SW features:

- Waveshaper → flex-grid switch, use wavelength bit 11-64 to report the start and stop frequency

SW config: generate the configuration profile according

```
profiletext = profile_gen(CF,(12.5*args[3]), args[4]);
```

Figure 3.19: WSS-based Optical ToR’s interface with control plane

The OF agent sits between the SDN controller and WSS-based optical ToR and includes two software tools: one is running on a Linux server which directly communicates with the SDN controller in the northbound and with a windows proxy (where WSS driver are installed) in the southbound. The interaction between these two parts is implemented through socket.

The OF agent (windows proxy) communicates with WSS through Ethernet interface (management interface) and messages are transmitted in raw socket. For the feature advertisement, except the flex-grid switch capability, the agent also reports the supported frequency range (with start and stop frequency) and port count, fetched from device (using the management API command). For configuration purpose, after the agent receives the configuration message (OF cflow_mod as shown in Figure 3.19) from the controller, it uses the central frequency/bandwidth and the output port to generate a configuration profile according to the specification defined by the WSS. This profile will be loaded by WSS to implement the channel configuration. The channel release follows the same approach.

3.3.2. Interface with control plane for hybrid NIC

The NIC Agent is implemented in C language. It sends and receives OpenFlow v1.0 messages to the SDN controller and uses a raw Ethernet socket to send frames to and from the NIC FPGA. These frames either set part or all of the NICs internal LUT or request that the NIC transmits the state of the LUT to the agent. The device has two operating modes: OCS and Ethernet switch. The desired mode is inferred by the agent when a specific kind of FLOW_MOD request is received from the controller. Although each register of the FPGA’s LUT can be programmed and inspected individually, the agent sets or retrieves the entire 864-byte table with each set and gets command. Detailed information will be reported in deliverable D4.7 where the implementation of the OF agents are presented.

4. OCS switch evaluation

4.1. Implementation and System Evaluation

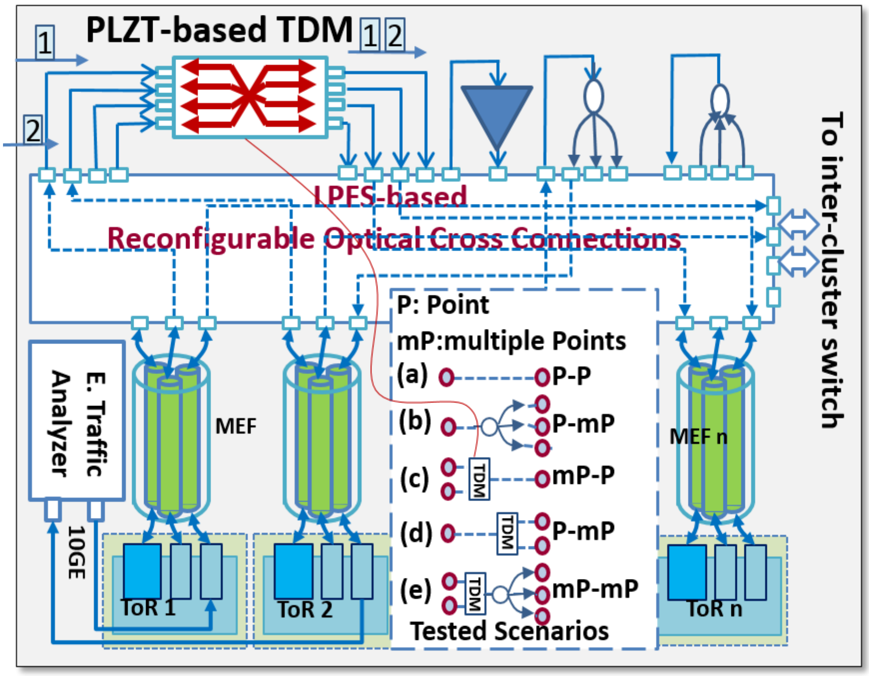


Figure 4.1: Experimental Setup of intra-cluster OCS network

Figure 4.1 shows the experimental setup of the intra-cluster network. SMFs or 3-element multi-element fibres (MEFs) are used to connect all the transmitters and receivers from FPGA-based ToRs to a 192×192 fibre switch (Polatis) [6]. The 4×4 PLZT-based TDM switch, SDM/WDM converter, WSS, EDFAs, couplers and splitters are already connected to the fibre switch for network function programmability. By reconfiguring the connection matrix of the LPFS in each cluster and the inter-cluster switch, a single hop OCS is achieved to serve the long-lived, large capacity data flows both for intra- and inter-cluster communication. The OCS capacity can be reconfigured according to the traffic request, by configuring the BVTs in the ToRs. In addition to the OCS, the re-configurability of the LPFS introduces network function programmability in DCNs. According to the traffic requests, the interconnections between the connected components, such as optical splitters, optical couplers, and EDFAs, can be connected to the OCS network, to obtain variable network functions, e.g. time aggregation, broadcasting and time de-multiplexing.

The fast PLZT-based TDM switch provides sub-wavelength switching with short reconfiguration time in order to serve short-lived low-capacity data flows for intra-cluster communication. The TDM link capacity can be varied from 100Mbit/s to 5Gbit/s by configuring the length of the time slot used. As the TDM switch is synchronized with all the ToRs in the cluster, more functions, such as time aggregation and time de-multiplexing, can be achieved without an optical buffer. The TDM switch is connected to the LPFS, enabling more network flexibility.

For intra-cluster communication, OCS-based SDM (Ethernet) and TDM technologies are used to provide a range of connections of bandwidth and capacity services. The single-hop SDM with an element capacity of 10Gbit/s is realized by reconfiguring the interconnection of the LPFS to support high throughput Ethernet transport. A synchronized 4×4 PLZT TDM switch is connected to the LPFS to realize TDM connections with variable capacity from 0.1 to 5Gbit/s. To avoid network contention in TDM networks, the transmission requests are scheduled by the path computation engine (PCE) in the control plane [9]. Another FPGA-based PLZT TDM switch agent will control the TDM switch. Optical components, such as couplers and splitters are connected to the LPFS, and can be programmed to support aggregation, broadcasting and other network functionalities, as in the examples shown in the inset of Figure 4.1.

For the optical BVT link, the traffic is also aggregated and sent to the optical BVT to generate optical signals with variable capacities for inter-DCN or inter-cluster communication. Figure 4.2 shows the experimental setup of the FPGA-based ToR with an optical BVT. In the setup of the optical BVT, an electrical multiplexer multiplexes the four 10 Gbaud input signals to a 40 Gbaud electrical signal, which is used to drive an IQ modulator to achieve 40 Gbaud QPSK signals. An optical emulator is followed to generate 40 Gbaud 16QAM signals. The eye diagrams of the generated 40 Gbaud QPSK and 40 Gbaud 16QAM signals are shown in Figure 4.3 (a) and (c). The recovered constellation diagrams are shown in Figure 4.3 (b) for QPSK signals and in (d) for 16QAM signals.

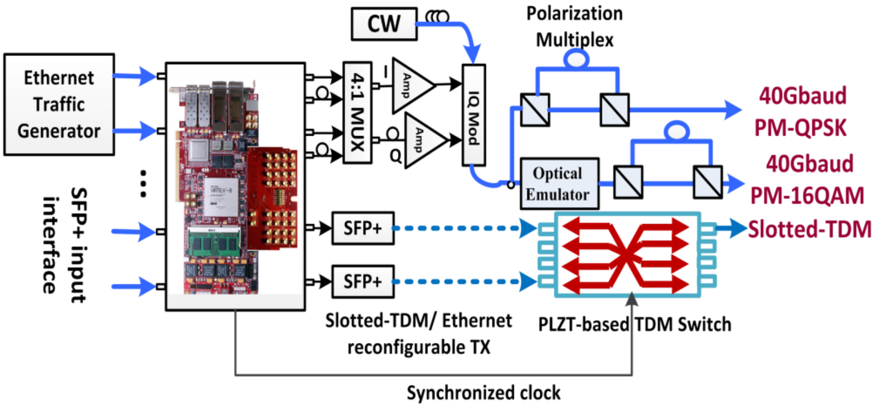


Figure 4.2: Experimental setup of the FPGA-based ToR

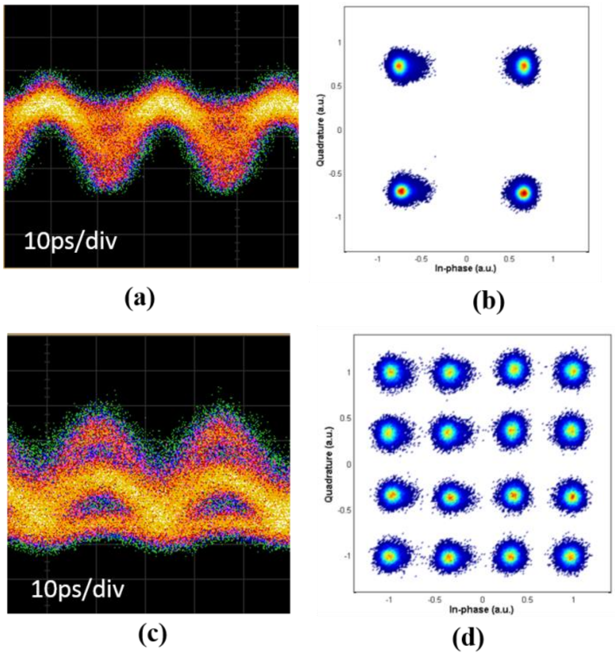


Figure 4.3: Eye diagrams of the generated 40Gbaud QPSK signal (a) and 40 Gbaud 16QAM signal (c). The corresponding recovered constellations are shown in (b) and (d)

Based on the link capacity request, the FGPA-based ToR can provide optical links with a capacity from 100Mbit/s to 5Gbit/s over a slotted-TDM transmitter or a 10Gbit/s Ethernet-over-SDM transmitter, or 160Gbit/s or 320 Gbit/s over an optical BVT. All the transmitters use fixed wavelength lasers to avoid expensive temperature control.

To validate the proposed design, several scenarios are designed to reflect the main network operation in DCNs, as listed in Figure 4.1. An Ethernet traffic analyzer is used to generate Ethernet traffic between servers. In scenario (a), the latency of the point to point (P-P) transmission is measured to be about 70 µs, where the FPGA-based ToR contributes about 4.2 µs (5% of the link latency). The maximum Ethernet capacity/port is about 9.8 Gbit/s. Figure 4.4 presents the results for both SDM-Ethernet network in both scenarios (a, b) and TDM network in scenarios (c, d, e). In the Ethernet-based OCS network, the broadcast operation introduces a power penalty of about 1.3 dB at a BER of 1E-9 due to the noise introduced by the EDFA used to compensate the loss of the splitter. For TDM-based intra-cluster communication, the scenario (e) contains the main network operations of other scenarios. So we test the network performance in scenario (e). Two time-slot data flows (2.5Gbit/s) from two ToRs are aggregated to 5Gbit/s and further broadcasted to three other ToRs. The PLZT-based TDM switch introduces a power penalty of about 2.1 dB at a BER of 1E-9 due to its cross talk (20dB). The received TDM capacity for different received optical powers after broadcasting is shown in Figure 4.5.

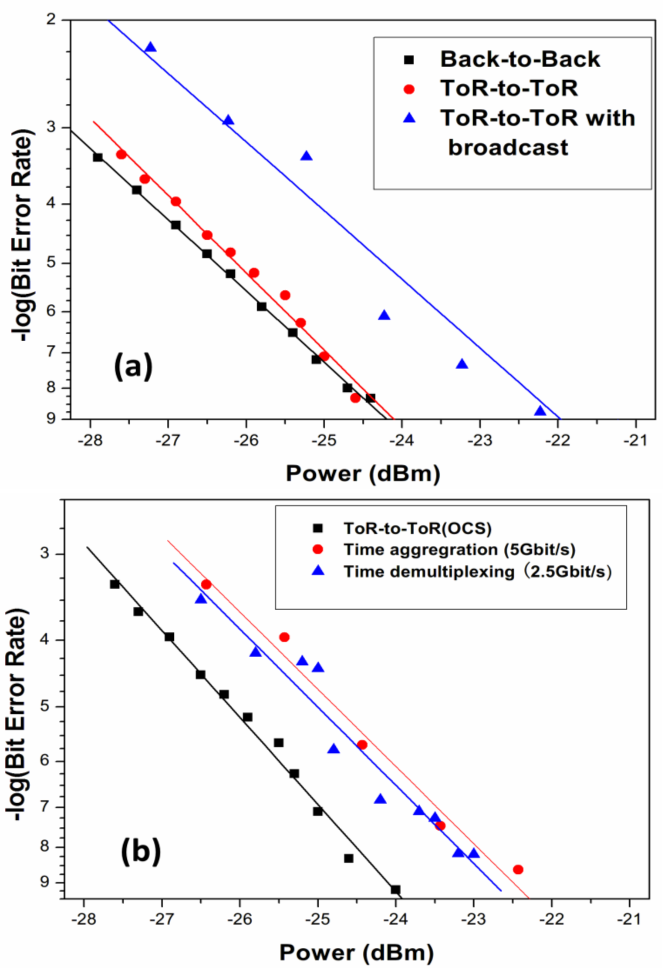


Figure 4.4: Experimental results of intra-cluster network: (a) Ethernet-based OCS network; (b) TDM network.

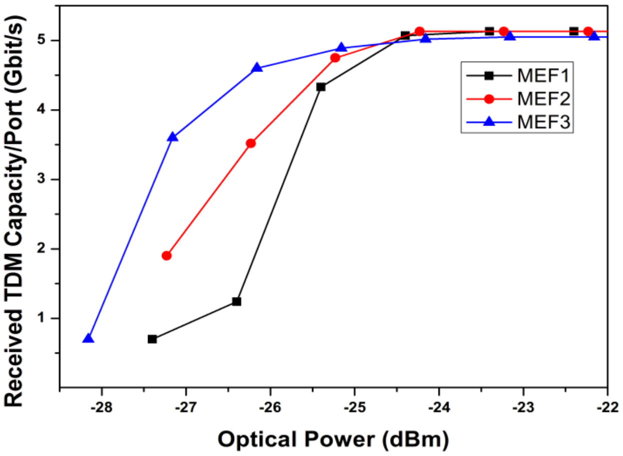
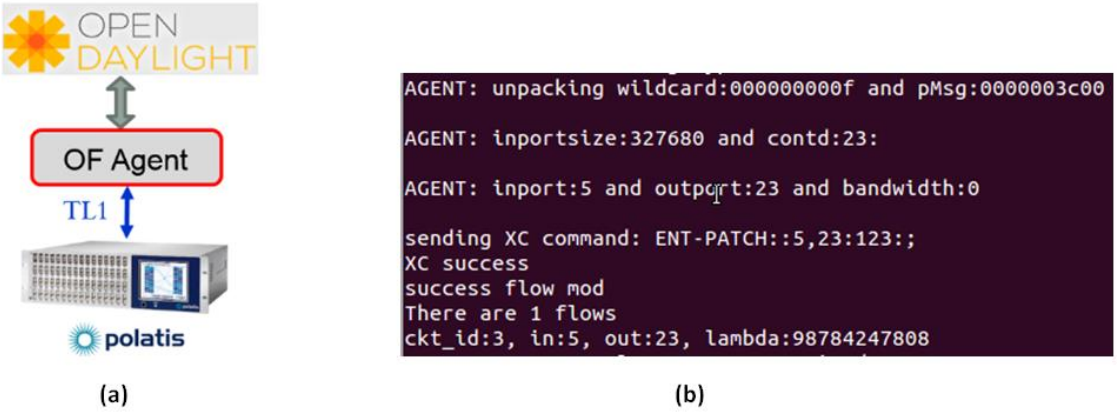


Figure 4.5: Received TDM capacity for different received optical powers after broadcasting for scenario (e)

4.2. Interface with control plane



```
AGENT: unpacking wildcard:000000000f and pMsg:0000003c00

AGENT: inportsize:327680 and contd:23:

AGENT: inport:5 and outport:23 and bandwidth:0

sending XC command: ENT-PATCH::5,23:123:;
XC success
success flow mod
There are 1 flows
ckt_id:3, in:5, out:23, lambda:98784247808
```

Figure 4.6: Port switch based OCS switch's interface with control plane: (a) system architecture (b) OF agent capture

As introduced in section 2.2, OF agent for the OCS Polatis switch is also implemented in Linux server with C language. The connection between the agent server and the device is implemented via Ethernet. For the feature advertisement, the agent fetches the port number and the port status from the switch with the TL1 command, and reports the optical port switching capability and the port number in switch features reply. For configuration purpose, after the agent receives the switch configuration message, agent will convert it to TL1 command through Polatis's management interface. More detailed information on the OF agent and southbound interface implementation can be found in D4.4 [17].

5. OPS switch evaluation

5.1. Implementation

The inter-rack traffic is handled by OCS and OPS nodes. Specifically, the short-lived traffic is grouped in packetized data and processed by the OPS. To minimize the latency of delivering these bursty traffic flows, a modular OPS architecture with highly distributed control has been introduced [2]. As can be seen in Figure 5.1(a), the OPS implements the efficient optical flow control without occupying extra bandwidth and space resource, leading to an improved performance in terms of latency within a simple structure [10]. The parallel processing of the different channels as well as the label bits allows for sub-μs latency while connecting a large number of ToRs in a flattened network. In addition, the optical packets could be switched transparently with multiple modulation formats and multiple wavelengths to deal with data-intensive applications [11]. More detailed information about the OPS design can be found in D3.1 [2].

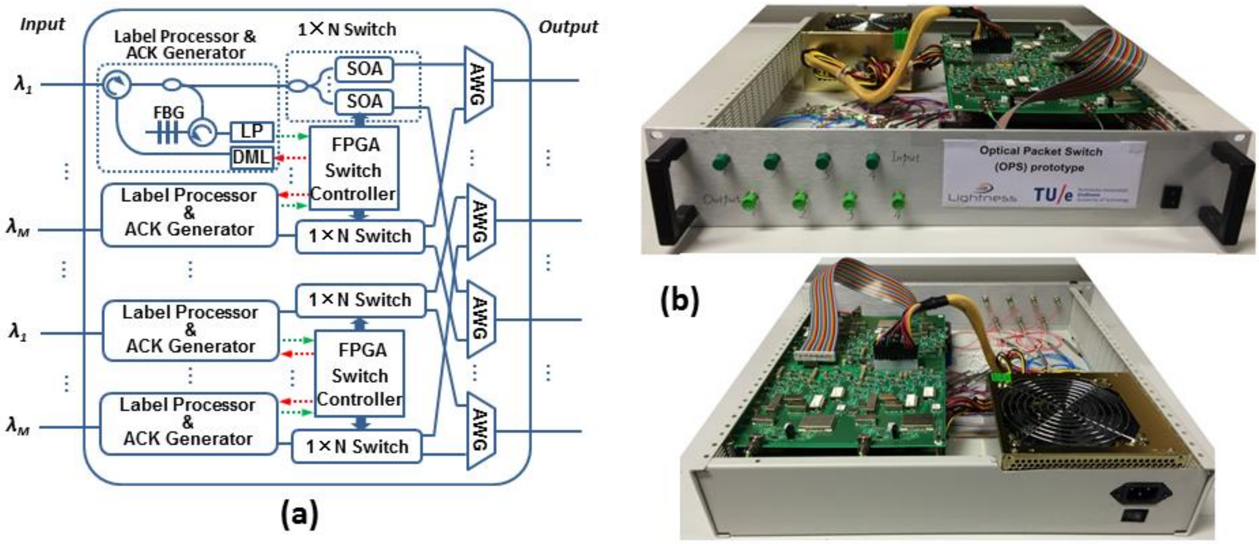

Input
Output
Label Processor & ACK Generator
1×N Switch
SOA
SOA
λ1
FBG
LP
DML
FPGA Switch Controller
Label Processor & ACK Generator
λM
1×N Switch
AWG
AWG
AWG
AWG
1×N Switch
Label Processor & ACK Generator
λ1
FPGA Switch Controller
Label Processor & ACK Generator
λM
1×N Switch
(a)
Input
Output
Optical Packet Switch (OPS) prototype
Lightness
TU/e
(b)

Figure 5.1: (a) Modular OPS architecture with distributed control. (b) OPS prototype

Based on the design scheme and preliminary implementation results, the OPS prototype has been finalized. Figure 5.1(b) shows the photos of the final prototype of the OPS. The label processor detects the optical label signal and processes it into digital label bits. The FPGA is the switch controller which receives the label information and resolves the contention. It provides the control signal to semiconductor optical amplifier

(SOA) driver to forward or block the packets. Moreover, the FPGA is responsible of communicating with the SDN controller to update the stored LUT and report the collected statistics. Passive optical devices (circulators, couplers, Fibre Bragg Gratings (FBGs), etc.) are arranged in a separate space.

RF tones labeling technique has been utilized to efficiently transmit the label signal and maintain the instant processing which significantly reduces the latency. In the OPS prototype, the label signal is processed by the label processor which quite values on RF electronics and requires fine design. Following the preliminary designs of the single label processor and label generator module that have been introduced in D3.2 [3], here we present the design of the label processor in the OPS prototype.

Compared with the single module reported in D3.2, the label processor in the prototype has more completed functions. First, it groups all the four processing modules on the same PCB board. Secondly, it also includes the optical-to-electrical (O/E) interfaces that the extracted optical label can be plugged into the photodiode (PD) on the board. Moreover, the detected baseband analog label signals will be further digitalized so that the output of the label processor can be directly interfaced to the FPGA switch controller.

The fabricated label processor PCB board is shown in Figure 5.2. It is capable of processing in parallel 4 optical label signals each carrying an 8-bit RF tones label. As an example, the Module 1 has been highlighted and the functional blocks of one bit process have been depicted. The PD will first convert the optical RF tones label into electrical signal. Then a band pass filter (BPF) separates each tone from the others and the envelop detector (ED) demodulates the base band label bit. At the shaping stage which consists of an amplifier and a compactor, the analog signal will be digitalized and outputted to the interface with the FPGA. The compact design greatly simplifies the interfaces and saves the needed space.

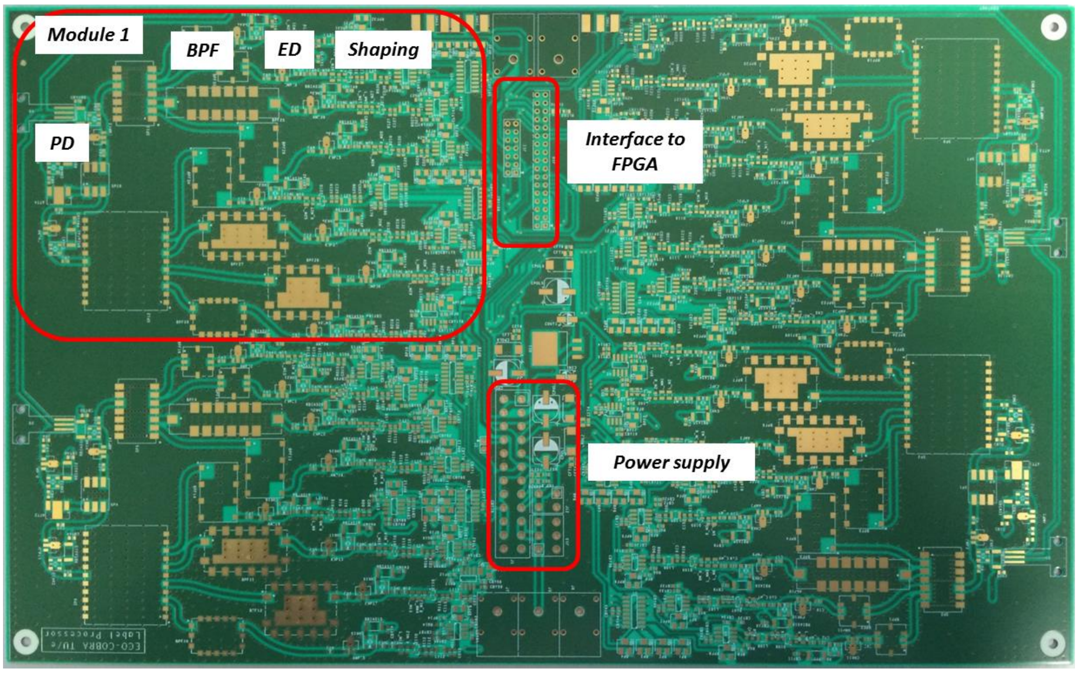


Figure 5.2: PCB board of the label processor

After the label bits are detected, the FPGA performs the contention resolution and controls the SOA gates. For a 4x4 OPS, 8 SOAs are needed to compose four 1x2 switches. They are placed on the SOA driver board which provides the coarse temperature control and the driving current controlled by the FPGA switch controller. The fabricated SOA driver board is shown in Figure 5.3. Besides the 8 SOAs, there are 4 directly modulated lasers

(DMLs) to generate the flow control signal. As the wavelength of the DML should match with the WDM channel, fine temperature control is provided to stabilize the output wavelength. Based on the label processor and SOA driver boards, the OPS prototype will be assembled and utilized in future experimental validation task.

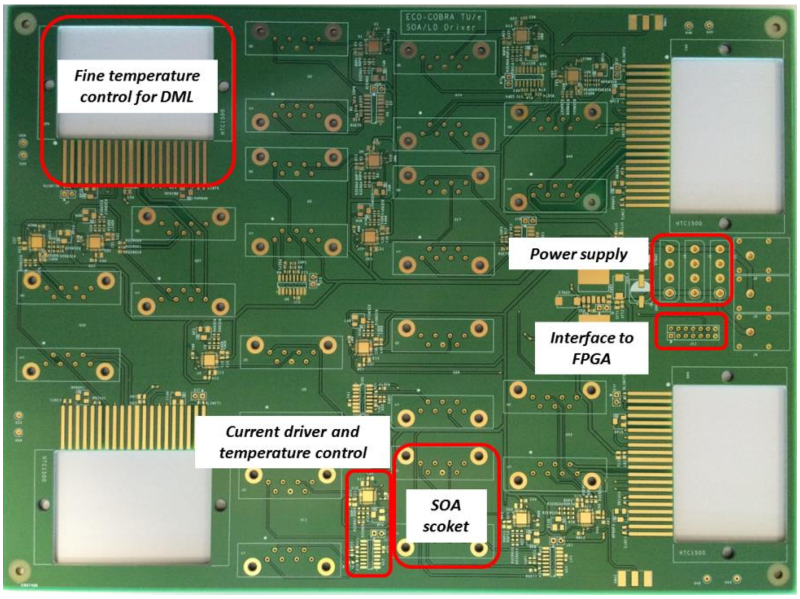

Fine temperature control for DML
Power supply
Interface to FPGA
Current driver and temperature control
SOA scoket

Figure 5.3: PCB board of the SOA driver

Figure 5.4 presents more detailed information on the implementation of the final OPS prototype.

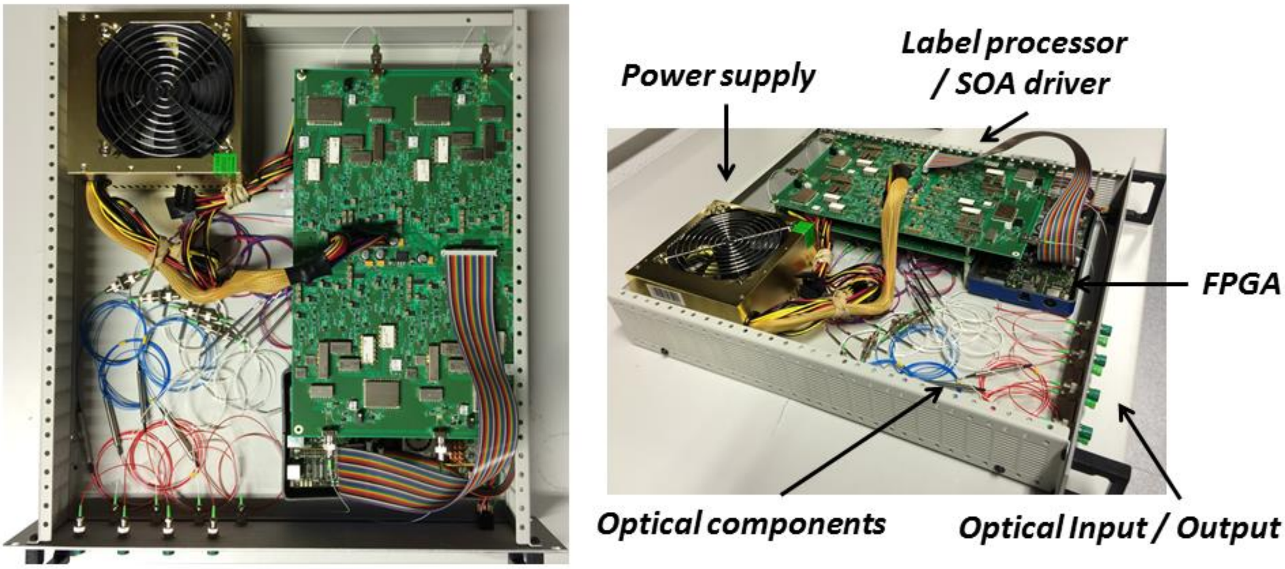


Figure 5.4: Detailed implementation schematic for OPS prototype

5.2. System evaluation

5.2.1. Performance assessment with burst-mode receiver

Although DCN is a closed environment with more controlled optical power variation, the receiver should handle packets with different length ranging from sub-microseconds to tens of microseconds, moderate different optical power levels, phase synchronization, and clock. Moreover, signal impairments due to the SOA switches, such as pattern dependent amplification and OSNR degradation, may affect the performance of the receiver. Thus, for practical implementation, packet-based networks need burst mode receivers (BM-RXs).

Typical BM-RX includes several functions such as fast automatic gain control (AGC) and decision threshold extraction, clock and phase synchronization. Each of those functions contributes, with a different overhead, to the overall BM-RX operational time. This time determines the minimum length of the preamble and the packet guard-time to properly detect the signals [12]. From a network point of view, minimizing the preamble and packet guard-time (packet overhead) would result in higher throughput and lower latency. This is especially important in an intra-DC scenario where many applications produce short sub-microseconds traffic flows. We have experimentally investigated the individual time contributions of the AGC and phase synchronization of a 10 Gb/s BM-RX for the OPS switching node.

The system utilized for investigating the performance of BM-RX in OPS system is shown in Figure 5.4. Input packets at different wavelength (λ_1-λ_N) are switched by the OPS node. Each packet contains a sequence of “1010…” preamble. A certain guard-time is placed in between consecutive packets. At the output of the OPS node, the switched packets are detected by the BM-RX. It consists of a BM trans-impedance amplifier (BM-TIA) featured with fast gain setting and a BM limiting-amplifier (BM-LA) recovering the amplitude [13]. A reset signal is applied externally in the experiments. Oscilloscope and BER tester are used for qualitative and quantitative evaluation of the detected switched packets. Three operational cases have been considered to experimentally evaluate the required preamble length including the effects caused by dynamic power, guard-time and SOA impairments. In the 4x4 switching condition, 10Gb/s NRZ OOK packets at λ_1 = 1552.56nm and λ_2 = 1555.74nm consisting of 1200ns (1500Bytes) payload are generated and sent to OPS node. The attached packet labels are set so that the packets are alternatively forwarded by the OPS to Port 1 and Port 2.

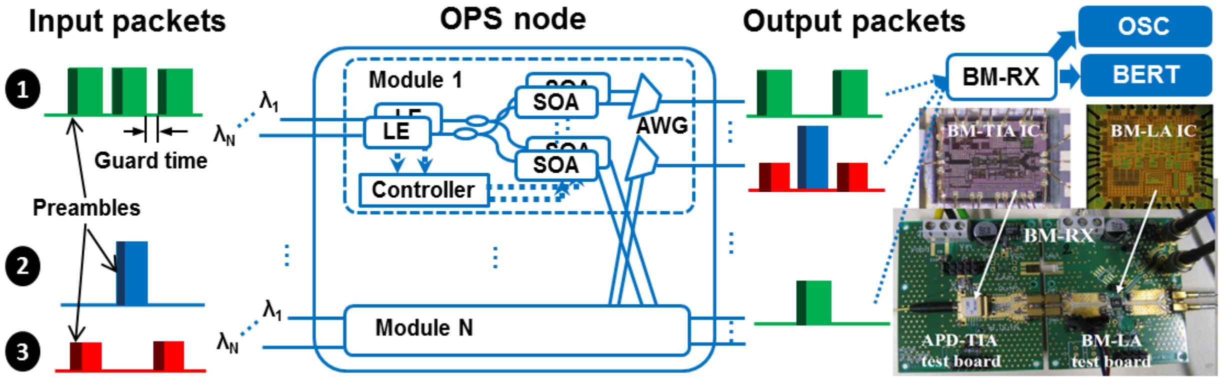


Figure 5.5: Experimental set-up to evaluate the OPS system with burst-mode receiver

Case I determines the preamble as a function of the dynamic power range and guard-time. In the experimental set-up shown in Figure 5.4, packet flow 1 may be switched to all the possible output ports at a certain power level. There should be enough preamble length and guard-time for the proper operation of BM-RX and at the same time, without compromising the throughput and latency performance. The preamble length is investigated as a function of received optical power and guard-time and it would also provide the information on the minimum guard-time that could be placed in between the packets.

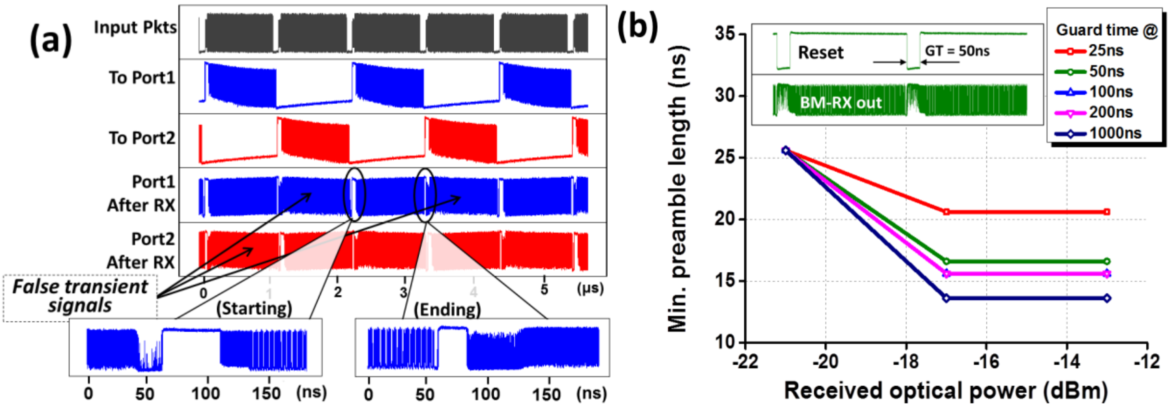


Figure 5.6: (a) waveforms detected by two separate BM-RXs; (b) minimum preamble length vs. input power at different guard-time

The waveforms of the input packets, the switched packets at port 1 and port2, and the detected packets by the BM-RXs are shown in Figure 5.5 (a). The zoom-in at the starting and ending of a packet provides a better vision of the preamble and the clear payload bits indicates correct amplitude recovery. Original empty switched time slot are then filled with false transient signals due to high gain of the BM-LA. These results have been obtained for 50ns guard-time and -22dBm input power. BER curves for back-to-back signal and switched packets after BM-RX are reported in Figure5.6. Error free operation has been obtained for both output ports with 1dB power penalty. In the next experiment, the guard-time and the optical power of the packets are varied to investigate the required preamble length that guarantees BER = 1E-9. The guard-times considered are 25ns, 50ns, 100ns, 200ns, and 1000ns. The optical power varies from -21dBm to -13dBm. The preamble length is optimized with 1ns step. Figure 5.5(b) indicates that input optical power ranging from -21dBm to -13dBm ensures a dynamic range larger than 8dB for the BM-RX with several nanoseconds decrease of the preamble. For a guard-time of 25ns, a preamble of 25.6 ns guarantees BER < 1E-9. Larger guard-time slightly reduces the required preamble because the charges at parasitic nodes in the circuits will be fully discharged. It also illustrates that the BM-RX would function properly after long empty packets sequence, as in the case of low traffic load.

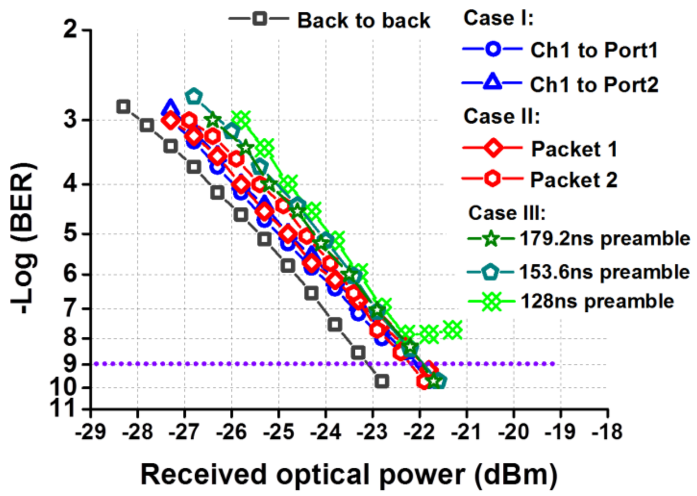


Figure 5.7: BER curves for B2b and 3 cases

Case II investigates the capability of the BM-RX to detect asynchronous switched packets. Packet flows 2 and 3 in the set-up shown in Figure 5.4 represent the situation that asynchronous packets with different wavelengths and power levels (representing packets that experience different link distances) are forwarded by the OPS to the same output port. As one of the key functions of the BM-RX, the gain and threshold should be fast settled to equalize the power fluctuation of incoming packets.

The waveforms of asynchronous packets at different wavelengths and optical power are detected by the BM-RX as shown in Figure 5.7(a). Fast AGC is of great significance to guarantee the BM-RX could handle the power fluctuation that may occur in the DCN. The BM-RX output trace shows that the power of two packet flows are equalized. The zoom-in also indicates the successful recovery of both asynchronous flows coming from different sources with different power level. The false transient signal which is caused by the reset settling of the BM-TIA has also been observed in the guard-time. BER curves for the two asynchronous packets are plotted in Figure5.6. Power penalty of 1dB due to switching operation has been measured.

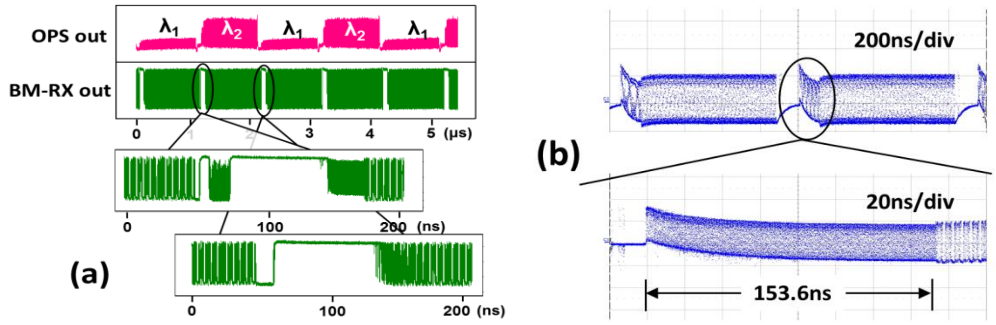


Figure 5.8: waveforms of asynchronous packets equalized by the BM-RX; (b) waveform of BM-CDR output

Case III investigates the preamble as a function of the clock data recovery (CDR) locking time. After the amplitude recovery conducted by the BM-RX, the clock phase alignment should also be realized by a BM-CDR. The employed BM-CDR is a fast-lock PLL-based CDR which is AC coupled to the BM-RX with a time constant of ~100ns. The preamble length is therefore increased to ~150ns, mainly in line with the time constant of the CDR settling time. The output waveform of BM-CDR is reported in Figure 5.7(b) which clearly shows the transient response after AC coupling. The BER curves as a function of different preamble lengths are also shown in Figure5.6. For preamble length > 153.6ns (including the 25.6ns due to the BM-RX) error free operation is achieved with 1dB penalty with respect to B2B signal. The Gated-VCO based CDR or Over-sampling based CDR would be a better solution to ultimately decrease the preamble contributed by BM-CDR.

5.2.2. Performance evaluation in a 1.3 µm system

Commercial deployment of optical communications in DCs is mainly focusing on two wavelength bands: 850 nm multimode and 1.3 µm single mode. Using the 1.3 µm band has two important advantages. Firstly at 1.3 µm the bandwidth distance product is superior compared to 850 nm communication. Secondly at 1.3 µm WDM can be exploited, since recently, low cost and low power consumption WDM transceivers have been demonstrated at this wavelength band [14]. This paves the way to the introduction of WDM in DCNs. Considering the advantages of deploying 1.3 µm band in the DCN, the performance of an OPS at 1.3 µm for intra-DCN is evaluated. The scheme used here is similar to the implementation of c-band operation and a 4x4 OPS is experimentally demonstrated.

The architecture of the investigated fast optical switch system is shown in Figure 5.8. WDM packets are de-multiplexed and processed in parallel. Payload is fed into a 1.3 µm SOA-based broadcast & select switch to allow non-blocking operation of the OPS while the label is processed by the switch controller. The nanoseconds switching time of the SOA in combination with the parallel processing of the label bits allows reconfiguration time of the switch in the order of few tens of nanoseconds. The switch controller checks the OPS LUT (updated by the SDN controller) and possible packet contentions, and then configures the switch to block packets with low priority. In case of contention, an optical flow control signal (ACK) is sent back to the cluster for packet retransmission. A low-speed DML at 1.3 µm driven by the switch controller is used to generate the ACK. This efficient optical flow control link ensures fast retransmission to minimize the packet loss and the latency.

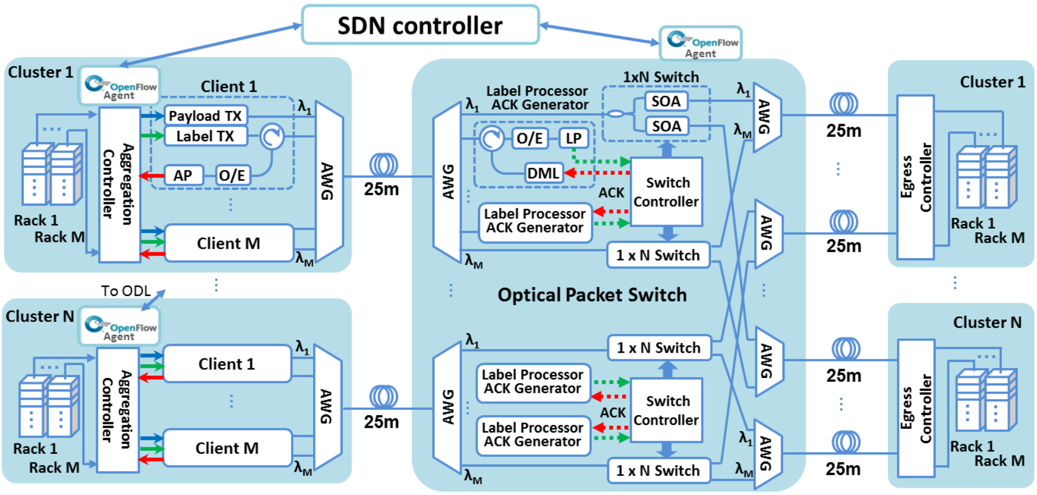


Figure 5.9: Architecture of the 1.3 µm optical packet switch

The assessments of 4x4 OPS at 1.3 µm are given in Figure 5.9. The BER curves and the eye diagrams for the 40Gb/s payload after the switches are presented in Figure 5.9(a). Error free operation with only 1dB power penalty after the switch is measured. As shown in Figure 5.9(b), a packet loss lower than1E-5 and an average end-to-end latency less than 430ns (including 250ns offset for 25m transmission link) could be achieved under relatively high traffic load of 0.5 and buffer capacities of 16 packets. These results are in agreement with the numerical studies conducted to investigate the scalability to 100's port-count and the latency performance. Capability of scaling to large port-count of the 1×N switch is also experimentally investigated by using an attenuator to emulate the splitting loss of the broadcast stage. The penalty (measured at 1E-9) and OSNR degradation at the switch output as a function of N and for different SOA bias currents are reported in Figure 5.9(c). It clearly shows that 128x128 ports at 40 Gb/s could be implemented with less than 2 dB penalty.

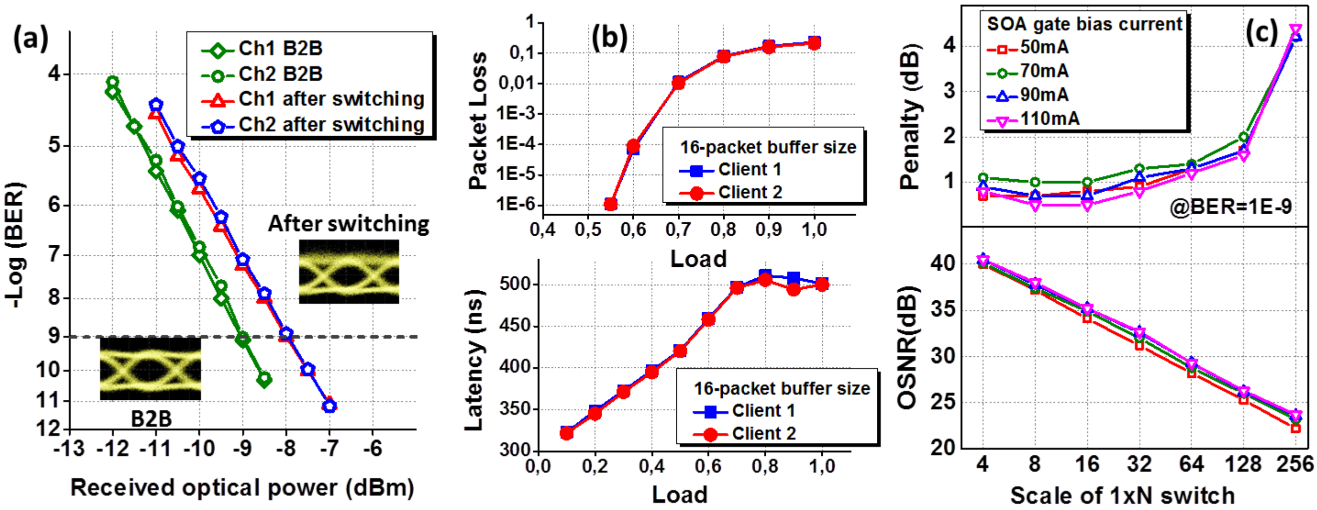


Figure 5.10: (a) BER curves and eye diagrams; (b) packet loss and latency; (c) penalty and OSNR for scaling

5.2.3. Evaluation of OPS-based virtual network reconfiguration

The next-generation DCs are required to provide more powerful IT capabilities, i.e. more bandwidth, storage, shorter time to market for new services to be deployed [15]. One of the key requirements of the future DC is the multi-tenancy. DCN virtualization is the key enabler for supporting multi-tenancy [16]. By taking advantage of virtualization, multiple coexisting DC virtual networks (VNs) will be created allowing for the efficient sharing of the heterogeneous DCN resources to support diverse services and applications running on top of VNs. As the demand of users or applications has been changing, the established VNs need to be reconfigurable and adaptive to the dynamic applications requirements. As one of the key benefits enabled by SDN framework, the OPS-based dynamic network virtualization is evaluated. In the scenario under study, we define a virtual network as a set of OPS flows that are associated in some way (e.g. they belong to the same user/tenant). From the OPS node perspective, the data flows associated to the VNs are stored in the LUT of the FPGA controller that configures the actual switching of the optical packets accordingly. Hence, by modifying the content of the LUT we can dynamically reconfigure the virtual networks of the scenario.

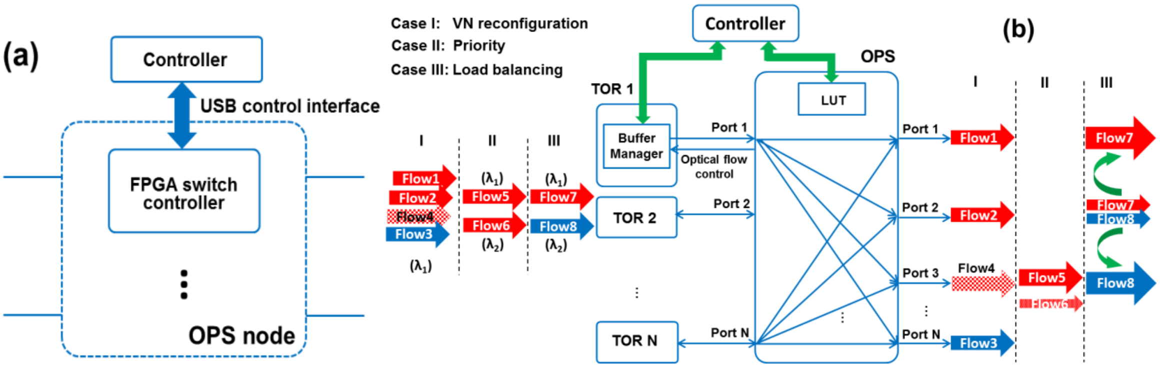


Figure 5.11: (a) Schematic of the control interface; (b) experimental set-up for the reconfigurable VN

Figure 5.10 (b) illustrates the experimental setup with three selected cases to validate the reconfiguration, statistical multiplexing with quality-of-service (QoS) guarantee, and load balancing operation of the virtual network, respectively. The data flows, which aggregate sequences of packets from a source to certain destinations, are statistically multiplexed at the ToR and then transmitted to the OPS node. The buffer manager inside the ToR handles the flow destination and generates the labels that will be associated to the flows and include the forwarding information and the class of priority of the flow. The buffer manager stores the label information and implements the (re-)transmission according to the ACK sent from the OPS node. The gate used for controlling the transmission of packetized 40Gb/s NRZ-OOK payloads (460ns duration and 40ns guard time) is triggered by the buffer manager to emulate the (re-)transmission.

In Case I we have demonstrated the VN reconfiguration and resource sharing exploiting statistical multiplexing. Centralized controller has provisioned the VN1 comprising ToR1 and ToR2, and the VN2 comprising ToR2, ToR3, and ToRN. Flow1 and Flow2, belonging to VN1, and Flow3 belonging to VN2 are statistically multiplexed on the same wavelength to different destinations. As resource competition may exist between flows from same ToR, the controller has been given the authority to assign different priority levels for each flow. A reconfiguration of VN1 is now required to include in VN1 also connectivity with ToR3 (to support Flow4 forwarding). To do this, the controller updates the LUTs of the ToRs and the OPS, accordingly. Figure 5.11(a) shows the LUT update for the original VN1 and the reconfigured VN1' (LUT'). The traces in Figure5.11(a) show that, before the reconfiguration, flows having ToR3 as the destination are dropped since no matched label will be found at the OPS. On the contrary, after VN1 reconfiguration and the LUT update, the flows towards ToR3 can be properly delivered.

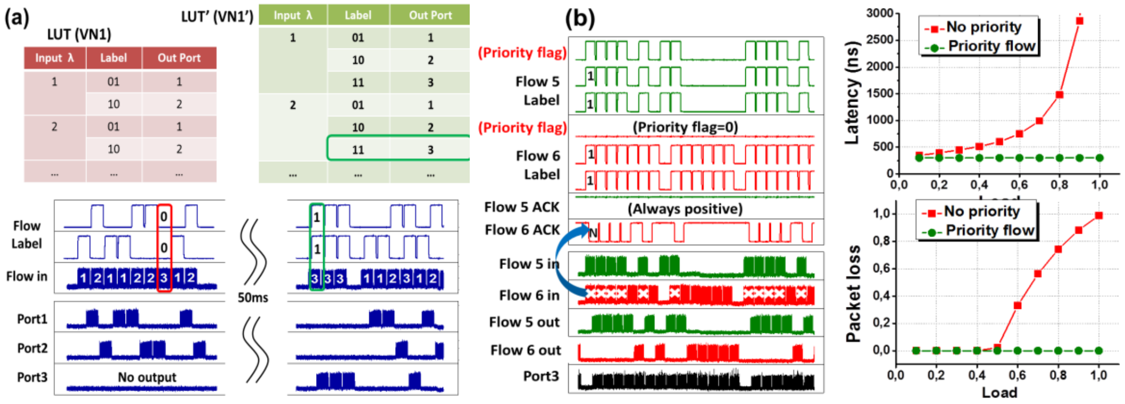


Figure 5.12: (a) VN1 reconfiguration; (b) Time traces of Flow5 & Flow6 and packet loss & latency

It has been measured that it takes about 50ms for the update procedure (control communications between the controller with the ToR and the OPS); after that, flows are statistically multiplexed and switched at sub-microseconds time scale. It is worth to note that the other flows destined to ToR1 and ToR2 perform hitless switching during the VN reconfiguration time.

In Case II we have demonstrated statistical multiplexing flows operation under different classes of QoS. Flow5 and Flow6 are heading to the same output port (Port3). Flow5 has been assigned higher priority. One label bit has been used as priority flag. In case of contention at OPS, Flow5 will be forwarded to the output Port3 to avoid packet loss and large latency caused by retransmission. Figure5.11(b) shows the label and flow traces, the packet loss, and the latency performance for the two contented flows. Note that the ACK signals for Flow5 are always positive (always transmitted), while the negative ACK signals for Flow 6 indicate retransmission. The packet loss curves confirm no packet loss for Flow5, while the buffer employed at the ToR prevents packet loss up to load < 0.4 for Flow6 and then it increases linearly with the load. Similar behavior is observed for the latency.

In Case III we have demonstrated load balancing operation for the flows belonging to different VNs. Let us assume that Flow7 in VN1 and Flow8 in VN2 have the common output Port2 among two potential destinations. The contention at Port2 would cause high packet loss for both flows. The contention probability for the output ports indicating the destination usage could be collected from optical flow control signal. Upon reception of the real-time status of the packet loss per flow and the occupancy of each alternative port from the ToR, the controller can enable the balance of the load to output port with less usage. Performance improvement by such load balancing strategy between two flows from different VNs has reported in Figure5.12. In the experiment, the load of Flow7 and Flow8 has been increased from 0 to 0.8, with 50% probability that a contention occurs at Port 2 at the beginning. Firstly, Figure5.12(a) shows that a packet loss larger than 1E-3 for both flows is measured, in case the controller does not take any action. Once the load balancing is triggered by the controller, a target packet loss threshold of 5E-5 has been set. Above this threshold, the dynamic adjustment of controller would balance the load at Port2 to Port1 (for Flow7) and Port3 (for Flow8) with a step of 0.1. It can be observed in Figure 5.12 (b) that the packet loss is kept less than the threshold and the latency goes down which guarantee the QoS meeting the requirement.

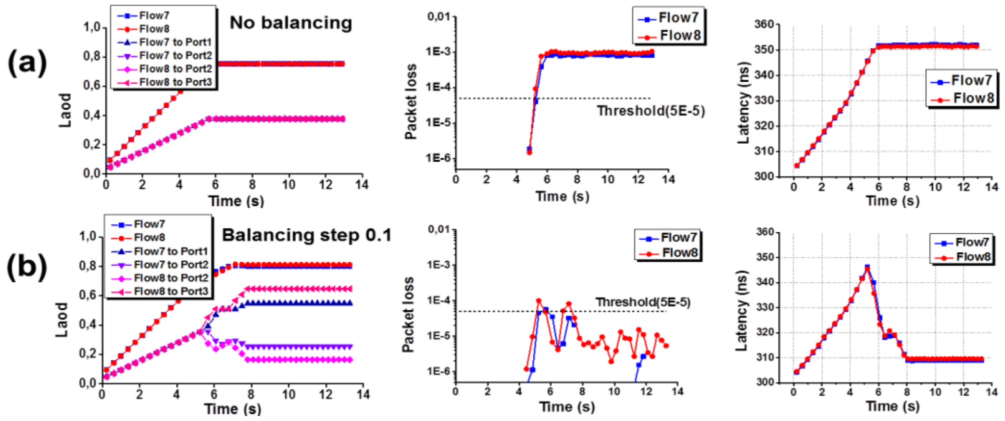


Figure 5.13: Load, packet loss and latency changes without adjusting (a) and with load balancing with step of 0.1 (b)

5.3. Interface with control plane

The implementation of the southbound interface to enable the SDN-based control of the OPS is described in D3.1 [2]. As highlighted in section 2.2, the key element in such implementation is the OF agent. Residing on top of the OPS hardware, the agent bridges the control plane and the FPGA-based switch controller. To do this, the agent receives the extended OF protocol messages from the SDN controller and transforms them into a set of actions conducted over the FPGA-based OPS switch controller. More specifically, the agent configures the LUT implemented in the FPGA, which is the responsible to control the switching capabilities of the OPS node as well as to keep traffic statistics. Hence, the OF agent writes/reads the LUT of the FPGA upon request of the SDN controller. As illustrated in Figure 5.13(a), in the OPS case the OF agent is connected to the FPGA through a USB link. Figure 5.13(b) provides the implementation perspective of the southbound interface and the OF agent. As shown in the figure, the OF agent is actually split into two software modules, namely the Java Agent and the C Agent, which communicate with each other by means of a UDP socket interface. Hence, the

Java Agent implements the extended OF API that enables the communication with the SDN controller, and the C Agent is the responsible to configure the FPGA-based OPS switch controller.

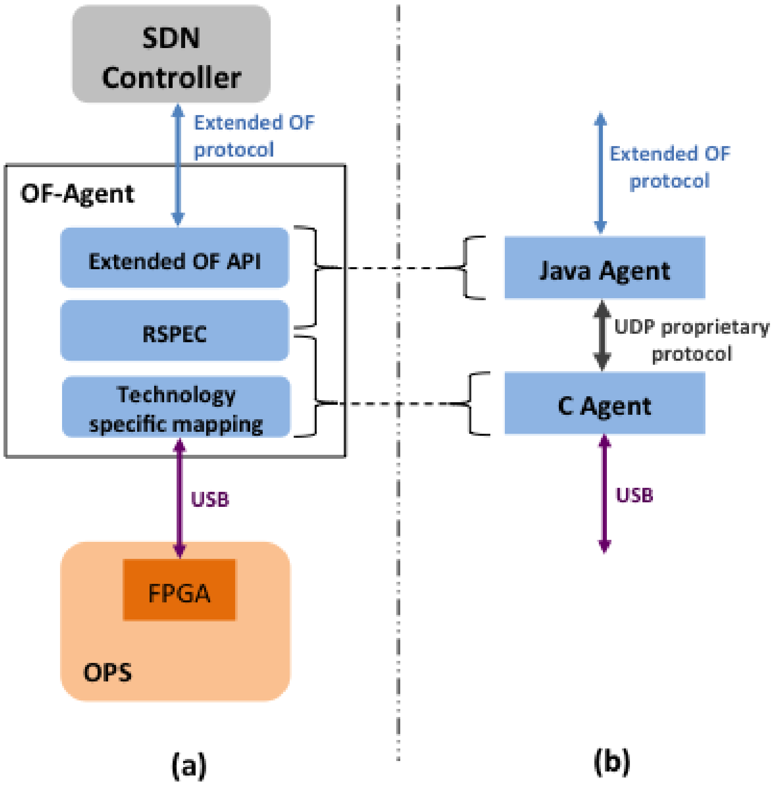


Figure 5.14: (a) OF-Agent architecture (b) OF-Agent implementation scheme [17]

During operation, the Java Agent receives the OF messages coming from the SDN controller and transforms them into a set of UDP messages that are sent to the C Agent. Upon the reception of such messages, the C Agent acts over the FPGA to configure the LUT or request for monitoring parameters. Further details on the OF agent and southbound interface implementation can be found in D4.4 [17].

6. Synthetic hybrid DCN architecture

In this section, we present the concept and performance of the synthetic hybrid DCN architectures. Exploiting the dynamic composition and reconfiguration of the hybrid OCS/OPS data plane, it is possible to reconfigure the DCN in order to cope with the heterogeneity of the workloads of the applications running at the DC, avoiding potential congestions. Moreover, it allows for the creation and composition of mission-specific virtual slices for the rental of external entities in order to develop their own business models. To this end, the Section 6.1 elaborates on the benefits of dynamic on-demand DCN configuration. Additionally, it presents an experimental implementation of hybrid DCN. The Section 6.2 focuses on the composition of hybrid virtual slices. To highlight the benefits of a hybrid DCN in this regard, it focuses on a multi-tenant scenario, where several tenants request for a set of virtual slices. Finally, the Section 6.3 evaluates the performance of real world scientific applications when executed in a DCN based on the LIGHTNESS solution.

6.1. Synthetic hybrid DCN configuration

The advantage with the LIGHTNESS DCN design is that interconnections between ToRs in the same cluster can be reconfigured in an arbitrary manner. The capability of traffic between each ToR pair could be dynamically adapted and expanded to the maximum of 100% utilization of connections provided by each ToR. Furthermore, by setting up appropriate links in the NFP node, the architecture is not only able to plug in OCS connectivity between any ToR pair, but also able to compose any topology with associated OPS/OCS functions. Any link from any ToR can be assigned to any switching module on demand, so that programmable intra-cluster DCN can be constructed dynamically in terms of function as well as topology. By doing this, collocated workloads or heavy traffic loads from the same ToR can be split and loaded on different paths in the network. Meanwhile, traffic loads from different hot ToRs can be isolated by rearranging topological locations of hot ToRs in the DCN. Therefore hot-spots in the DCN can be “cooled down” as skewed traffic pattern in the DC can be accommodated in a more balanced way. Likewise, such programmability works also for the inter-cluster DCN, thus the overall network utilization and network performance are improved. Last but not least, the plug-in function pool (OPS, SSS, etc.) in each NFP node can be shared by the whole DCN through the inter-cluster node. Such sharing scheme brings tremendous agility to the DCN in a cost-efficient way: faulty modules in one cluster can be replaced by modules among the whole DC range without sending human hands into it; network topology and operation can be decided according to the QoS required by each task, and also the network resources available or assigned to each cluster in the DC.

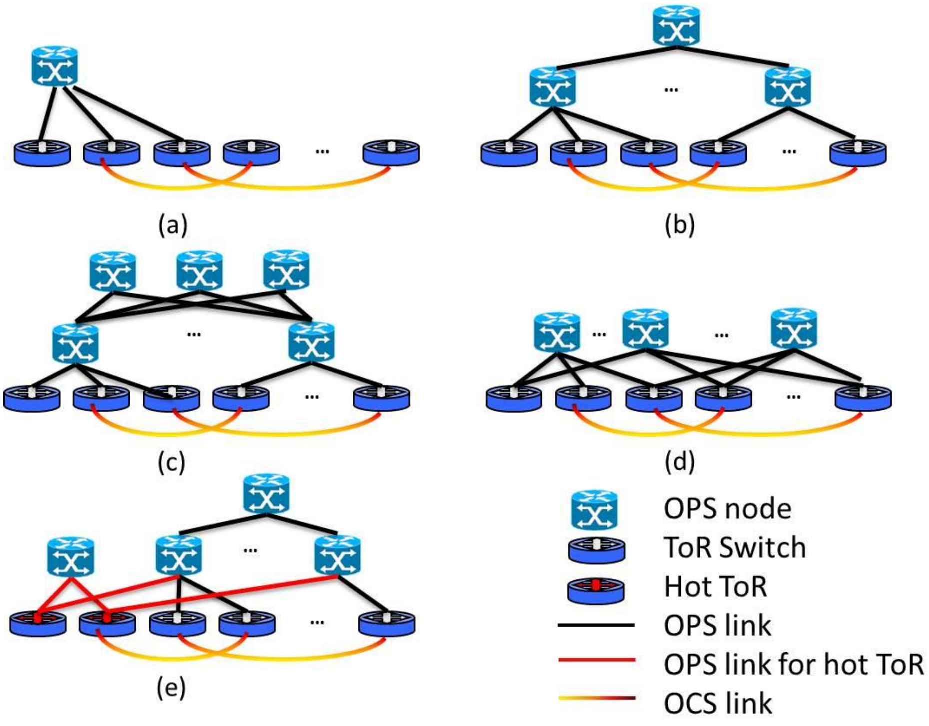


Figure 6.1: Different network topologies supported by LIGHTNESS DCN approach

Figure 6.1 illustrates several network topologies that can be generated by the DCN architecture presented in Section 2.1. Here we focus on the intra-cluster DCN architecture. However the inter-cluster DCN architecture can be analyzed in the same way. If only a few OPS modules are accessible in a cluster, given the limited port line rate of OPS module, only part of ToRs can be connected to OPS modules at the same time, as shown in Figure 6.1(a). The connections between OPS modules and ToRs can be reconfigured frequently so that communications between different ToRs are enabled through OPS in different time slots. And OCS can provide part of connectivity between ToRs as well. However, this architecture cannot provide full connectivity and the traffic between ToRs without connection has to be hold or blocked. Figure 6.1(b) shows a non-blocking architecture by cascading OPS nodes following single-rooted tree topology. Full bisection bandwidth is given between ToRs in the same OPS branch node but not for those from different branches due to hierarchical oversubscription. Apart from the OPS system, OCS system provides supplementary connectivity dynamically between ToRs where augmented capacity is required in this architecture (e.g. long-lived bulk data transfer or inter-branch communication). This use case fits to tasks using intelligent workload placement algorithms which intend to allocate network-bound service components to servers/racks in the same OPS branch [18]. However, the performance of network starts to degrade when more traffic is required to travel form one branch to other branches. If more OPS modules are available, architectures enabling full-bisection bandwidth can be constructed. Figure 6.1(c) gives an example leveraging multi-rooted tree topology. Other topologies such as Fat-tree, D-cell or B-cube can be realized as well, depending on the amount of OPS modules available. Moreover, given a tight requirement of latency for latency-sensitive tasks, Figure6.1(d) illustrates a spin-leaves topology with strict “one hop” OPS connections between any ToR pair by connecting them with sufficient individual OPS nodes.

Besides, our design also provides a traffic load management strategy to improve the total throughput by reconfiguring the connections between ToRs and OPS nodes, and dispersing hot ToRs in different branches depending on traffic loads they are carrying, as illustrated in Figure 6.2. By doing this, heavy traffic loads from hot ToRs can be placed to the branches with fewer loads thus the traffic in each branch is balanced and the whole network performance is thereby improved. In addition, providing appropriate OCS links can further improve whole network performance. However, a trade-off of channel assignment has to be made between OCS and OPS system as they share the capacity provided by the same ToR. Only when the increment of efficient bandwidth brought by OCS links exceeds the loss of it due to OPS capacity reduction (e.g. provision of OCS links between hot ToRs in different branches where efficient bandwidth is bottlenecked in OPS system due to oversubscription), the whole network will benefit.

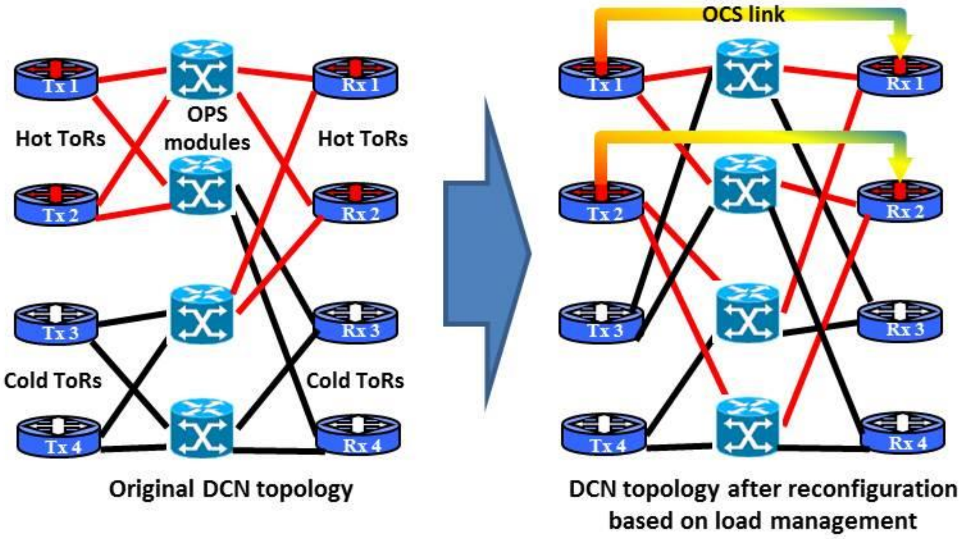


Figure 6.2: DCN reconfiguration with traffic load management strategy

Nevertheless, all the observation about highly skewed traffic pattern in DCs stands for the viewpoint that it is unnecessary to provide static full-bisection bandwidth between all ToRs, which leads to a waste of network resource at most of time. A more efficient way of network utilization is allocating network resources targeting at traffic types on demand. Given the fact that the traffic in DC is dynamic and the locations of hot-spots are usually unpredictable, higher capacity needs to be dynamically provided between ToR pairs that can benefit from it. In our design, network topology and functions can be decided and constructed dynamically on the basis of a global view of network traffic loads. Thus traffic from hot ToRs can be picked out and given more resources with higher capacity, whereas the rest traffic can be accommodated with minimum resource occupation, as shown as the star-tree hybrid architecture in Figure 6.1(e). In this architecture, hot ToRs, with heavy loads and fan-in/fan-out traffic pattern, are identified in real time. Traffic from/to hot ToRs is given the maximum capacity and loaded on several OPS links in parallel to reduce congestion. On the contrary, traffic among cold ToRs is accommodated by hierarchical oversubscribed network. Bounded-degree bulk data transmission, if there is any, are handled by OCS links.

6.1.1. Implementation of hybrid DCN communication

In this section, we present a practical implementation of the hybrid DCN solution. Figure 6.3 shows the overall DCN architecture, including the optical data plane and the SDN-enabled control plane. In the data plane, instead of a hardwired interconnection of different network elements, a more flexible DCN architecture equipped with an AoD node is adopted, which consists of an optical backplane (i.e. a Polatis fibre switch [6] with a large port number) with switching modules (i.e. OPS, Wavelength Selective Switch) and passive devices (e.g., Mux/DeMux, splitter) attached. With this AoD architecture, different arrangements of input/output and modules can be constructed by dynamically setting up appropriate cross-connections on demand in the optical backplane according to different applications' requirements. OCS connections can be established through configuration of the backplane itself. The OPS consists of a modular SOA-based architecture with highly distributed control for port-count independent reconfiguration time, which has been designed and prototyped as shown in Chapter 5. A hybrid ToR is able to parse the input traffic from servers and send it out in different modes/connections (OPS/OCS). In this way, the FPGA-based ToR performs traffic aggregation (e.g., traffic destined to the same server/Rack) and application-aware traffic classification to initiate either OPS or OCS connection.

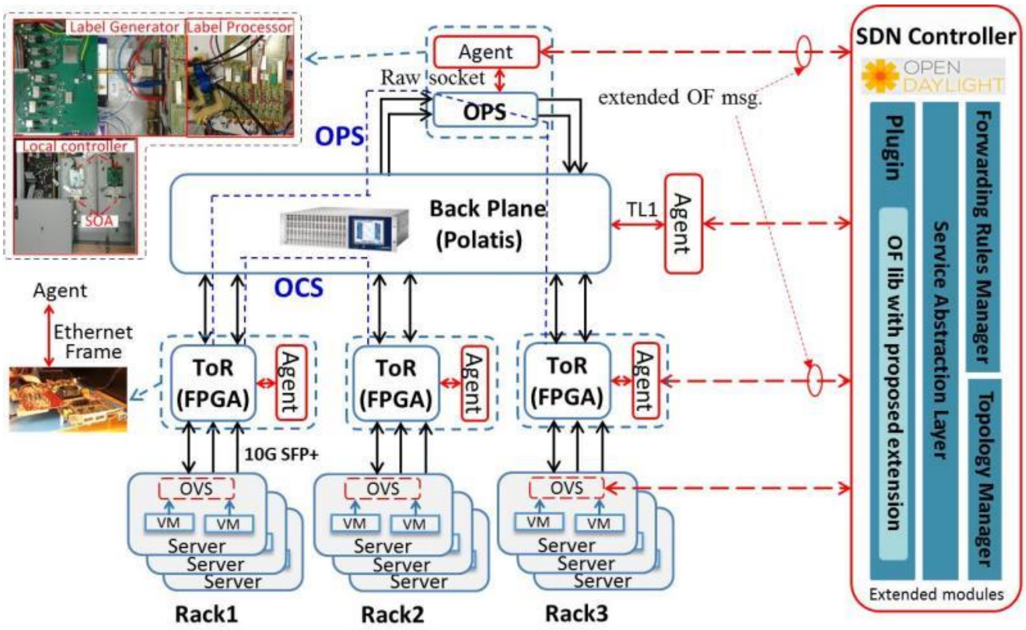


Figure 6.3: SDN enabled Hybrid OPS/OCS DCN Architecture

In the experimental setup, a 192×192 Polatis fibre switch is used as backplane with a 2×2 OPS switch and other elements such as (De)Mux plugged in to compose an AoD node. Hybrid ToRs are implemented with FPGA opto-electronics (HTG Xilinx V6 board) with 12 10GE ports. All the ToRs are connected to the backplane via 10GE port. Servers are connected to ToRs via 10GE optical links. OPS could forward the traffic to the possible destination with ~10ns switching time, while OCS ensures high-throughput data transmission. Here, we demonstrated the SDN enabled dynamic OPS/OCS connectivity provisioning and application triggered OPS/OCS switchover.

Figure 6.4 shows the control messages for the OPS and OCS connection provisioning. Generally, when a new traffic flow arrives and there is no matching rule in Open vSwitch (OVS), OVS will send a packet_in message to the OpenDaylight (ODL) SDN controller through to trigger a new connection establishment. An application has been implemented to calculate the configurations of switches i.e. AoD, OPS and ToR. Flows are pushed to different devices via the REST API of ODL, which has been also extended to enable optical flow provision. In this experiment, two services (i.e., short-lived http request and long-lived Rsync service that consumes more bandwidth) are running in two virtual machines (VMs) within the same rack. Their packets are tagged with different VLAN IDs according to the flow entries in OVS, which is updated by ODL. To establish an OCS connection, the backplane cross-connection (cflow_mod, OF message type 22) and flow entries (flow_mod) for in ToR1 (source) and ToR2 (destination) need to be configured. The measured ToR-to-ToR OCS connections provisioning time are 753ms including OF backplane configuration message round trip time (251ms×2), backplane hardware configuration time (16ms) and source and destination ToR LUT update and configuration time (235ms). For the OPS connection, besides a flow entry with label configuration information sent to the source ToR, another flow entry for the OPS node need to be installed to update the OPS LUT. The measured time required to update the LUTs is 214 ms. Please note that once the LUTs are setup, the underlying OPS connections will operate at its own speed (sub-microseconds) decoupled from the control plane.

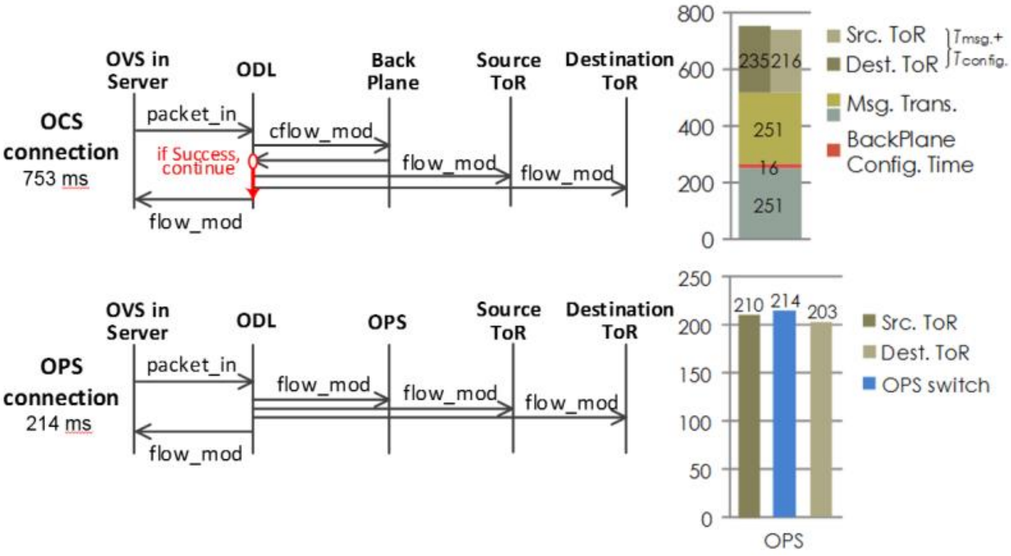


Figure 6.4: Port switch based OCS switch's interface with control plane

6.2. Multi-tenant synthetic hybrid DCN composition

In the following section, the performance evaluation of programmable synthetic hybrid OCS/OPS DCN is presented. For this, as a case study, the multi-tenant scenario where each tenant requests several virtual slices according to their needs is being considered. To this end, algorithms that compose the synthetic hybrid OCS/OPS DCN over which the multiple virtual slices will be supported are presented. To evaluate the benefits of a hybrid synthetic DCN, a pure OCS solution is used as a benchmark.

6.2.1. Scenario description

Modern DCs allow for the existence of multiple sets of applications managed by different tenants to be run in parallel, each one with its own allocated resources and associated Service Level Agreements (SLAs). Thus, multi-tenancy becomes a very important feature for nowadays DCs, allowing offering a high customizable architecture for exploitation by external entities that will utilize the leased architecture to develop their business models. In such scenario, the DC owner must compose the synthetic infrastructures over which the requested resources by the different tenants will be mapped. Such synthetic architecture will be then exposed to the tenant so he can control and manage it according to the applications that will run on top of it. Figure 6.5 illustrates this scenario.

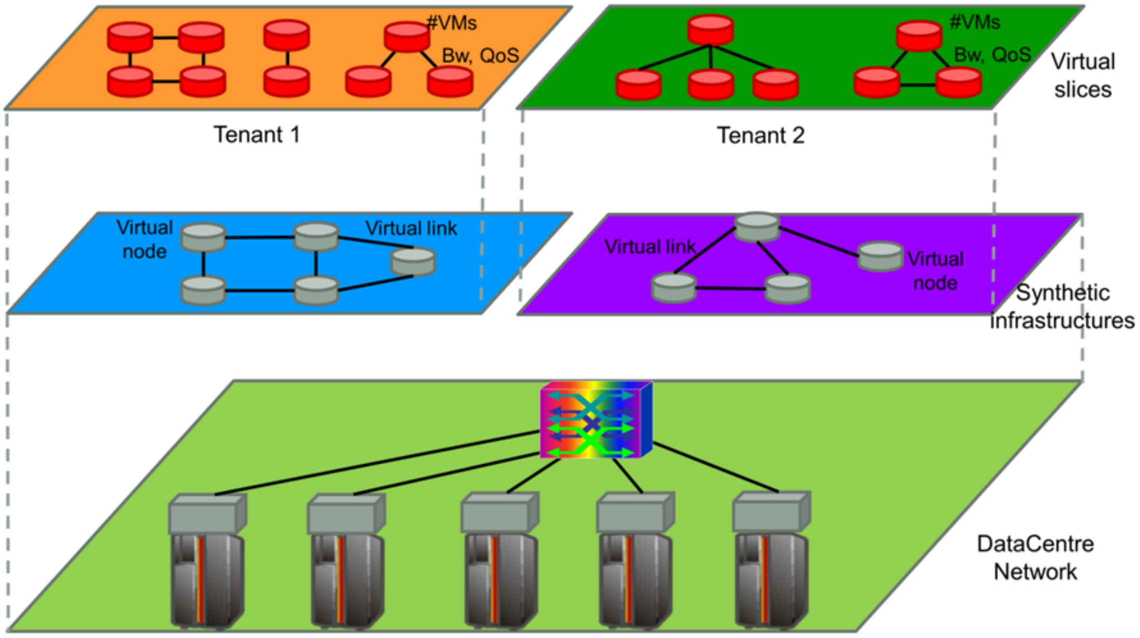


Figure 6.5: Multi-tenant synthetic DCN scenario.

An optimal composition of the synthetic architectures plays a capital role on the overall performance of the DC. Since optical resources are expensive, in order to minimize the capital expenditures (CAPEX) of the DC owner, an algorithm with the goal of minimizing the needed optical resources, in particular transponders (TSPs) at the DCN, is presented. The algorithm takes as an input the set of tenants and their requests and provides the minimal cost (in terms of TSPs) necessary synthetic architectures to successfully allocate all the requests. Additionally, such approach provides better physical resource utilization, thus allowing a larger number of virtual slices and tenants to be successfully allocated on top of the DC infrastructure, potentially increasing the revenues of the DC owner.

The presented algorithm and the scenario under evaluation follow up the studies presented in [19], where the composition of virtual data centres (VDC) was targeted. Given a DC architecture, several tenants request each one of them a VDC composed of pools of VMs interconnected through virtual links that are characterized by a requested bandwidth and a required QoS. A key point on the overall VDC embedding process relates to the mapping of the virtual nodes in actual physical servers and the mapping of virtual links in actual physical network resources. Additionally, in a hybrid OCS/OPS data plane architecture, the most suitable switching technology has to be chosen to map the virtual links of the VDC according to their characteristics, namely bandwidth and QoS.

Depending on the chosen technology, additional considerations have to be taken into account. If a VDC is served employing OCS, its virtual links will be mapped onto optical circuits. Said optical circuits will last for the whole duration of the VDC instance and can only be utilized by the particular virtual link mapped on it. In this way, the isolation between VDCs employing OCS is guaranteed by means of physical isolation. On the other hand, if a VDC is served employing OPS, its virtual links will be mapped onto optical packet flows. Under such circumstances, different packet flows may share some physical resources (wavelength and link) thanks to the statistical multiplexing property of OPS. However, the QoS in terms of tolerated packet loss ratio of the virtual links must be guaranteed in order to ensure a proper logical isolation. In OPS, the average packet loss ratio is related to the average load of a wavelength [2], so more load implies more packet loss due to contention at the output ports of the OPS switch. In order to guarantee the QoS of the virtual links, the total packet flow circulating through a particular wavelength and port has to be limited so as to not surpass the load limit associated to the packet loss ratio of the flows. Note however, that such restriction only applies when several different optical packet flows share physical resources; this limitation would not take place in the presence of a single packet flow, since there is no possibility of packet contention in such situation.

The composition of synthetic architectures shares some similarities with the concept of VDC embedding but with some differences: although the physical resources that are allocated to a particular tenant cannot be shared with other tenants so as to guarantee the physical isolation between them, the physical resources that are allocated to a virtual slice (i.e. VDC) of a tenant may be shared with other slices of the same tenant. In particular, if OCS is employed to map the virtual resources, an optical circuit may be shared by multiple virtual links as long as they share both endpoints, so end-to-end optical grooming can be applied. If OPS is utilized, virtual links may share the same wavelength in a physical link as long as the QoS restrictions are respected as explained previously. In this way, the resulting synthetic DCN, rather than being the union of all the requested slices of a tenant, is the composition of their nodes and virtual links, resulting in an aggregated infrastructure that ensures both minimal resource utilization and guarantees the exact requirements of all virtual slices. Figure 6.6 provides a schematic representation of such process.

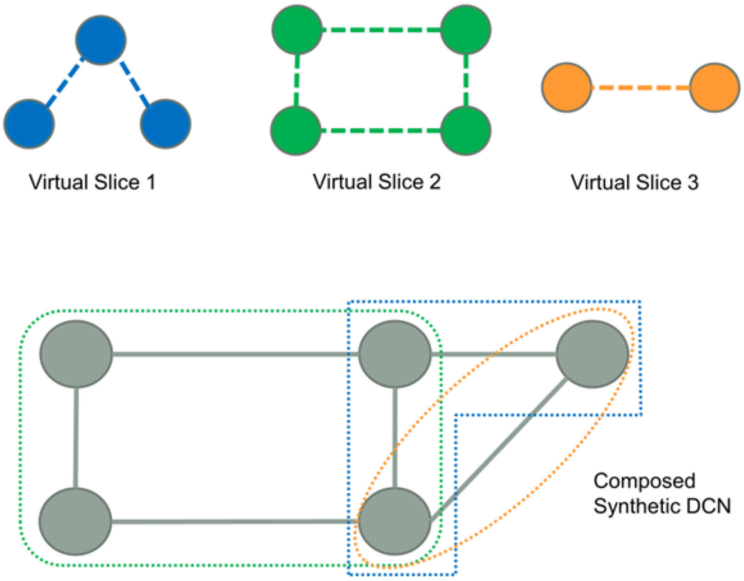


Figure 6.6: Example of synthetic DCN composition.

After this discussion, the following section describes the proposed algorithm for synthetic hybrid OCS/OPS DCN composition in a multi-tenant scenario.

6.2.2. Synthetic DCN composition algorithm

The proposed algorithm is utilized for obtaining the synthetic DCN architectures that will support the multiple virtual slices requested by a set of tenants. In this regard, the algorithm is utilized in a static scenario, where the goal is to obtain the minimum number of TSPs needed to correctly allocate all the virtual slice requests, searching for both the node and link mapping of the virtual slices as well as the technology to be employed by the virtual links. Note that, since hybrid OCS/OPS synthetic DCNs are considered, links may be mapped individually over OCS or OPS depending on their characteristics, but not both at the same time.

The proposed algorithm is executed sequentially for all the tenants, one at each time. Since tenants must be physically isolated from the others, the resources allocated to one tenant cannot be shared with other tenants. This makes the iterative approach completely valid as the resulting number of needed resources will be the summation of the needed resources per tenant, which have been minimized during the execution of the algorithm. Moreover, the iterative approach allows for a better scalability of the composition process when compared to a joint approach, where all involved tenants are considered at once.

After this discussion, the presented algorithm can be structured in the following steps:

Step 1: Identify the chances of statistical multiplexing in the virtual slices. This involves identifying the virtual nodes that would allow for more virtual links sharing the same wavelength at source/destination. This can be obtained by adding the total flow outgoing/incoming from the same virtual node and see how many TSPs would be needed to accommodate all of them. Then, according to this, virtual slices are ordered according to their chances to utilize statistical multiplexing in OPS, from more to less.

Step 2: Node mapping; order the virtual nodes of the virtual slice in descending order according to the number of requested VMs. Then, map iteratively the virtual nodes in the racks in a first fit fashion. One important restriction is that virtual nodes belonging to the same virtual slice cannot be mapped in the same rack. This is done in order to guarantee some degree of fault tolerance in the case that a failure happens in one of the racks.

Step 3: Link mapping; after the node mapping, map virtual links to OPS if they allow for sharing of virtual resources thanks to the statistical multiplexing. If not, they are mapped to OCS. In both cases, the mapping is performed employing a shortest-path for the routing and a first-fit selection for the wavelength.

Step 4: Flow aggregation; determine if the following virtual slice can be aggregated with the current utilized resources. First, try if it can be mapped to currently established virtual links employing OPS, respecting both its own QoS limit and the one of the already established links. If not, try if the virtual slice can be aggregated utilizing OCS with already established OCS flows. If not, go to step 2 for the current virtual slice.

After all the virtual slices of a tenant have been mapped, the algorithm is executed for the next tenant, starting at step 1. Note that all the optical resources employed by previous tenants will not be available for the next tenant in the set.

6.2.3. Algorithm evaluation

To evaluate the performance of the proposed algorithm and to assess the benefits of the LIGHTNESS hybrid data plane solution, a series of simulations have been executed. The scenario under consideration consists in two clusters of 8 racks per cluster. Each rack is connected to the DCN thanks to a ToR switch, which are connected in a tree fashion to an intra-cluster Architecture on Demand AoD OCS switch and an OPS switch, which at the same time are connected to an inter-cluster AoD OCS and OPS switches. Each rack contains enough servers to allocate all the requests. Moreover, since the algorithm is targeting on the minimum number of necessary TSPs, it is considered that each ToR has enough TSPs and optical channels to support all the virtual links. Moreover, each optical channel has the capability to either transmit OCS or OPS optical flows, but not both at the same time. Two scenarios have been considered: 1) a pure OCS case, where the data plane consists only of OCS switches; and 2) a hybrid OCS/OPS case, reflecting the LIGHTNESS solution.

As for the request set, 40 tenants have been considered. Each tenant may request between 1 and 5 virtual slices. At its turn, each virtual slice can be composed of 2-6 virtual nodes, which may request between 1-10 VMs. The virtual nodes are connected randomly (with the same probability) with virtual links requesting between 10-100% the capacity of an entire wavelength, in steps of 10%. As for the QoS restrictions, the QoS of the virtual links is chosen uniformly between 10-6, 10-4 and 10-3 which corresponds to 60%, 64% and 70% average maximum load in OPS [20].

Figure 6.7 shows the necessary number of TSPs to be equipped in the DCN as a function of the number of virtual slices requested per tenant. It can be appreciated that the hybrid architecture yields to savings on the number of the necessary TSPs, ranging from about 10 to 15% savings. This is due to the statistical multiplexing capacity of OPS, which may be exploited for virtual links sharing the same source but with different destination.

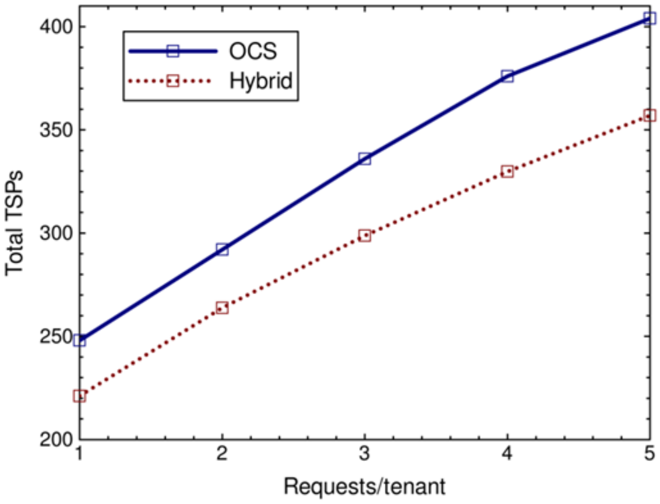


Figure 6.7: TSPs as a function of the number of requests per tenant

In light of these results, additional tests have been done. For the following set of results, the number of requests per tenant has been fixed to 3, with the rest of the parameters remaining untouched. The potential savings of the hybrid architecture are tightly related to the statistical multiplexing capabilities of OPS, which at its turn are tightly related to the QoS restrictions of the packet flows. To investigate the impact of this parameter, Figure 6.8 shows the evolution of the number of necessary TSPs as a function of the tolerated QoS by the virtual links, which is fixed the same for all of them. The depicted QoS values refer to the average normalized maximum load allowed to guarantee a certain packet loss ratio, meaning that higher QoS values reflect the case where a higher packet loss is allowed. It also can be understood as the OPS switch tolerating more load per wavelength for the same packet loss ratio, meaning that a higher performance OPS switch is utilized.

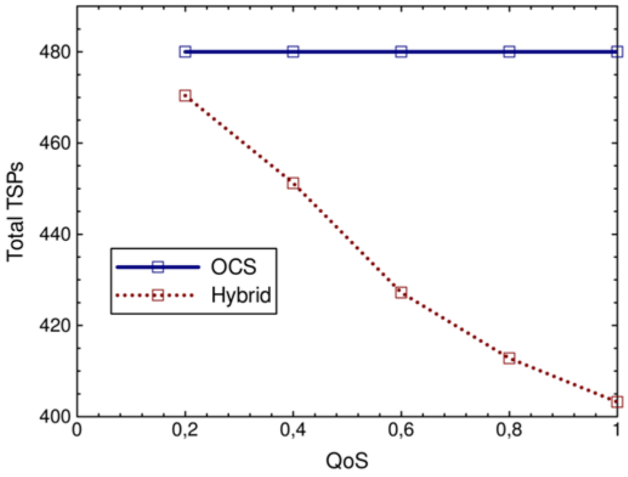


Figure 6.8: TSPs as a function of the QoS per virtual link

As expected, higher QoS limits allow for further reductions in TSP utilization, since more virtual links benefit from the statistical multiplexing properties of OPS as more load can be packed in a wavelength without surpassing the QoS limits. An additional 8% reduction is appreciated when going from a QoS limit equal to 0.6 to 0.8.

Besides the QoS restrictions, another important parameter relates to the virtual nodal degree of the virtual slices, since the statistical multiplexing capabilities of OPS are potentially more utilized in more meshed virtual scenarios. This is because there are more chances to share the same wavelength in a particular physical link as the several virtual links may share the same source and/or destination, thus allowing the share of part of the whole end-to-end path. In contrasts, in a pure OCS architecture, although more meshed virtual networks may allow for more aggregation at the optical circuits, such aggregation is only possible if virtual links share the whole end-to-end path and wavelength. For this reason, a hybrid OCS/OPS architecture is potentially more beneficial when there is a high number of point to multi-point communications. To investigate this, Figure 6.9 illustrates the evolution of the number of TSPs as a function of the number of nodes in a point to multi-point communication. Particularly, a value of 2 in the X axis means that only two nodes are involved, hence, a one to one communication is taking place. As for the rest of values, 3 would mean that 3 nodes are involved, with 1 node communicating simultaneously to the other 2, 4 would mean 1 node communicating simultaneously to 3 nodes and so on.

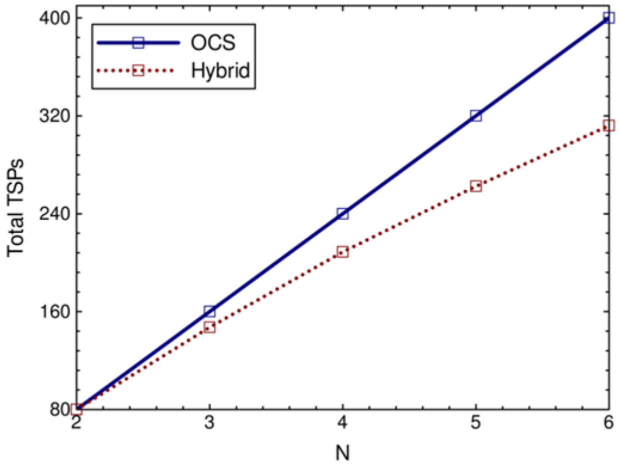


Figure 6.9: TSPs as a function of the QoS per virtual link.

It can be appreciated how the number of TSPs grows linearly in the OCS case, since in a point to multi-point communication a virtual link always accounts for two TSPs, one at the source and another at destination. As for the hybrid case, it can be appreciated that higher reductions can be obtained when the amount of nodes involved in the point to multi-point communication increase, with the obtained reductions almost doubling when going from 4 to 6 nodes.

6.3. Performance evaluation on scientific applications

In this section, an evaluation of the performance impact of the new proposed network data plane for LIGHTNESS to scientific applications is provided. As described previously in D3.2 [3], this new data plane consists on the AoD network, the optical ToR, and the hybrid NIC at the servers. This is the first time that this new data plane is evaluated on scientific applications, and thus it provides a value input for LIGHTNESS.

A simulator framework has been used for this purpose which consists on the LIGHTNESS simulator and other existing tools. This simulator has been developed during the second year of the project in the context of WP2 simulation activities. For this evaluation, it was needed to extend the LIGHTNESS DCN simulator in order to be able to model the new optical components such as the Optical ToR and the hybrid NICs. Additionally, the SDN control plane that manages this network has been also modeled in the simulator. A detailed description of the different modules developed in the simulator will be described later on.

A general overview of the simulation framework is shown in Figure 6.10. In order to be able to assess the performance on scientific applications, the simulation framework consists on the following tools:

- **Extrae** is a package developed at Barcelona Supercomputing Centre (BSC) that can instrument applications based on MPI, OpenMP, pthreads, CUDA, etc. The information gathered by Extrae typically includes timestamps of events of runtime calls, performance counters, and source code references. Additionally, Extrae provides its own API to allow the user to manually instrument the application of interest. In the case of our simulation framework, apart from tracing applications from the High Performance Computer (HPC) domain, the Extrae API has been used to trace BigData applications as well.
- **Dimemas** allows us to replay the traces collected by Extrae. It also provides as an output a new trace that can be analyzed once the simulation has finished.
- **Paraver** is a visualization and analysis tool developed at BSC as well. Paraver is very flexible and can be easily extended to support new performance data or new programming models, without changes to the visualizer. The tool offers a large set of time functions, a filter module, and a mechanism to combine two time lines, which allows displaying a huge number of metrics with the available data.
- **LIGHTNESS simulator** models the DCN architecture proposed by the LIGHTNESS project by modelling one by one the new proposed LIGHTNESS modules. It collects every packet from Dimemas and models the transfer through the optical network. Both simulators are running at the same time exchanging messages between them and advancing the application execution.

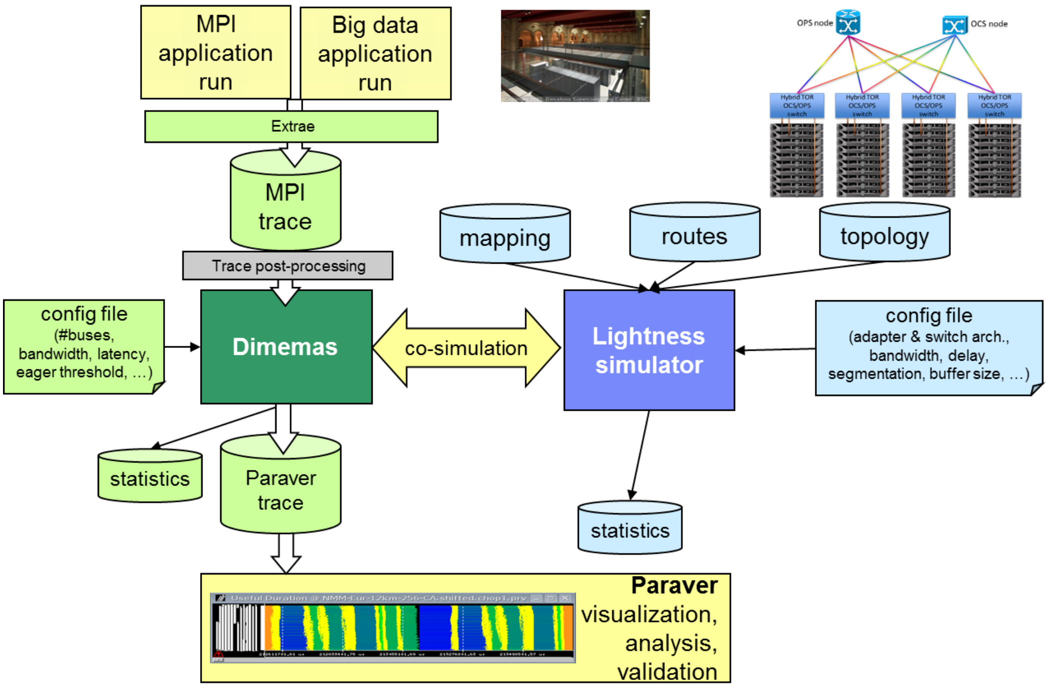

MPI application run
Big data application run
Extrae
MPI trace
Trace post-processing
config file (#buses, bandwidth, latency, eager threshold, …)
Dimemas
co-simulation
Lightness simulator
config file (adapter & switch arch., bandwidth, delay, segmentation, buffer size, …)
mapping
routes
topology
OPS node
OCS node
Hybrid TOR OCS/OPS switch
Hybrid TOR OCS/OPS switch
Hybrid TOR OCS/OPS switch
Hybrid TOR OCS/OPS switch
statistics
Paraver trace
statistics
Paraver visualization, analysis, validation

Figure 6.10: Simulation framework

A flattened optical DCN is constructed by plugging optical devices into an optical backplane to form an AoD node. The AoD, orchestrated by the SDN control plane, can provide dynamic DCN connectivity between ToRs over different optical technologies such as OPS and OCS. Different arrangements of input/output and modules can be constructed by dynamically setting up appropriate cross-connections on demand in the optical backplane according to different applications' requirements.

For illustration purposes, Figure 6.11 shows the diagram of the AoD for a simple DC that consists of four servers, two optical ToRs, one OCS, and one OPS. Optical ToRs are connected to the OCS and on top of that is placed the OPS. Also, note as shown in the figure, two servers are connected to a ToR through the hybrid NIC. In addition, the two NICs, connected through a ToR, are also connected each other with a direct optical cable to allow intra-rack communication. As it can be seen, all of these components have an *OF Agent* that interacts with the SDN controller to configure properly the DCN switches. In particular, it sets up the proper connections from the input to the output ports in the OCS and loads the look-up tables (LUT) in the OPS. Traffic flows within the servers of the same rack do not need to contact the SDN controller as the NIC can use the direct optical connection to communicate with the remote NIC. Notice that the optical ToR may be used as a traffic aggregator of different flows coming from the NICs.

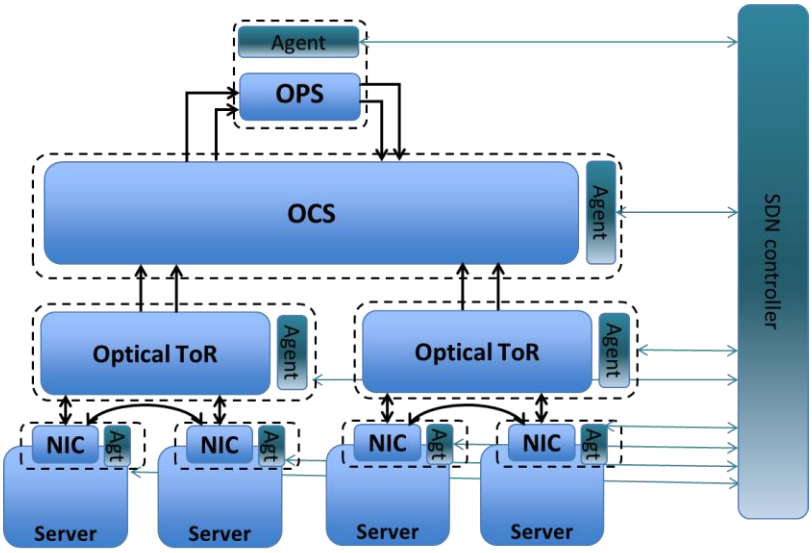


Figure 6.11: Architecture on Demand overview

6.3.1. Simulator modules

The modules built in the LIGHTNESS simulator have been implemented using OMNEST, which is the commercial version of OMNeT++, a framework for discrete-event network simulation and very popular in the community for building network simulators. To give a brief introduction, OMNEST models are based on simple modules, written in C++, using the simulation class library. These modules can have several parameters in order to specify their behaviour and they implement the algorithms to define the model's operation. Simple modules might be further grouped into compound modules. This approach enables the user to reuse and combine components in a flexible manner. The input and output interfaces of modules are represented by gates. An input gate and output gate can be linked by a connection. Connections can be characterized by parameters such as: propagation delay, data rate, bit error rate, and packet error rate, etc.

The modules in a simulator can communicate with each other through messages. These are sent via gates, or directly to destination modules and can represent frames or packets in a computer network, jobs or customers in a queuing network, or other types of mobile entities. Messages can contain arbitrarily complex data structures, defined by the user, such that to support the desired operation for the model. We proceed now on detailing the modules of the simulator and their functionalities.

❖ SDN controller module

The LIGHTNESS simulator implements the basic functionality of an SDN controller. The basic functionality is to set up the optical devices in the LIGHTNESS network. For this purposes, it has direct connections to every optical device in the network. Figure 6.12 shows the diagram of the compound module in OMNEST that models the SDN controller. The SDN controller receives from each NIC requests to setup the different optical devices (OPS, ToR, OCS), by means of the OpenFlow protocol following a bottom-up approach, that is, the configuration of DCN switches is triggered on-demand by new traffic flows generated within servers. The SDN controller accesses the different devices to setup the LUT or parameters of the devices. For this purpose, it has *M* ports to connect to each NIC, and then it has ports to connect independently to each of the three optical devices, *sdnops, sdntor, sdnocs*. The number of *sdnops, sdntor*and *sdnocs* ports depends on the number of OPS, ToRs and OCS switches used.

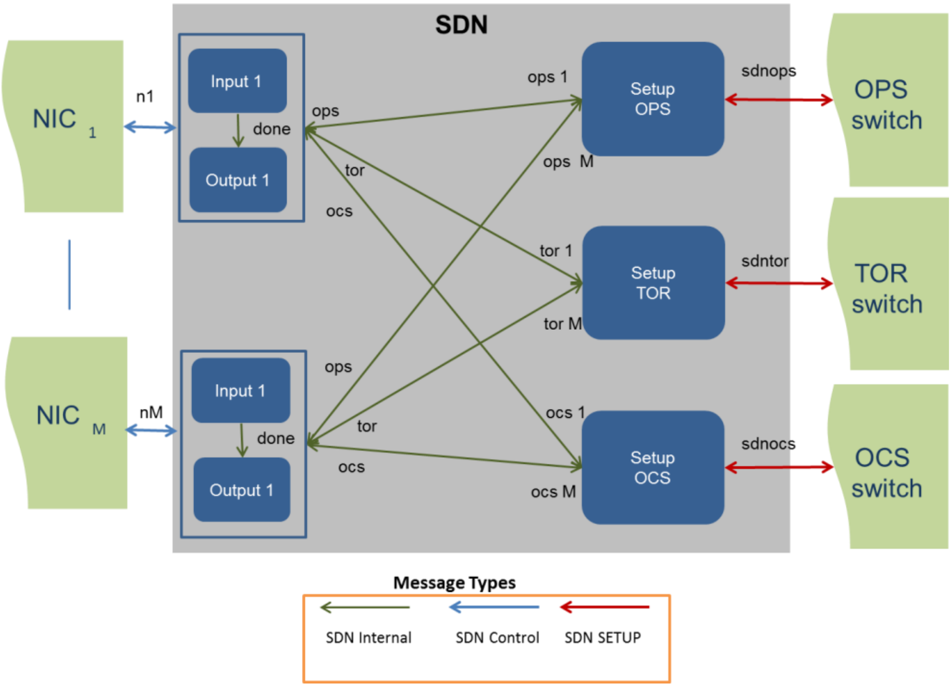


Figure 6.12: SDN controller design in the LIGHTNESS simulator

As shown in Figure 6.12, several sub-modules compose the SDN controller module, consist of:

- *Input*: It connects directly to a NIC and receives requests from the NIC to establish connections in the optical network. If the request is for only the OCS then it creates a request to the ToR and OCS. In addition, if the request also involves the OPS then, besides the TOR and OCS, it also creates a request to the OPS. All of these requests are sent at the same time to each optical device. After all the requests have been forwarded, it waits until a confirmation from the optical devices is received. Once all the confirmations have been received then it informs that the corresponding connection request has been already established.
- *Output*: It is directly connected to the NIC in order to inform the NIC when the setup has been done. It receives the confirmation from the Input module.
- *SetupOCS, SetupTOR, SetupOPS*: This ports manage the setup to each optical device, OCS, ToR, and OPS, respectively. It collects the requests from the Input and forwards them to the corresponding optical device. In case there are multiple requests coming at the same time, then it buffers them in order to forward them when the output port becomes available.

Additionally, several messages have been defined to handle SDN related messages. More specifically:

- *SDN Internal*: used to send requests and confirmations between the different sub-modules of the SDN controller.
- *SDN Control*: These are the packets that the NIC is sending to the SDN controller in order to request a setup to the optical devices.

- *SDN Setup*: These are the particular packets that the SDN controller sends to the different optical devices.

Finally, the following parameter is used in the setup of the SDN controller module:

SDN Delay: This corresponds to the delay to process the requests from the NICs before it forwards the request to the corresponding Control OPS, OCS or ToR.

❖ Hybrid NIC module

The Hybrid NIC performs the electronic/optical conversion between the server and the optical network in the LIGHTNESS network. Figure 6.13 shows the diagram of the submodules implemented in OMNEST. The hybrid NIC has connections with the server and also two connections with the optical network which corresponds to connections to other NICs in the same rack and to the connection with the ToR in order to access to other racks in the system. Packets to/from the server to the hybrid NIC are transferred through the PCI express using a DMA IP Core in the hybrid NIC. Packets are then first stored to the hybrid NIC internal memory before being transferred to the optical network following the store-and-forward flow control.

In addition, the hybrid NIC triggers a new connection establishment to the SDN controller when a new traffic flow arrives and there is no path already set up in the network. Once the connection is established in the network, the hybrid NIC can release the connection later on depending on various strategies. The release of connections depends on whether the OPS is being used or not, and also on the number of messages that has been already transmitted. In the experiments that will be presented later we evaluate different strategies on this regard.

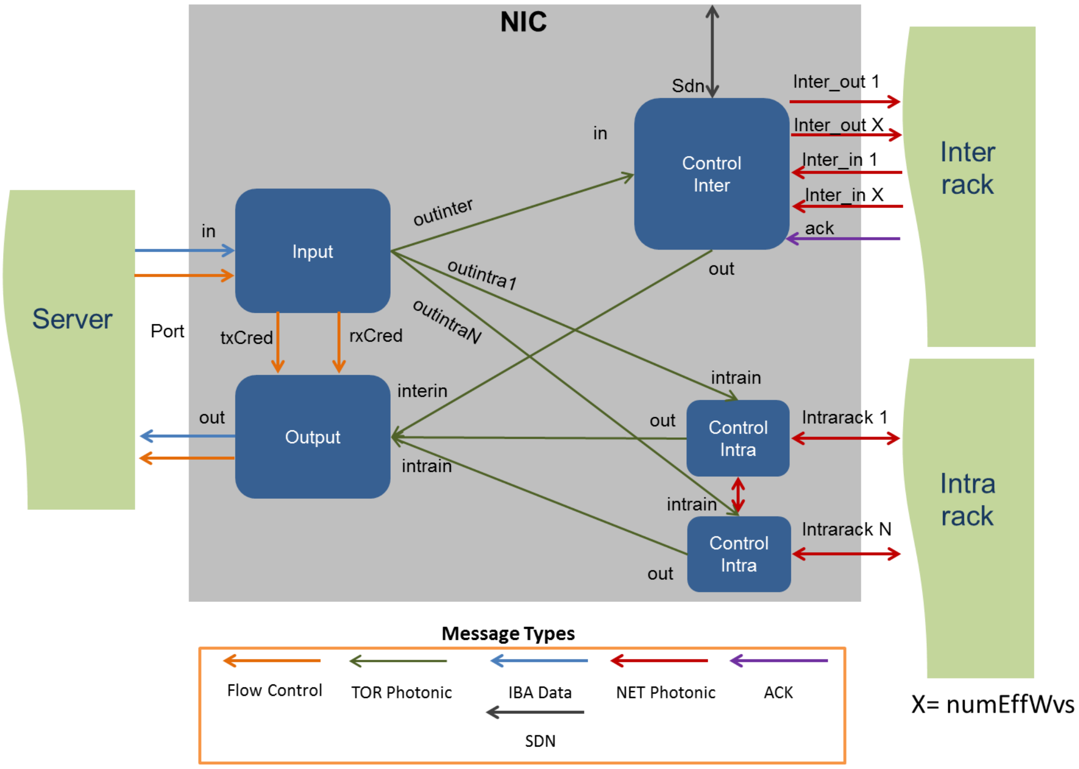


Figure 6.13: Hybrid NIC design in the LIGHTNESS simulator

The sub-modules of the hybrid NIC are described as following:

- *Input*: It is directly connected to the server. It receives data packets as well as flow control packets from the server. It is also directly connected with the Output sub-module in order to communicate the flow control information. Additionally, it also distributes the traffic to the corresponding output looking at the destination server identifier of the received packets. In case that the destination is placed in another rack in the system then it forwards the packet to the Control Inter sub-module. On the other case, it forwards the packet to the corresponding Control Intra module which is connected to a NIC in the same rack.

- *Output*: It is directly connected to the server. It sends data packets to the server and also flow control packets. It can receive packets from either the Control Inter or any of the Control Intra sub-modules.

- *Control Inter*: This sub-module is managing the traffic that communicates with other racks in the system through accessing the ToR, OCS, and OPS optical devices in LIGHTNESS. For this reason, it has a direct connection (SDN port) with the SDN controller in order to configure these optical devices. Also, it is connected with one or multiple wavelengths with the ToR. Normally, it will be using only one wavelength, but it could be extended to multiple wavelengths. There are output ports and also their corresponding input ports. Finally, there is a specific port (ACK) that it is directly connected to the OPS in order to receive the ACK/NACK signals.

- *Control Intra*: It manages all the traffic that flows to the different NICs. There are N ports to connect to the N NICs, one port per remote NIC. There is only one wavelength per port. In addition, it also performs a switching functionality for packets that are not directly destined to the current NIC. In this case, it forwards the packet to the corresponding Control Intra module.

The messages that have been defined to handle SDN related messages are:

- *Flow Control*: the packets belonging to the flow control with the server.

- *NIC Photonic*: internal packets of the NIC that are used to communicate packets among the different submodules of the hybrid NIC.

- *App Data*: packets from the application running in the server, which will be converted to optical packets in order to be sent through the optical network.

- *NET Photonic*: Packets that are destined to the optical network that connects different racks.

- *ACK*: This is a particular packet that comes from the OPS in order to acknowledge the successful forwarding of the packet to the OPS output.

- *SDN*: packet that connects to the SDN controller and sends requests for connections in the optical network. It also receives the ACKs when the connections have been established.

The following parameters are used to setup the SDN controller:

- *NICDelayOptical*: delay to process packets in the NIC and performing the Hybrid NIC to optical conversion.

- *NICDelayHybrid*: delay to process packets in the NIC and performing the Optical to Hybrid NIC conversion.

- *NWavelenghts*: The number of wavelengths used to connect the hybrid NIC with the ToR.
- *NNICs:* The number of NICs that are placed in the same rack.

❖ OPS module

The OPS module was implemented in the LIGHTNESS network simulator following the architecture described in the previous deliverable D3.2. Figure 6.14 shows the sub-modules developed in OMNEST to model the OPS architecture. This implementation is an extension of the previous implementation described in deliverable 2.3. For this reason, we are going to highlight only the changes made from our previous architecture already described. As shown in Figure 6.14, the OPS design is being decomposed into M modules. Each of these modules is managing packets from a specific input fibre. There are several Photonic Contention sub-modules to handle request to the corresponding output, and also there are N wavelengths ports that manages the wavelengths that are coming from this particular fibre.

The interesting part that differs from the previous OPS design is that now the Output sub-module has (m) outputs instead of only one as before. The reason for this is because in the new OPS architecture there is a specific fibre to every of the possible destinations. In addition, there is a port that connects to the SDN controller in order to set up the OPS routing tables. This port it is not shown in Figure 6.14.

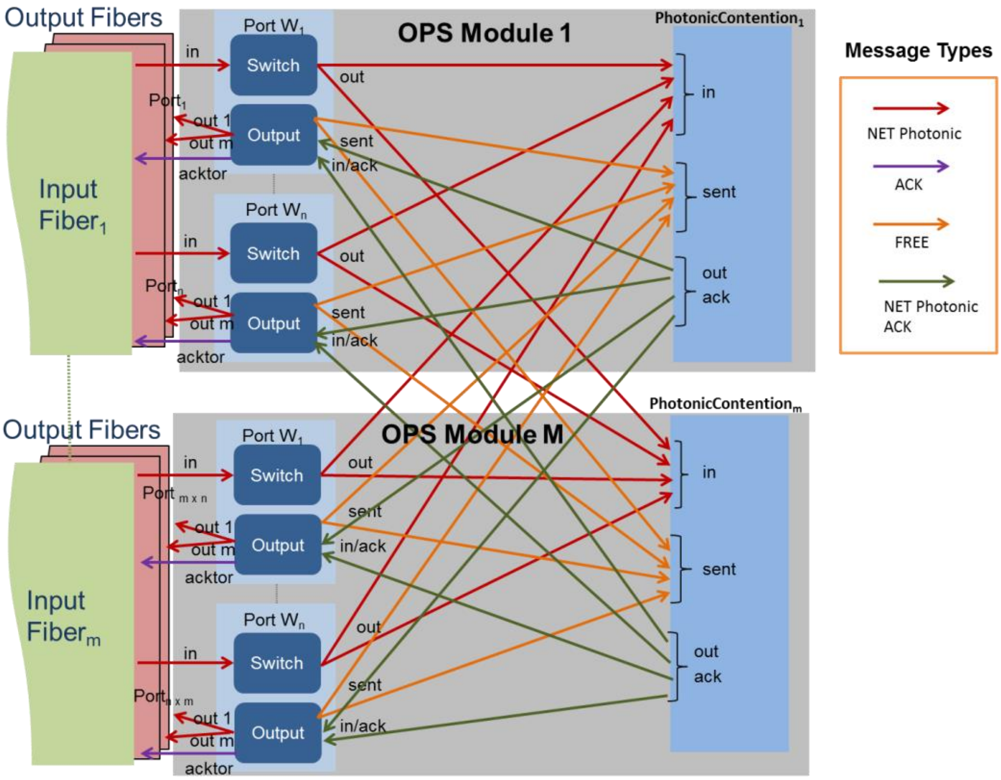


Figure 6.14: OPS design in the LIGHTNESS simulator

In order to illustrate the connections required between the OPS and the OCS and the ToRs, Figure 6.15 shows a diagram of a simple network consisting of three servers distributed each one in an independent rack. Therefore, we have three ToRs as well, each one with a single server. Thus, the OPS consists of three modules where each one is processing incoming packets from a single fibre. In each of the OPS module there are two output fibres in order to allow packets from different NICs to communicate to the destination NIC without collisions. For example, in Module 1, the fibre f2 is used by Server2 to transfer packets to Server1 and at the same time the fibre f3 could be used by Server3 to communicate to Server1 as well without any optical packet collision because each packet is using a different fibre. It is important to highlight the number of required connections needed in the OCS in order to connect the OPS to the corresponding servers. As it can be observed, the number of ports required in the OCS is tripling the number of servers.

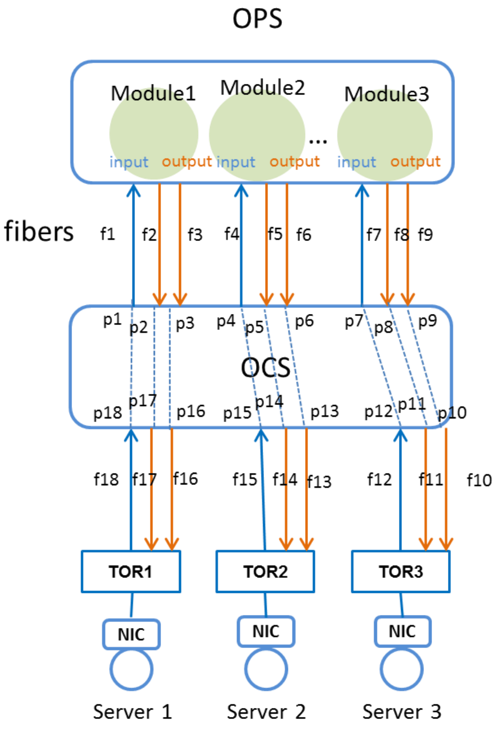


Figure 6.15: Required connections in the AoD in order to connect the OPS

6.3.2. Simulation scenario and results

In this section, we state the scenario utilized to evaluate the impact on the performance of scientific applications of the new proposed LIGHTNESS DCN. This evaluation has been performed by using our simulation framework based on OMNEST. The evaluation has been carried out using real traces from HPC applications. These traces have been obtained in the MareNostrum III supercomputer located at BSC. This supercomputer has a peak performance of 1.1 Petaflops. It contains 100.8 TB of main memory and 3,056 compute nodes. Each compute node contains two Intel SandyBridge-EP E5-2670/1600 20M 8-core at 2.6 GH processors.

For this evaluation we have selected the following parallel scientific applications using MPI:

- **MILC**: It performs 4D-SU3 lattice gauge computations. A problem size per MPI process of 8x8x8x8 was used in the experiments.
- **MINIMD**: A molecular dynamics application based on LAMMPS. A problem size per MPI process of 32x32x32 was used in the experiments.
- **SNAP**: It represents a particle transport application. A problem size per MPI process of 4x4x4 was used in the experiments.

All applications run on 16 processes, allocating only one application process per server. The model of the system used in our simulator is a small version of the MareNostrum system that consists of 16 racks and 18 servers per rack. In the experiments, we have used 16 racks with one process per rack, where each process is connected to a NIC and all the NIC are connected to one OCS and this OCS is connected to the OPS.In Table 6.1 we report the employed simulation parameters as well as the network model for the DC based on the LIGHTNESS network.

For simplicity, we are assuming that the ToR in the LIGHTNESS network does not introduce any significant overhead. The parameters for the OPS and the Hybrid NIC were taken from experimental measurements on the real optical devices described in the previous chapters. The OCS timings were collected from the Polatis 6000 optical switch. In addition, for the sake of simplicity in the experiments, only one port is used for intra- and inter-rack communication in the hybrid NIC.

Parameters	Value
Cable delay (50m)	250ns
OPS switching time	30ns
OPS update LUT time	214ms
Hybrid NIC to optical latency	7.45us
Optical to Hybrid NIC latency	4.51us
Hybrid NIC inter-rack bandwidth	10Gbps
Hybrid NIC intra-rack bandwidth	10Gbps
Hybrid NIC switching latency	68ns
Number of inter-rack ports Hybrid NIC	1
Number of intra-rack ports Hybrid NIC	1
Read-Write bandwidth Server to Hybrid NIC	10.46Gbps
OCS switching time	25ms
SDN processing time	512ms

Table 6.1: Simulation parameters

Figure 6.16 depicts a network with all the components of the AoD network modelled using the Omnest simulator. In this example we are showing four hosts connected through Infiniband (IB) switches (SW_1 to SW_4) to hybrid NICs. Each NIC has a connection with the SDN controller in order to be able to request connections with the OCS and the OPS. The IB switches are in charge of the packetization of the application messages. Before transmitting a message, each NIC requests for a connection with the OCS, then, once a connection is established, the SDN notifies the NIC so it can start the communication through the established channel. If the NIC is going to use also the OPS to communicate to other hosts, the SDN also sends a request to the OPS and confirms when all the connections are established.

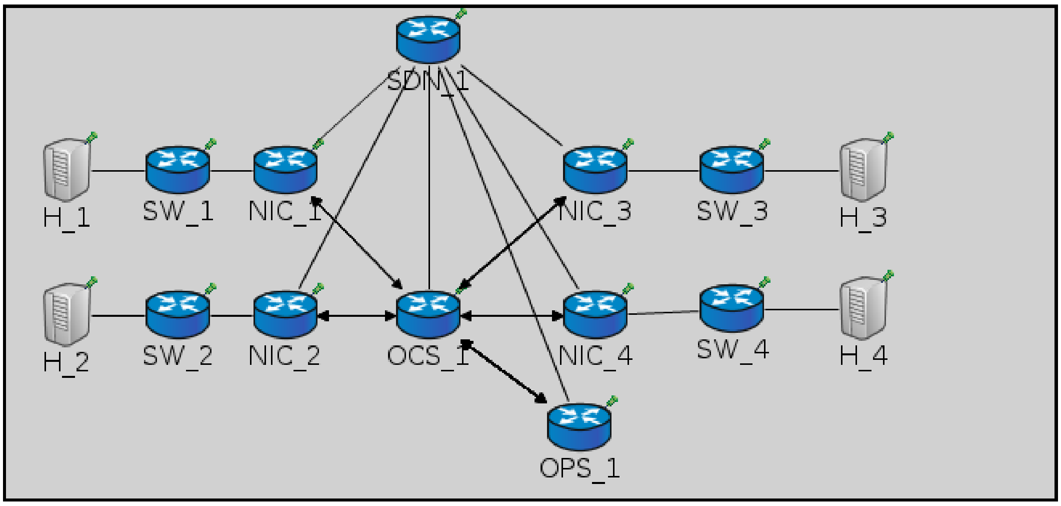


Figure 6.16: Network representation in the Omnest simulator. One host per NIC is being used as well as one OPS, one OCS and an SDN controller

When the NICs are communicating using only the OCS, they need to release the connections so other NICs can also use those released connections. In the experiments shown in Figure 6.17, two different strategies were used:

a) The NICs release the connections after five packets were transmitted or when their queues are empty (*OCS-MessageBurst|EmptyQueue*). This method allows other NICs to ask for connections without being blocked until a NIC empty its queue. However, this method could negatively affect the execution time because the number of requests to the SDN could increase and the cost of setting up connections is very high (512ms plus the OCS switching delay).

b) The NICs release the connections only when their queues are empty (*OCS-EmptyQueue*). This method reduces the number of disconnections and reconnections since a NIC could first empty its queue before releasing the connection. As the cost of requesting and setting up a connection through the SDN is normally very high, this methodology normally reduces the execution time of applications as can be observed in Figure 6.17. Note that other strategies can be implemented and explored in order to maximize the performance of HPC applications.

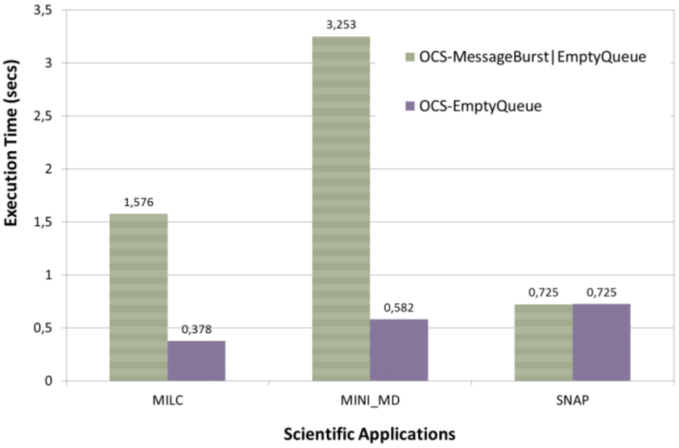


Figure 6.17: Comparison of two different disconnection mechanisms when using the OCS

To further investigate the impact of the way the NIC transmits the traffic on the applications' performance, Figure 6.18 shows a comparison of different scientific applications where the NICs transmit packets using different components:

- ***OCS-EmptyQueue***: in these experiments the packets are transmitted using only OCS and connections are released when the NIC's queues are empty.
- ***OPS:*** the NICs transmit the packets through OCS and then they use OPS to reach other hosts.
- ***Mixed:*** where NICs from 1 to 8 transmit messages just through OCS and NICs from 9 to 16 transmit messages first through OCS and then they use OPS.

As shown in Figure 6.18, using OPS could reduce the execution time when the number of packets transmitted is high (right secondary axis), which is the case of MILC and MINI_MD applications. On the other hand, when the number of messages that are going to be transmitted is low (SNAP), the extra configuration delay for the OPS could increase the execution time. The mixed configuration could bring more penalties to the execution time because the extra time used in some NICs to setup the OPS could delay some packets that trigger the other communications, and then delaying the request for new connections through the OCS. However, this impact is related to the application behaviour.

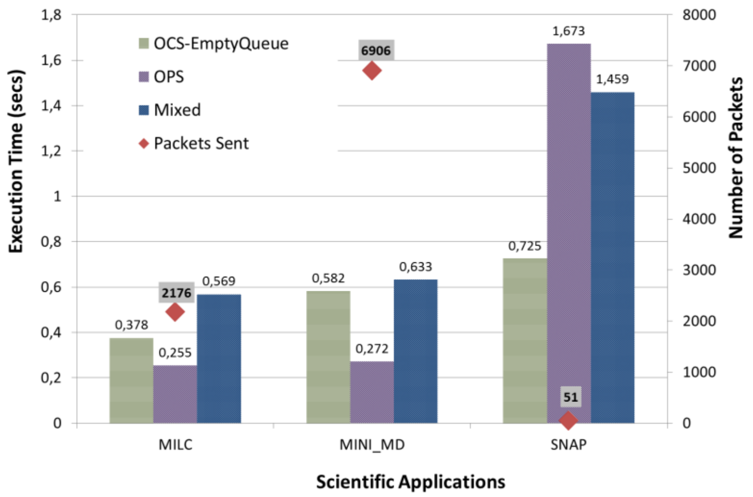


Figure 6.18: Comparison of performance when applications use the OCS, the OPS and a combination of them

In light of this, in Figure 6.19 we analyse why using a mixed configuration could have a more negative impact than using just the OCS or the OCS with the OPS to send packets in a network infrastructure. An example of two packet transmissions is shown: one packet from NIC0 to NIC1 and another one from NIC1 to NIC2. The transmission of the first packet triggers the transmission of the second one. For the sake of simplicity we assume a zero packet transmission time through the network. Only the delays to configure the network are shown.

The first case (the graph on top) corresponds to using only OCS. In this case, NIC0 needs to establish a connection to the OCS, and then after that the NIC1 is establishing connection to NIC2. Therefore, it results in a total delay of 1,074msto transmit the packet from NIC1 to NIC2. The next case corresponds to using OPS for all the NICs. In this case, both NIC0 and NIC1 can establish the connection in the OCS at the same time. It is assumed that NIC1 was transmitting to NIC2 before so it could trigger the network setup. Both NICs pay a 751ms delay at the beginning, but after that, packets can be transmitted faster as there is no need to further setups. The latter case (bottom graph) shows the case of mixing OCS and OPS. In this case NIC0 uses OPS whereas NIC1 uses only OCS. As seen, NIC0 has to suffer a 751ms connection establishment delay to send to NIC1 and then NIC1 has to again pay for the OCS setup of 537ms, resulting in total a higher delay to transmit the packet from NIC1 to NIC2 than the other cases described above.

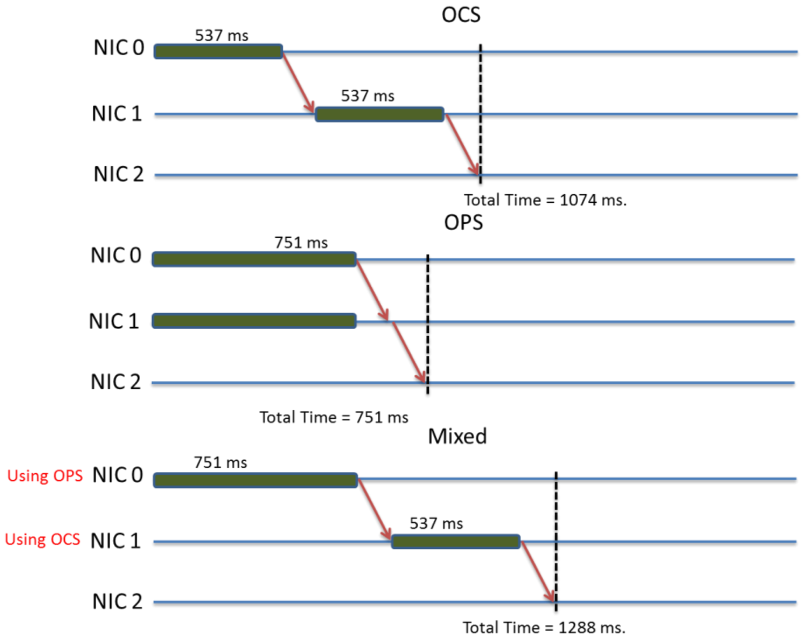


Figure 6.19: Analysis of delays that are added because of reconnections

7. Design, implementation of intra- DCN to inter-DCN interface

In this section, an all-optical multi-dimensional and programmable solution for both intra- and inter-DCN communications is proposed and demonstrated. The multi-element fibres-based SDM technology provides low-loss and easy to handle links between ToRs and cluster fibre switches. SDM and TDM technologies are used in an OCS manner to realize intra-DCN communications. The inter-DCN communication is realized by converting SDM signals to WDM signals using the SDM/WDM converter, and then transferring them to another DC. The NFP enables the DCN to realize different network functions.

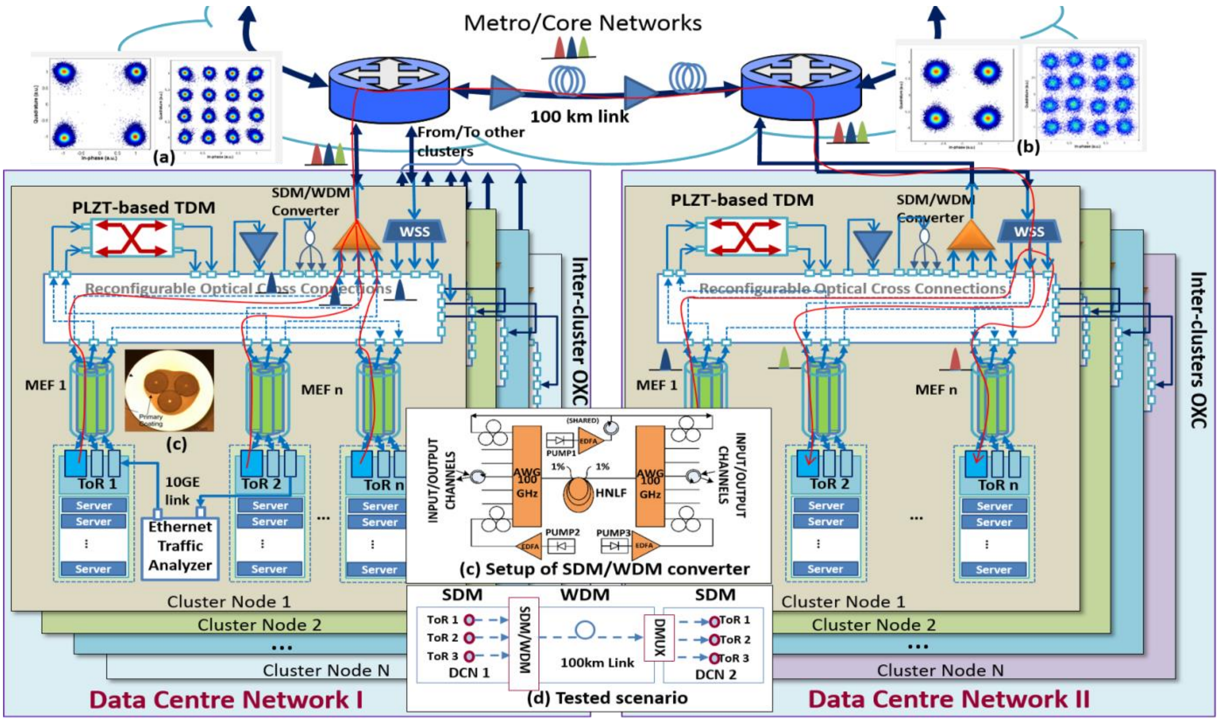


Figure 7.1: Proposed solution for intra-DCN communication (SDM+TDM) and inter-DCN communication (SDM+WDM)

Figure 7.1 shows the proposed solutions for both intra and inter-DCN communications. Each DCN consists of clusters with tens/hundreds of racks networked together and each rack is filled up with tens of servers. Servers are interconnected to ToRs via 10GE optical links. In our design, ToRs play a pivotal role in both inter- and intra-DCN communications. Figure7.2 shows the design of the proposed FPGA-based ToR. The ToR, implemented using FPGA optoelectronics (HTG XilinxV6 board), parsed the input traffic from servers and sent them out through different transmitters according to their destination. Programmable slotted-TDM/Ethernet over SDM signals are sent out through two SFP+ transceivers for intra-cluster communication. Another four transmitters feed the traffic to a BVT to provide a link of up to 320 Gbit/s for high-capacity inter-cluster and inter-DCN communications. All the transmitters use fixed wavelength lasers to avoid expensive temperature control. Then all the transceivers on the ToRs are connected to a LPFS with MEFs.

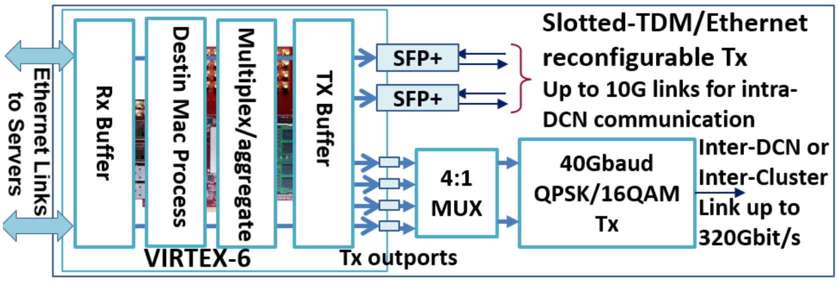

Ethernet Links to Servers
Rx Buffer
Destin Mac Process
Multiplex/aggregate
TX Buffer
VIRTEX-6
SFP+
SFP+
Slotted-TDM/Ethernet reconfigurable Tx
Up to 10G links for intra-DCN communication
4:1 MUX
40Gbaud QPSK/16QAM Tx
Inter-DCN or Inter-Cluster Link up to 320Gbit/s
Tx outports

Figure 7.2: FPGA-based ToR providing slotted-TDM/Ethernet signals for intra-DCN and BVT for inter-DCN

Directed cross-DCN ToR-to-ToR connections are setup through the metro/core network using an all-optical SDM/WDM converter. The experimental setup of the SDM/WDM converter is shown in Figure7.1 (c). The converter is a fibre-based dual-pump four-wave-mixing device [21], and uses a shared-pump bi-directional configuration that allows conversion of two SDM channels at the same time. It provides polarization-, rate- and modulation format-independent operation. Using this converter, contiguous 3-carrier super-channel signals are obtained from three same-l SDM signals (either PM-QPSK or PM-16QAM signals were used in our experiments), which are subsequently launched into the metro/core networks. Then the super-channel signal is dropped at the edge node and sent to another DCN. By adopting optical de-multiplexing, each carrier is sent to different ToRs. The tested inter-DCN scenario is shown in Figure 7.1 (d). The inter-DCN link adopts either PM-16QAM or PM-QPSK signals at 40 Gbaud to trade off capacity against transmission distance. 16QAM/QPSK constellations of the CH1 signals are shown in Figure 7.1 (a) for back to back and (b) after 100 km transmission.

For the 40 Gbaud QPSK signal, the performance of the setup is analysed by measuring the BER to OSNR curve. The results are shown in Figure 7.3. The SDM/WDM converter introduces an OSNR penalty of about 1.61 dB and 2.53 dB at 1E-3 for the two converted channels: CH1 and CH3. Another SDM signal without wavelength conversion is put in the CH2 wavelength slot. After 100 km transmission, the CH1 signals are transferred back to the ToR in another DCN. The penalty of the ToR-to-ToR connection is about 3.28 dB.

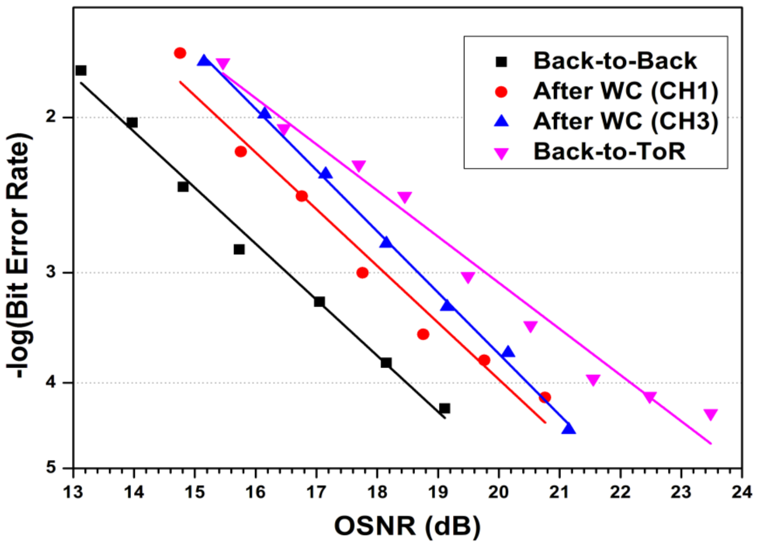


Figure 7.3: OSNR vs. BER for 40Gbaud PM-QPSK signals for inter-DCN communication

8. Conclusions

This document presents the final detailed implementation results and evaluation results of the LIGHTNESS data plane elements. Following the previous design and implementation reported in D3.1 [2] and D3.2 [3], experimental assessments of the prototypes including programmable optical NIC, optical ToR, OCS and OPS, as well as interfaces between switching nodes are presented. The implementation schemes of the OF-agents are explained in detail and the performances of the control interfaces are experimentally evaluated.

The implementation schemes of the LIGHTNESS data plane including both intra-cluster and inter-cluster scenarios are considered. As a comprehensive summary of the achievements in WP3, the overall data plane architecture is presented and the benefits of synthetic hybrid OCS/OPS architecture have been analyzed and experimentally assessed. A simulator framework has been developed for numerically evaluating the system performance. Moreover, the design and implementation of an all-optical multi-dimensional and programmable solution for the intra- DCN to inter-DCN interface has been given.The results reported in this deliverable will be further employed as a guideline for the experimental evaluation of the DCN integration in the WP5.

9. References

[1] C. Kachris and I. Tomkos, “A Survey on Optical Interconnects for Data Centers,” IEEE Communications Surveys & Tutorials, vol.14, no.4, pp.1021-1036, 2012.

[2] LIGHTNESS Deliverable D3.1 “Release of the design and early evaluation results of OPS switch, OCS switch, and TOR switch”.

[3] LIGHTNESS Deliverable D3.2 “Implementation results of the OPS switch, the OCS switch, and the TOR switch”.

[4] G. Bell et al., “Petascale computational systems,” IEEE Computer, Vol. 39, no. 1, p. 110, 2006.

[5] N.Amaya et al., “Introducing Node Architecture Flexibility for Elastic Optical Networks”, Optical Comm. and Networking, 2013.

[6] http://www.polatis.com/

[7] http://www.finisar.com/

[8]LIGHTNESS Deliverable D4.2 “The LIGHTNESS network control plane protocol extensions”.

[9] Y. Yan, G. Zervas, Y. Qin, B. R. Rofoee, and D. Simeonidou, “High performance and flexible FPGA-based time shared optical network (TSON) metro node,” Opt. Express, vol. 21, no. 5, pp. 5499–5504, Mar. 2013.

[10] W. Miao, et.al., Novel flat datacenter network architecture based on scalable and flow-controlled optical switch system, OE, 22 (3), 2465- 2472, 2014.

[11] N. Calabretta, W. Wang, T. Ditewig, O. Raz, F. Gomez Agis, S. Zhang, H.de Waardt and H. Dorren, “Scalable optical packet switches for multiple data formats and data rates packets,” IEEE Photonics Technology Letters, 22(7), 483-485, 2010.

[12] X. Qiu et al., “Fast synchronization 3R burst-mode receivers for passive optical networks,” J. Lightwave Technol., Vol. 32, no. 4, p. 644, 2014.

[13] X. Yin et al., “Experiments on a 10Gb/s fast-settling high-sensitivity burst-mode receiver with on-chip auto-reset for 10G-GPONs,” J. OPT. COMMUN.NETW., Vol. 4, no. 11, p. B68, 2012.

[14] R. Brian et al., “Integrated silicon photonic laser sources for telecom and Datacom,” Proc. OFC, PDP5C.8, 2013.

[15] S. Sakr et al., “A survey on large scale data management approaches in cloud environments,” IEEE Com. Sur. & Tut., Vol. 3, no. 13, p. 311, 2011.

[16] M. Faizul Bari et al., “Data Center Network Virtualization: A Survey,” IEEE Com. Sur. &Tut., Vol.15, p.909, 2013.

[17] LIGHTNESS Deliverable D4.4 “Preliminary LIGHTNESS network control plane prototypes”.

[18] J. Hamilton, “Data Center Network are in my way”.

[19] LIGHTNESS Deliverable D4.6 “SDN-based intra-DC network virtualization mechanisms”.

[20] N. Calabretta, R.P. Centelles, S. Di Lucente, H.J.S. Dorren, “On the performance of a large-scale optical packet switch under realistic data center traffic”, Journal of Optical Communications and Networking, Vol. 5, No. 6, p. 565-573, 2013.

[21] V. J. F. Rancano et al.,”100GHz grid-aligned reconfigurable polarization insensitive black-box wavelength converter,” Proc. OFC, JTh2A.19, Anaheim, 2013.

10. Acronyms

AGC	automatic gain control
AoD	Architecture on Demand
API	Application Programming Interface
AWG	Arrayed Waveguide Grating
BER	Bit Error Rate
BM-RX	burst mode receivers
BPF	band pass filter
BVT	bandwidth variable transmitter
CDR	clock data recovery
CFP	C Form-factor Pluggable
DC	Data Centre
DCN	Data Centre Network
DMA	Direct Memory Acces
DML	directly modulated laser
DWDM	Dense Wavelength Division Multiplexing
ECL	external cavity lasers
EDFA	Erbium doped fibre amplifier
ED	envelop detector
FBG	Fibre Bragg Grating
FDL	Fibre Delay Line
FPGA	Field-Programmable Gate Array
HPC	High Performance Computer
IB	Infiniband

LA	limiting-amplifier
LUT	Look-Up Table
MEF	multi-element fibre
NIC	Network Interface Card
OCS	Optical Circuit Switching
ODL	OpenDayLight
O/E	Optical-to-Electrical
OF	OpenFlow
OPS	Optical Packet Switching
OSNR	Optical Signal Noise Ratio
OVS	Open vSwitch
PCB	printed circuit board
PCE	path computation engine
PD	photodiode
QoS	Quality of Service
SDM	Space Division Multiplexing
SDN	Software Defined Networking
SLA	Service Level Agreement
SOA	Semiconductor Optical Amplifier
SFP+	enhanced small form-factor pluggable
SNMP	Simple Network Management Protocol
SSS	spectrum selective switches
TIA	trans-impedance amplifier
ToR	Top of Rack
TDM	Time Division Multiplexing
TL1	Transaction Language 1
TLP	Transaction Layer Packet
TSP	transponder
VDC	virtual data centre
VM	virtual machine
VN	virtual network

WDM	Wavelength Division Multiplexing
WSS	wavelength selective switches