International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Dynamic QoS Management in Reconfigurable Optical Data Center Networks for Cloud-Centric Computing

Sougata Bera[1], Chandi Pani[2]

[1]Assistant Professor, Department of CSE, Meghnad Saha Institute of Technology
[2]Professor, HOD, Department of ECE, Meghnad Saha Institute of Technology

Abstract:

The exponential growth of cloud-centric and real-time applications has revealed the inherent limitations of conventional electrical packet-switched Data Center Networks (DCNs), which suffer from bandwidth bottlenecks, high latency, and inefficient resource utilization. Optical circuit-switched DCNs offer superior bandwidth and data rates; however, their slow reconfiguration times restrict their applicability in dynamic, heterogeneous traffic environments. To address these challenges, this paper presents a Dynamic QoS Management Framework within a Reconfigurable Optical DCN Architecture designed for adaptive traffic control and efficient resource allocation.

The proposed system, termed the Passive Optical Data Center Switch (PODS), integrates Arrayed Waveguide Grating Router (AWGR) technology with an intelligent control unit capable of real-time traffic classification, buffer reuse, and heuristic-based path optimization. A loopback-enabled reconfiguration mechanism dynamically reallocates optical paths, alleviating congestion and ensuring consistent QoS across diverse service classes.

Extensive simulations—implemented in Python on the Google Colab platform—demonstrate that PODS achieves a 46.3% reduction in latency and a 5% improvement in network load compared to the existing Passive Optical Data Center Architecture (PODCA). A hardware prototype comprising 7 Top-of-Rack (ToR) switches and Raspberry Pi-based control modules, interconnected via 112 optical links across 16 wavelengths, further validates the design. Experimental results confirm an 18.3% reduction in blocking probability and zero blocking for high-priority traffic under full load, highlighting the architecture's real-time adaptability.

By combining passive optical components with intelligent QoS-aware control, PODS delivers a scalable, energy-efficient, and latency-optimized solution for next-generation data centers. This work establishes a foundation for self-adaptive optical DCNs capable of meeting the stringent performance and reliability demands of cloud-centric computing environments.

Keywords: - Optical DCN, AWGR, QoS Provisioning, Wavelength Assignment

1. Introduction

The rapid growth of data-intensive applications has driven the evolution of next-generation Data Centers toward adopting advanced architectures and novel technologies that enhance throughput, latency,

scalability, and power efficiency [1]. At the forefront of this transformation are optical Data Center Networks, which employ a range of optical switching technologies such as Semiconductor Optical Amplifier (SOA)-based switches, Micro-Electro-Mechanical Systems (MEMS) switches, and Arrayed Waveguide Grating Routers (AWGRs).

MEMS switches, utilized in systems like c-through [2] and Helios [3], offer reconfigurable optical switching through electromechanical actuation. However, their relatively high reconfiguration time limits their suitability for high-speed packet-switched DCNs. To overcome these constraints, several hybrid electro-optical interconnect architectures have been proposed [4–5], most relying on centralized schedulers that dynamically adapt to fluctuating network traffic. In contrast, RotorNet employs a predefined static scheduling scheme, making it less responsive to real-time traffic variations [6].

AWGRs, with their cyclic wavelength routing property, effectively resolve contention in the wavelength domain, enabling multiple inputs to reach the same output simultaneously. This feature has inspired several AWGR-based DCN architectures [7], including DOS [8] and Petabit [9–10], which incorporate Tunable Wavelength Converters (TWCs) for flexible wavelength allocation [11]. Although these systems deliver high performance, TWCs remain power-intensive, increasing energy overhead. To address this, the Passive Optical Data Center Architecture (PODCA) [12] integrates AWGR and TWC under a centralized Control Unit that dynamically assigns wavelengths, achieving packet latencies below 9 μs. While PODCA outperforms DOS and LIONS architectures in latency, it provides lower throughput compared to them [13].

To mitigate these trade-offs, this paper proposes a hybrid optical framework that synergistically combines the strengths of PODCA, DOS, and LIONS architectures to achieve higher throughput and reduced latency, thereby optimizing overall network performance. Furthermore, flexible Quality of Service (QoS) provisioning and scalable scheduling remain major design challenges [14–15]. Ongoing research focuses on addressing demand estimation, bandwidth reconfiguration [16–17], and switch interoperability in future DCNs [18–19]. Emerging paradigms such as synergistic switched-control optical networks, capable of nanosecond-level path configuration, and Software-Defined Networking (SDN)-enabled optical systems [20–21], have shown promise in improving contention resolution and enabling adaptive, flow-aware traffic management in the optical domain.

This paper delves into these advancements, presenting a comprehensive approach that amalgamates cutting-edge technologies and methodologies to enhance the performance and scalability of future data center networks. By leveraging the combined strengths of existing architectures and addressing key challenges in QoS and scheduling, we pave the way for the next generation of high-performance optical data center networks.

In this paper, the main contribution is to:

- **Propose a Re-configurable, Dynamic QoS Provisioned DCN Architecture**: We introduce a novel DCN architecture that integrates a scalable switch with dynamic QoS provisioning. This architecture features path reconfiguration through re-routing by different wavelength assignments and dynamic buffering to prevent packet loss, all within a single, cohesive module.
- **Priority Buffering Algorithm**: We propose a heuristic algorithm that runs over the DCN architecture where packets are initially stored in a priority buffer. If the requested service class buffer is unavailable, packets are redirected to the next available buffer before being forwarded. This

approach ensures efficient utilization of buffer resources and enhances the overall network performance.

- **Loopback Methodology for Path Reconfiguration**: To address wavelength unavailability, a loopback methodology [22] is incorporated that assigns different wavelengths for forwarding, which reconfigures the packet path, ensuring successful forwarding to the proper destination. This technique significantly reduces network congestion and enhances data flow reliability.
- **Dynamic Packet Transmission Priority**: Packet transmission priorities are dynamically assigned [16] to support mixed traffic. Higher-priority packets are forwarded directly to their destinations without retransmission or requiring the loopback method. This dynamic assignment minimizes delay and maximizes network efficiency.
- **Reduction of Packet Loss, and Blocking Probability**: Integration of all the above-mentioned methodologies, effectively reduces packet loss, and the blocking probability [23] of the entire network ensuring robust, efficient, and scalable DCN architecture.

The structure of the paper is as follows: Section II outlines the proposed PODS-based DCN architecture model and its operational mechanisms. Section III delves into the mathematical modeling employed for buffer and wavelength assignment. Section IV details the simulation process and presents the results of the analysis. In Section V, the test bench experimental model of the PODS-based DCN architecture, implemented using Raspberry Pi, is described. The experimental results are subsequently discussed in Section VI. Finally, Section VII provides the concluding remarks of the paper.

2. PODS-based DCN architecture model and Working Principle

The proposed PODS-base DCN architecture with a control unit [24] is shown in Fig. 1(a). The proposed model consists of ToR, AWGR, transmitter (TX) and receiver (RX) modules. Each ToR is connected to the end users or servers. The rest of the ports of the ToR are configured as input ports and output ports respectively. The input port of the AWGR is connected to the ToR via TX module and the output port of the AWGR is connected to the ToR via the RX module respectively as shown in Fig. 1(b) and Fig. 1(c). The TX module of each ToR consists of an electrical buffer (EB), optical channel adapter, optical label generator (OLG), packet encapsulator (PE), and electro-optic converter (lasers) to send the incoming packets to the AWGR through TWC. After passing through the TWC, wavelengths are combined by an Optical Multiplexer (OMUX) and finally reach the input port of the AWGR. The RX module consists of an Optical Demultiplexer (ODMUX) followed by an optical receiver called an optical-to-electrical converter (OE converter), electrical buffer (EB), and packet adapter (PA).

In this architecture, the generated packets from the server, first arrive to any input ToR (ToR_{IN}), then as per the service class of the packet, they are placed in the shared buffer marked as EB as shown in Fig. 1(b), similarly when the packets are out from the ToR_{OUT} port of AWGR, after demultiplexing the packets they are converted to optical to electrical and stored in a shared buffer of ToR RX module as shown in Fig. 1(c). As per the service class of the packet, the priority level of the buffer is set. In this paper we only consider four types of service class traffic, and based on it buffers are classified as high-priority real-time (HRT), standard-priority real-time (SRT), Earliest Deadline First (EDF), First-Come-First-Served (FCFS). Round-robin processing is used to handle packets from the buffer.

The number of buffers allocated to any service class is based on the incoming traffic requirement. Thus, the optimal architecture is determined by allocating a certain number of buffers under each service

class. The network's performance is then evaluated in software and hardware platforms in terms of maximum network load, latency, and blocking probability by reusing the buffer of a different service class (if it is empty) and rerouting wavelengths using the loopback method. Fig. 2 shows the flow chart for the proposed algorithm and the function of the PODS-based DCN architecture.

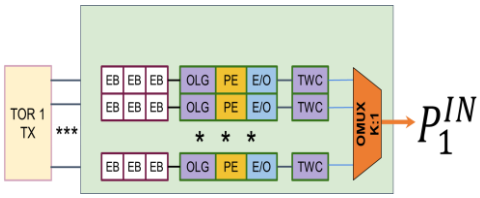


Fig.1b TX module of PODS AWGR

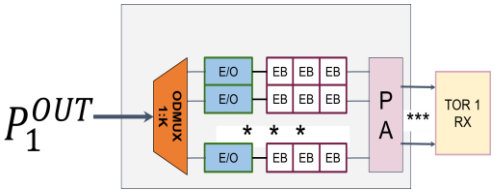


Fig. 1c RX module of PODS

Fig. 1a PODS-Based DCN Architecture

Fig. 1 PODS System model

To describe the performance of the system model we consider:

P is the total number of ports of AWGR. Out of P number of ports, some ports are connected to the ToR, and the rest of the ports are used for the loopback path for route reconfiguration. We denote W as the number of available wavelengths and let W=P× F where F ≥ 1 is an integer. Wavelength w is denoted as λ_w, where w is the wavelength index. The AWGR routes #wavelengths from an #input port to a specific #output port in a cyclic way and is denoted as λ_w^S means it is routed from input port S with wavelength w to reach the desired output port.

#wavelength=(# output port + # input port)%no. of port - 1 + f×no of port, where f ∈ F (1) [2]

Let B be the number of shared buffers in each ToR. B number of shared buffers is further subdivided into B_1, B_2, B_3 and B_4. Where B_1 number of buffers assigned for HRT, B_2 number of buffers assigned for SRT, B_3 number of buffers assigned for EDF and B_4 number of buffers assigned for FCFS. Each Packet having the same priority level is placed in the same priority buffer in a round-robin manner. The number of B_1, B_2, B_3, and B_4 buffers are finalized based on the arrival rate of a particular service type. In our model we consider a greater number of buffers in B_1 rather than B_2, B_3 and B_4 to reduce less loss in HRT traffic in terms of blocking probability. Additionally, packets may be placed in just below priority buffers, if the high priority buffers are unavailable. Depending upon the set of priorities, the packets are selected from the front of each buffer for transmission. Then the packets are ready for transmission and the wavelength is assigned depending on the output port number as per eq. (1). If the wavelength is not available for transmitting the packet to the destination port, the data is sent to the loopback port, and the data is forwarded to the output port with a different wavelength.

Table 1 shows the variables used to develop the analytical modeling of the proposed architecture.

Table 1. List of variables used in the architecture

Notation	Corresponding Meaning
P	Number of ports in AWGR
W	Number of available wavelengths in the system
F	Number of wavelengths used for every pair of input-output port
λ_w	w is the wavelength index
$\lambda_w^{S,D}$	$\lambda_w^{S,D}$ the wavelength w is selected for packet transmission from S^{th} input port P_S^{IN} to D^{th} output port P_D^{OUT}
$\lambda_w^{L,D}$	$\lambda_w^{L,D}$ the wavelength w is selected for packet transmission from L^{th} loopback input port P_L^{IN} to D^{th} output port P_D^{OUT}
B	The number of shared buffers is subdivided into B_1, B_2, B_3 and B_4
P_S^{IN}	S^{th} input port in AWGR
P_D^{OUT}	D^{th} output port in AWGR
$Pk_{pid}^{S,D,m}$	pid Packet transmits from m^{th} buffer of S^{th} input port P_S^{IN} to D^{th} output port P_D^{OUT}.
Pr_b[i]	Packet stored in the priority buffer Pr_b[i] where i ∈ B.
P_b	Blocking Probability

This flowchart illustrates the decision-making process for handling packets in a PODS-based Data Center Network based on service type and buffer availability.

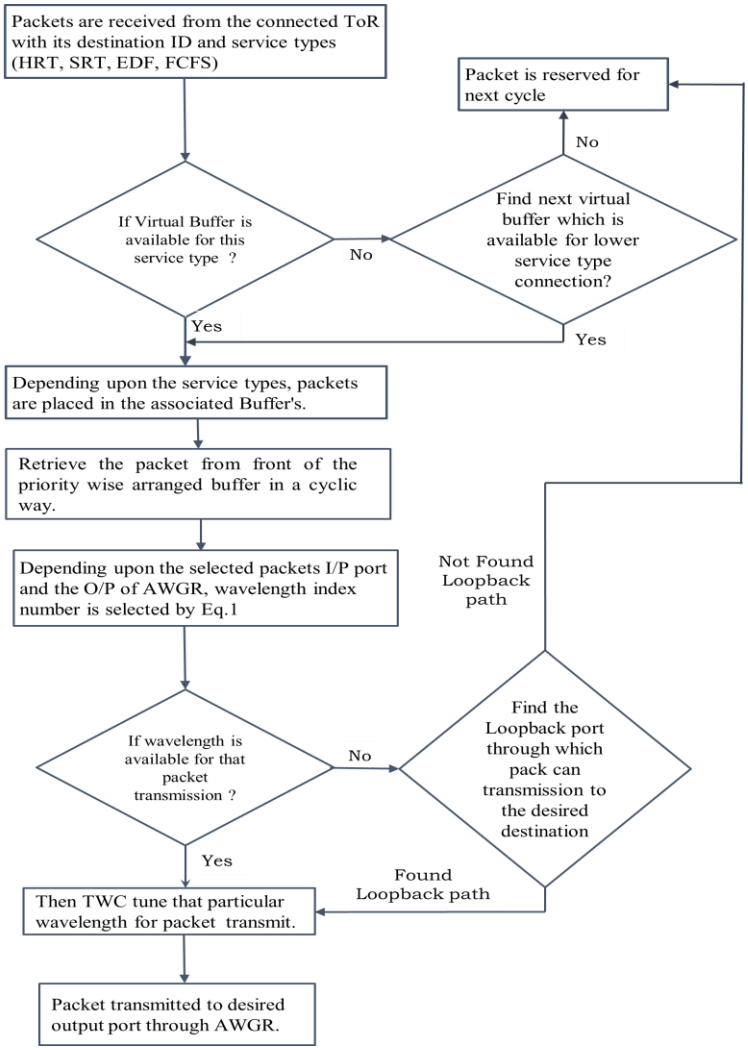


Fig. 2 Flowchart of PODS-Based DCN Architecture

3. Mathematical Modelling for Buffer and Wavelength assignment

To minimize the blocking probability (P_b), maximize Pr_b[i] i ∈ B

P_b is calculated as the ratio of the number of packets not successfully placed to the total number of packets generated:

$$P_b = \frac{total\_packets\_generated - successful\_placement\_counter}{total\_packets\_generated} \quad (2)$$

To maximize packet transmission Eq. 3 should be maximized subject to the condition described in Eq. 4, Eq.5, and Eq.6.

Maximize:

$$\sum_{S,D,m} Pk_{pid}^{S,D,m} \quad (3)$$

s.t.:

$$\forall\, \lambda_w^{S,D} = 0 \quad for\ the\ packet \quad Pk_{pid}^{S,D,m} \quad (4)$$

or,

$$\lambda_w^{S,D} = 0 \quad for\ the\ packet \quad Pk_{pid}^{S,D,m} \tag{5}$$

and $$\lambda_w^{S,D} = 0 \quad for\ the\ packet \quad Pk_{pid}^{S,D,m} \tag{6}$$

Algorithm of Packet Scheduling through Control Panel:

PROCESS 1: [PACKET CREATED AND STORED IN SELECTED BUFFER]

1. **Initialize Variables**
   a. successful_placements = 0 (packets), total_packets = 0 (Count of total packets generated)
   b. Pr_b[i] (Priority buffer, where i ∈ B, $1 \leq i \leq 4$)
   c. N_i (Number of buffers under each Priority buffer type Pr_b[i])
   d. $index_i$ (Next available position index in buffer type i)
2. **For each arriving packet:**
   I. Determine the `service_type` of the packet.
   II. Select the appropriate buffer type Pr_b [i] based on `service_type` and buffer priority.
   III. **If buffer Pr_b[i] has space ($index_i < N_i$):**
   IV. Store the packet in Pr_b [i] at $index_i$.
   V. Increment $index_i$ and Increment `successful_placements`.
   VI. **If buffer Pr_b [i] is full:**
   VII. Search for an available lower-priority buffer (Pr_b [j] ≠ Pr_b [i]).
   VIII. Store the packet in the first available buffer and Increment `successful_placements`.
   IX. Increment total_packets.

   Repeat steps I–IX for all ports.

Calculate the overall blocking probability P_b as the ratio of the number of packets not successfully placed to the total number of packets generated:

PROCESS 2: [ASSIGN WAVELENGTH FOR ROUTING]

1. Establish the retrieval order for packets from the buffer **Pr_b[i]**, where **i ∈ B**.

   $$F = \frac{number\ of\ wavelengths}{number\ of\ ports\ in\ AWGR}$$

   where the number of wavelengths is an integer multiple of the number of ports: number of wavelengths = (integer factor) × number of ports in AWGR.

2. For each **i** in **B**:
   a. Extract the leading packet from the priority buffer **Pr_b[i]** based on the input port index and output port index.
   b. Determine the corresponding wavelength index for the AWGR
   c. For each **f** in **F**:
      Compute the wavelength index as:

i. wavelength=(#output port+#input port)mod number of ports−1+f×number of ports, where f∈F
ii. If the computed wavelength is accessible, the TWC is adjusted accordingly, and the packet is scheduled for transmission.

d. If the desired wavelength is occupied for the given output port, the packet is redirected through a loopback port and reassigned to the output port using an alternate wavelength.
e. For each available loopback port, check the feasibility of using an alternative wavelength pair to forward the packet from the input port to the loopback port and then to the final output port.
i. If a valid wavelength pair is identified, configure the corresponding TWC and schedule the packet for transmission.
ii. If no suitable wavelength pair is available, the packet undergoes additional delay via the loopback path. If no transmission route is found, the packet is marked as Blocked.

4. Simulation Results

In the simulation, the packets are dynamically created with different service classes and assigned a random destination address. The interarrival rate follows poison distribution. To optimize the process, maximize the number of packets placed in the buffers considering the priorities and availability of the buffer.

Table 2: List of the parameter values considered for simulation

Sl No.	Name of Parameters	Values
1	Tuning time of tunable transmitters	8ns
2	The size of a packet	1500 bytes and interarrival follow Poisson distribution
3	Buffer size of each ToR	256MB
4	Total no. of buffer under each ToR	116
5	No. of service Class	4 (B1-(HRT), B2-(SRT), B3-(EDEL), and B4-(FCFS)
6	Distribution of buffer under each service class	B1=44, B2=34, B3=23 and B4=14
7	The traffic arrival rate per ToR	40 Gbps
8	Size of AWGR	128x128
9	No. of available wavelength	256
10	No. of AWGR port used for loopback	10% of 128 = 13
11	No. of ToR connected in AWGR port	115
12	Data Transmission speed achieved	(115x116x40Gbps) ≈533Tbps

We use Integer linear programming (ILP) as a mathematical tool to maximize the wavelength allocation for packet transmissions in a slot. We coded the algorithms in Python and executed them on Google COLAB in the Windows environment to find the performance of the proposed architecture and

compared it with PODCA architecture. For simulation purposes, we have assigned the transmission rate of a tunable transmitter (and wavelength capacity) to be 40 Gbps as per reference [12]. Table 2 shows the parameter values considered for simulation.

The total latency and network load of the framework is calculated as
Latency = Transmission time + Average queuing delay in the buffer.

$$\textbf{Network Load} = \frac{\textbf{NO OF PACKET CURRENTLY TRANSMIT}}{\textbf{MAX NO OF PACKET CAN TRANSMIT THROUGH AWGR PORT}}$$

To find the performance of the network, in terms of network load and latency, the arrival rate of the packet is increased from 0.1Gbps to 36Gbps. Fig. 3 shows the simulation snapshot of PODS-based DCN architecture as executed in the Google Colab environment. This figure illustrates the configuration and execution of the architecture, highlighting the key components and their interactions.

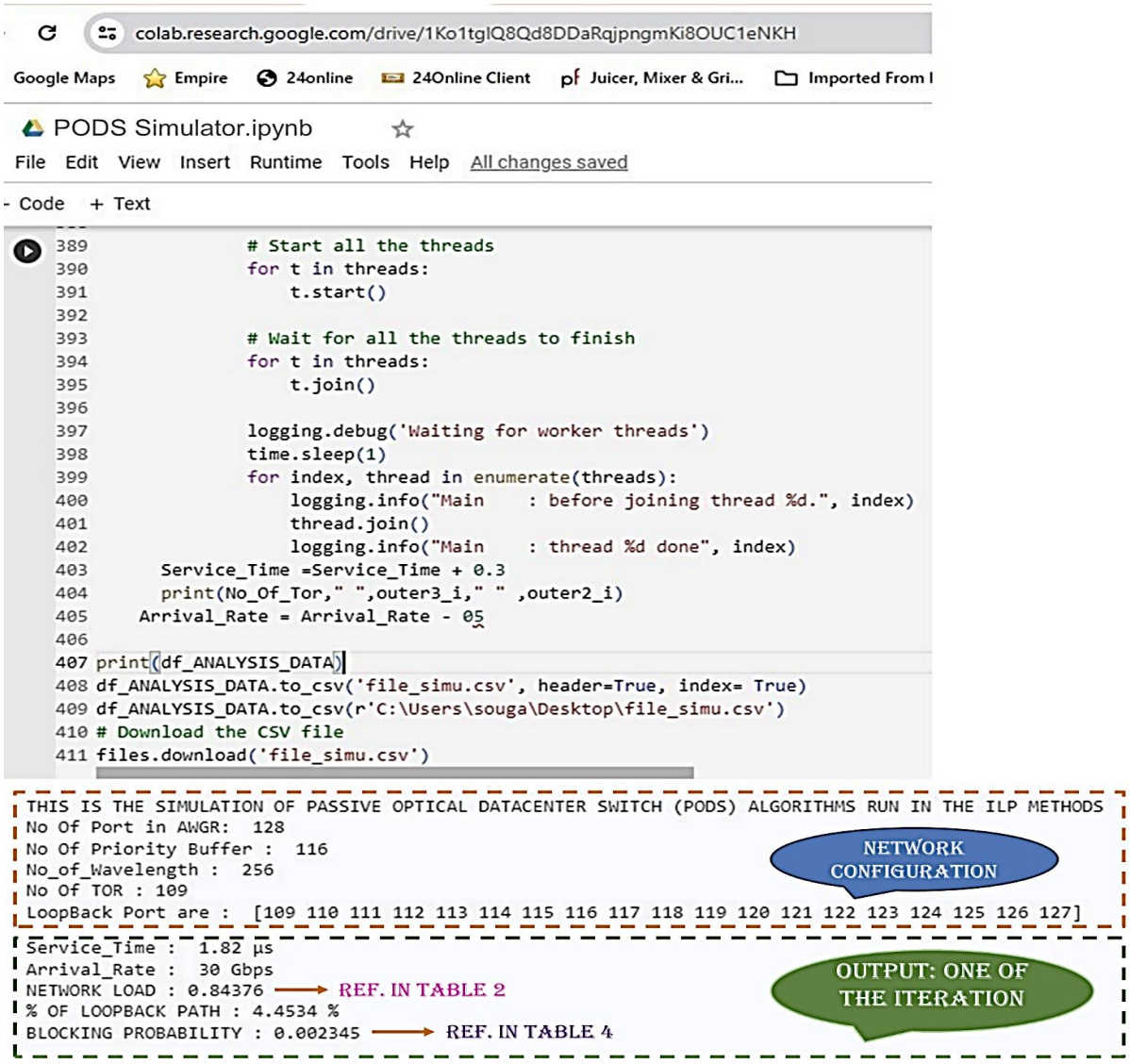


Fig. 3 Snap of PODS-based DCN Architecture executed in the Google Colab Environment

The Average Packet Latency Time (L_{avg}) is computed as the aggregate of three fundamental components: the transmission time, the average queuing delay, and the delay introduced by loopback path reconfiguration. Mathematically, this is represented as:

$$L_{avg} = T_{tx} + T_{queue} + T_{loopback}$$

Where:

- T_{tx}= Transmission time (time to send a packet over the optical link)
  $T_{tx} = \frac{S}{R}$ S: Packet size, R: Data rate
- T_{queue}= Average queuing delay (time the packet waits in the buffer before being scheduled)
  $T_{queue} = \frac{1}{\mu - \lambda}$ μ: Service rate of the buffer (packets/sec), λ: Arrival rate (packets/sec)
- $T_{loopback}$= Additional delay due to loopback path reconfiguration (only when no direct wavelength is available)
- $T_{loopback} = \delta . P_{loop}$ δ: Fixed delay penalty for loopback traversal (includes retuning + re-buffering), P_{loop}: Probability of a packet requiring loopback

By optimizing each component through architectural and scheduling improvements, we aim for the following target values:

- Transmission Time (T_{tx}) = 0.3 μs (achieved by using faster transceivers or reducing packet size),
- Queuing Delay (T_{queue}) = 2.0 μs (through increased buffer capacity and priority-aware scheduling),
- Loopback Delay ($T_{loopback}$) = 0.7 μs (by minimizing the loopback probability using efficient wavelength assignment).

Substituting these values into the latency equation:

$$L_{avg} = 0.3\mu s + 2.0\,\mu s + 0.7\,\mu s = 3.0\,\mu s$$

This optimized latency confirms that through strategic architectural enhancements, the proposed PODS-based DCN can meet the requirements of modern cloud-centric, real-time applications.

Table 3 shows the values of network load measured through simulation under different packet arrival rates for two different architectures under study. Fig. 4 illustrates the graphical representation of network load as a function of packet arrival rate. From the result, we see that for a 36 Gbps arrival rate, the network load reached 100% in the PODCA-S architecture. Still, in the case of PODS-based architecture, it becomes 95.4%, ensuring the enhanced scalability of the proposed architecture.

Table 3. Network Load with packet arrival rate for different architecture under study

Arrival Rate (Gbps)	**0.1**	**6**	**12**	**18**	**24**	**30**	**36**
N/W Load in PODS(%)	16.12	25.37	37.14	53.2	68.24	83.87	95.4
N/W Load in PODCA-S(%)	17.21	27.25	45.24	65.2	82.03	95.12	100

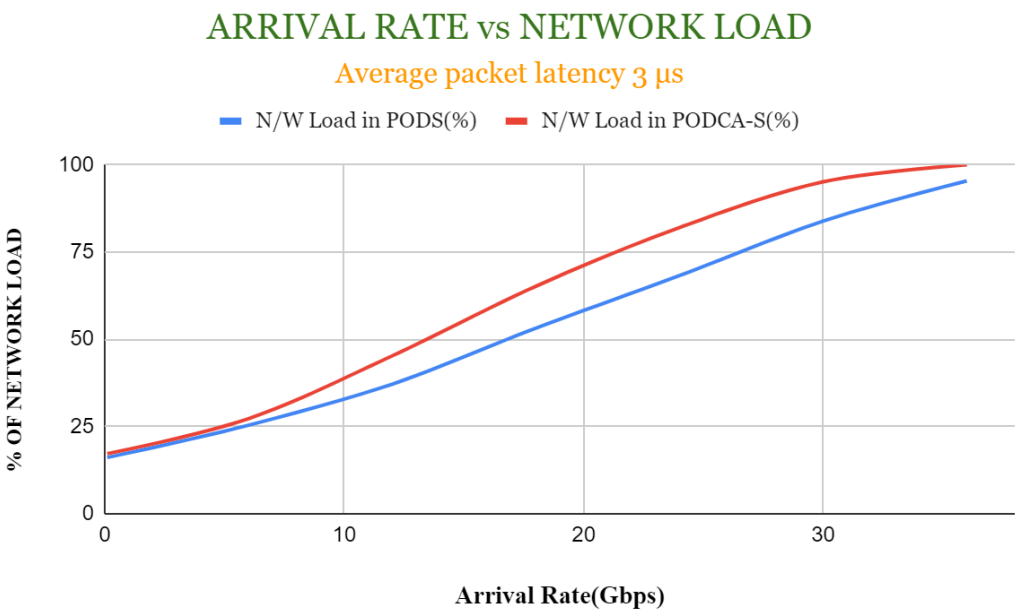


Fig. 4 Arrival Rate vs Network Load for different Architecture under study

Table 4 and Fig. 5 represent the latency of the network under study to arrival rate. At 36 Gbps arrival rate, the PODS architecture had a latency of 4.78 microseconds, whereas PODCA-S architecture exhibited a higher latency of 8.9 microseconds.

So, the latency of the network is reduced by 46.3% in the case of PODS-based DCN architecture, ensuring, its ability to process data more quickly and efficiently, making it more suitable for cloud-centric applications including low-latency communication, such as real-time data processing and high-frequency trading.

Table 4. Variation of latency with packet arrival rate for different protocol under study

Arrival Rate(Gbps)	**0.1**	**6**	**12**	**18**	**24**	**30**	**36**
PODS Latency (µs)	2.1	2.27	2.45	2.28	3.35	3.82	4.78
PODCA-S Latency (µs)	2.1	2.4	3.2	3.9	4.65	6.3	8.9

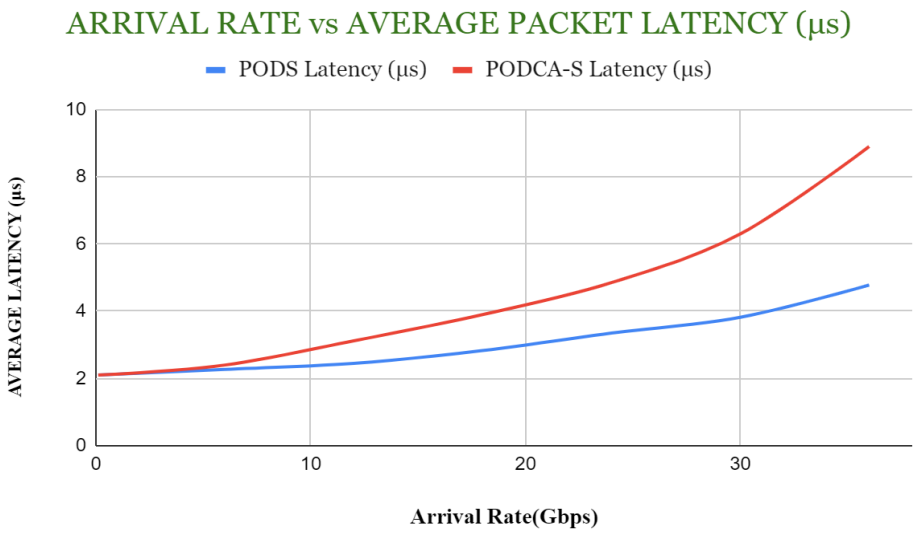


Fig. 5 Arrival Rate vs Average latency for the different protocol under study

From the above results, it is observed that for advanced switching and scalability PODS-based DCN architecture works better than that of PODCA architecture. So, for cloud-centric real-time applications, it will provide a more prominent solution. Further to find the performance in terms of the blocking probability of PODS-based DCN architecture, two cases are considered separately: using loopback and without using the loopback method. Table 5 and Fig. 6 show the blocking probability of the PODS-based architecture for different network loads. Observation shows 92% improvement in blocking in the case of PODS-based architecture using loopback.

Table 5. Blocking Probability with Network Load for PODS-based architecture

NETWORK LOAD	0.7	0.75	0.8	0.85	0.9	0.95	0.99
Blocking Probability in PODS with loopback	0	0	0.001	0.002	0.004	0.0069	0.009
Blocking Probability in PODS without loopback	0.0535	0.061	0.068	0.076	0.084	0.095	0.113

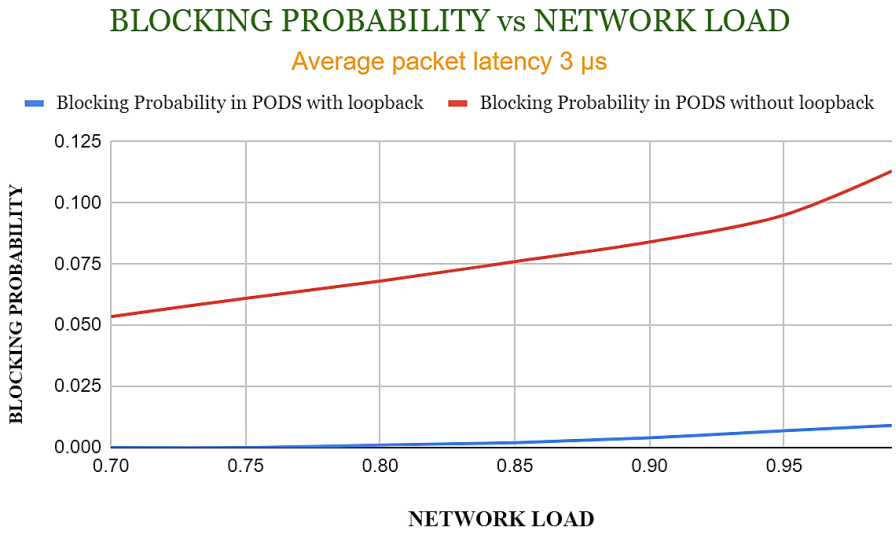


Fig. 6 Plot for Blocking Probability vs Network Load

The simulation confirms that the proposed PODS-based DCN architecture offers superior performance over PODCA-S, achieving up to 46.3% lower latency and reduced blocking probability by 92% with loopback. It also maintains a high network load capacity (95.4% at 36 Gbps) without saturation, demonstrating better scalability and efficiency. These results highlight its suitability for cloud-centric, low-latency applications.

5. Test Bench Implementation of the proposed model using Raspberry Pi

From the simulation result, we find that the PODS-based architecture using loopback provides an optimum solution in terms of network load, latency, and blocking probability for cloud cloud-centric real-time application environment. To validate the architecture in the hardware platform we developed a comprehensive test bench in our laboratory, consisting of seven servers. The AWGR and the control unit were implemented on Raspberry Pi module.

Hardware Configuration

Raspberry Pi Implementation

- **Model:** Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit.
- **Purpose:** Implementation of the 8x8 AWGR and Control Unit.
- **GPIO Ports:** Port assignments are as follows:
  - **Input Ports:** PIN 11, 13, 15, 29, 31, 35, and 37.
  - **Output Ports:** PIN 16, 18, 22, 32, 36, 38, and 40.
  - **Loopback Paths:** Two GPIO ports were dedicated to loopback paths for input and output.

Servers

- **Configuration:**
  - **Processor:** Intel Core i7-12700T, 1.40 GHz.
  - **RAM:** 8GB DDR4 3200MHz.
  - **Storage:** 512GB SSD.
- **Role:** These servers functioned as Top-of-Rack (ToR) units.

Connectivity and Pin Configuration

- **Server Connections:**
  - Seven GPIO ports of the Raspberry Pi were connected to the seven servers acting as TORs.
  - The connections ensured proper data flow management and control between the Raspberry Pi and the servers.

Memory Management

- **Memory Allocation:**
  - A total of 7GB of memory, was allocated within the Raspberry Pi for packet buffering.
  - Each input port was allocated 1GB of memory.
  - This 1GB was further subdivided into smaller buffers (B1, B2, B3, and B4) of variable sizes to manage incoming packets efficiently.

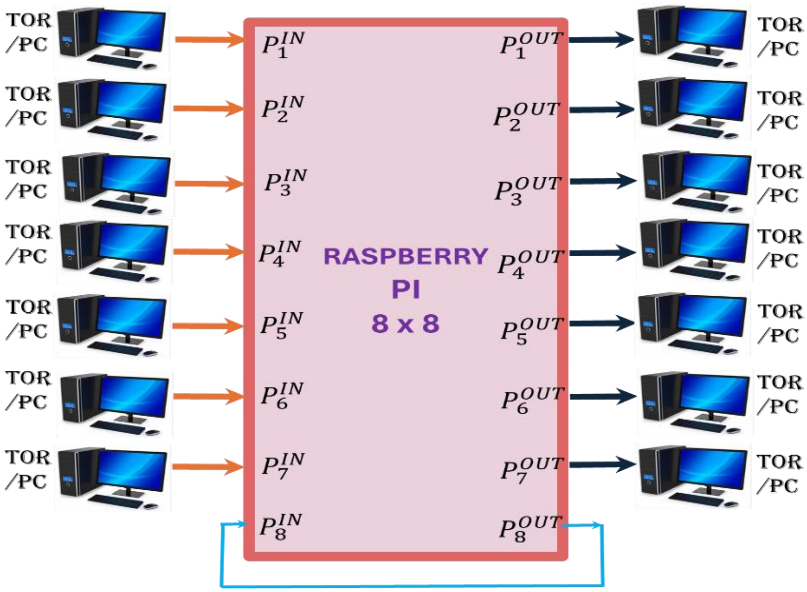
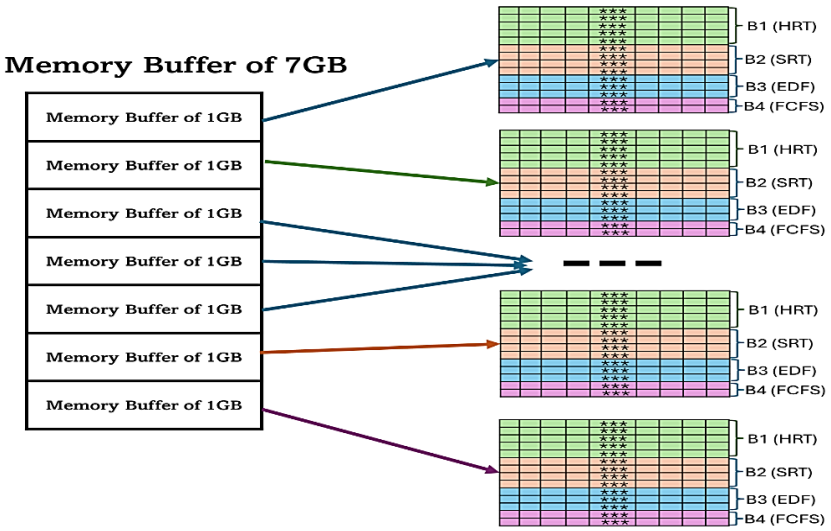


Fig. 7 Test Bench Setup Block Diagram

Fig. 7 illustrates the block representation of the test bench, showing the connection of Raspberry Pi and 7 servers (acting as TORs).

The memory allocation for the Raspberry Pi is shown in Fig. 8. Each input port is allocated 1GB of buffered memory, which is divided into four buffers: B1, B2, B3, and B4, each with different sizes. The Raspberry Pi module and its pin configuration are shown in Fig. 9. The test bench implementation of the proposed model is shown in Fig. 10.



.

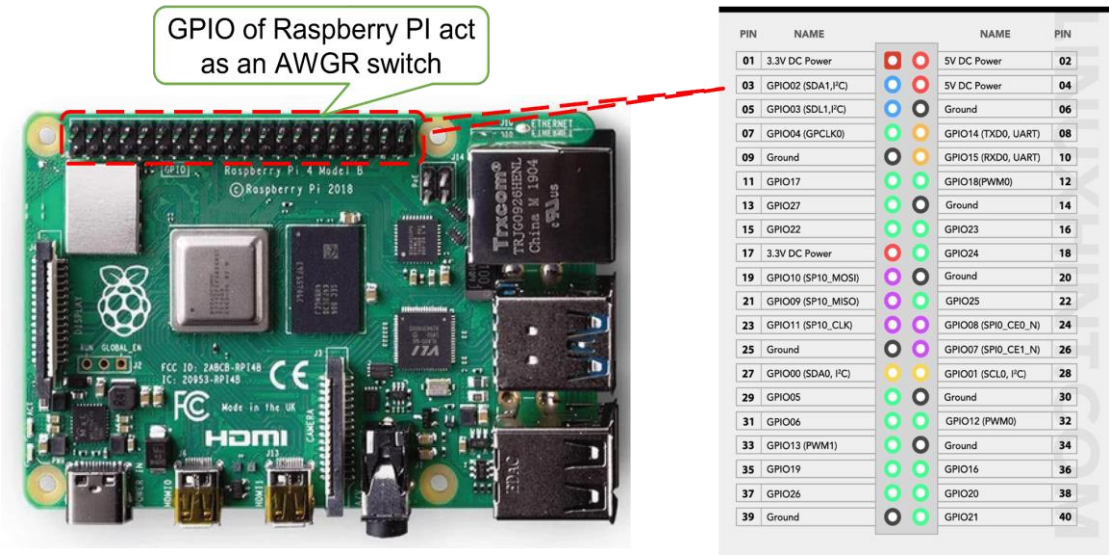


Fig. 8 Buffer allocation scheme in the proposed test bench implementation

Fig. 9 Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit Raspberry-Pi module and its pin configurations

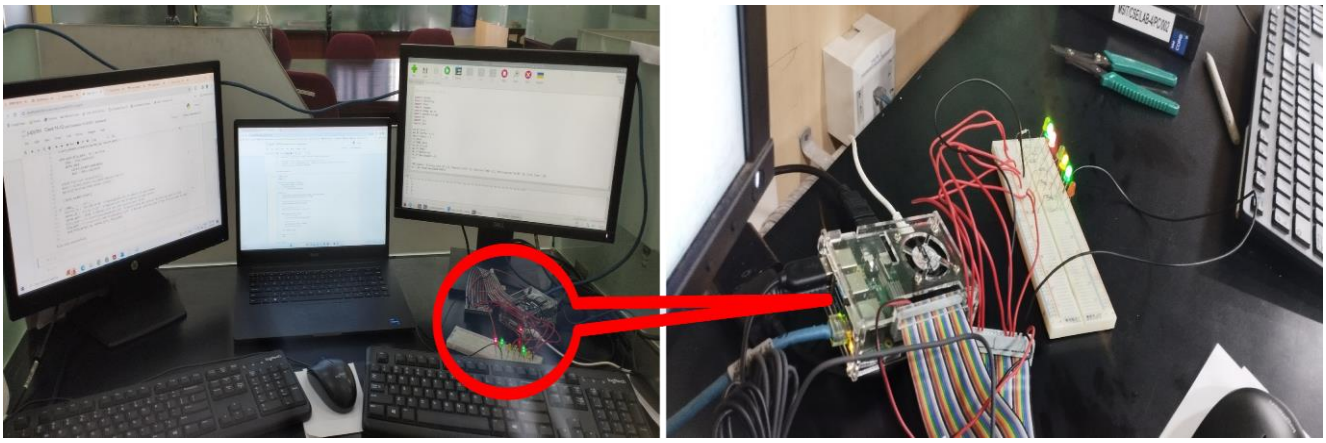


Fig. 10 Test bench implementation model of the proposed architecture

6. Experimental Results

In the experimental setup, 16 wavelengths are utilized for packet transmission from source to destination. Communication between the source and destination is facilitated through a socket connection established via the Raspberry Pi model. Each Top of Rack (ToR) switch is capable of using all 16 wavelengths for packet transmission, resulting in a total of 112 possible communication links (7 ToRs × 16 wavelengths).

Table 6. Buffer Allocation based on service type in terms of network Load per ToR

NO OF BUFFER	(B1) 35%	(B2) 30%	(B3) 20%	(B4) 15%
8	3	2	2	1
9	3	3	2	1
10	3	3	2	1
11	4	3	2	2
12	4	3	3	2
13	4	4	3	2
14	5	4	3	2
15	5	4	3	2
16	5	5	3	2

The network load is categorized into four distinct service classes, with traffic distribution as follows: 35% allocated to High-Real-Time (HRT) traffic corresponding to Buffer 1 (B1), 30% to Soft-Real-Time (SRT) traffic corresponding to Buffer 2 (B2), 20% to Earliest Deadline First (EDF) traffic corresponding to Buffer 3 (B3), and 15% to First-Come-First-Served (FCFS) traffic corresponding to Buffer 4 (B4). The number of buffers per ToR varies between 8 and 16. The specific allocation of buffers to each service class is detailed in Table 6, with the majority allocated to HRT traffic to minimize congestion and packet loss.

Performance Analysis

The network load was experimentally varied by establishing between 56 (50% load) and 112 (100% load) communication links, and the blocking probability was calculated with and without buffer reuse. Data was collected at 100 distinct timestamps for each measurement, and the results were averaged.

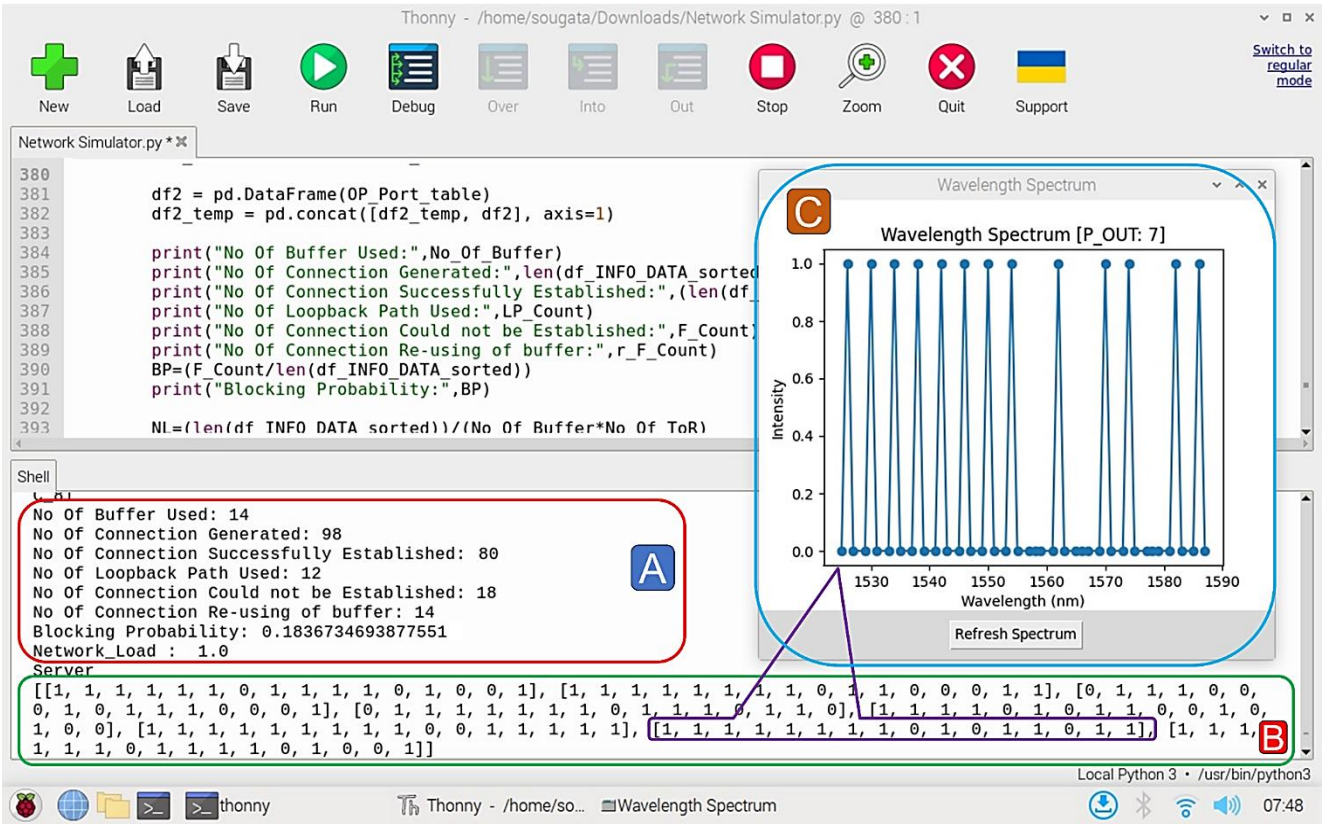


Fig. 11 Snapshot of execution of program and wavelength assignment for ToR1

Fig. 11 provides a snapshot of the program execution and wavelength assignment for ToR1. At a specific time instant depicted in Fig. 11, Block A shows that 98 connections were attempted, with 80 successfully established and 18 blocked due to insufficient buffer availability and wavelength allocation conflicts, resulting in a blocking probability of 0.18367. Block B illustrates the wavelength allocation, where the sequence [1,1,1,1,1,1,1,1,0,1,0,1,1,0,1,1] represents the assignment in port 7. Wavelengths w_1 to w_8, w_{10}, w_{12}, w_{13}, w_{15}, and w_{16} are set to 1, indicating successful connection establishment between the source and destination, while wavelengths w_9, w_{11}, and w_{14} remain unused. Block C of Fig. 11 displays the specific wavelengths utilized for successful communication.

Table 7 presents the relationship between blocking probability and network load. It was observed that there is minimal variation in blocking probability when the network load increases from 0.4 (40%) to 0.5 (50%). However, a significant increase in blocking probability is noted as the network load rises from 0.8 (80%) to 0.99 (99%). Fig. 12 provides a graphical representation of this relationship. The results indicate that implementing buffer reuse improves blocking probability by an additional 50%, which in turn reduces network congestion, while also enhancing scalability.

Table 7. Blocking probability with respect to network load with reused buffer

NETWORK LOAD	0.4	0.5	0.6	0.7	0.8	0.9	0.99
Blocking Probability in PODS without re-used buffer	0	0.041	0.078	0.122	0.193	0.282	0.374
Blocking Probability in PODS with re-used of buffer	0	0.03	0.06	0.091	0.1235	0.158	0.186

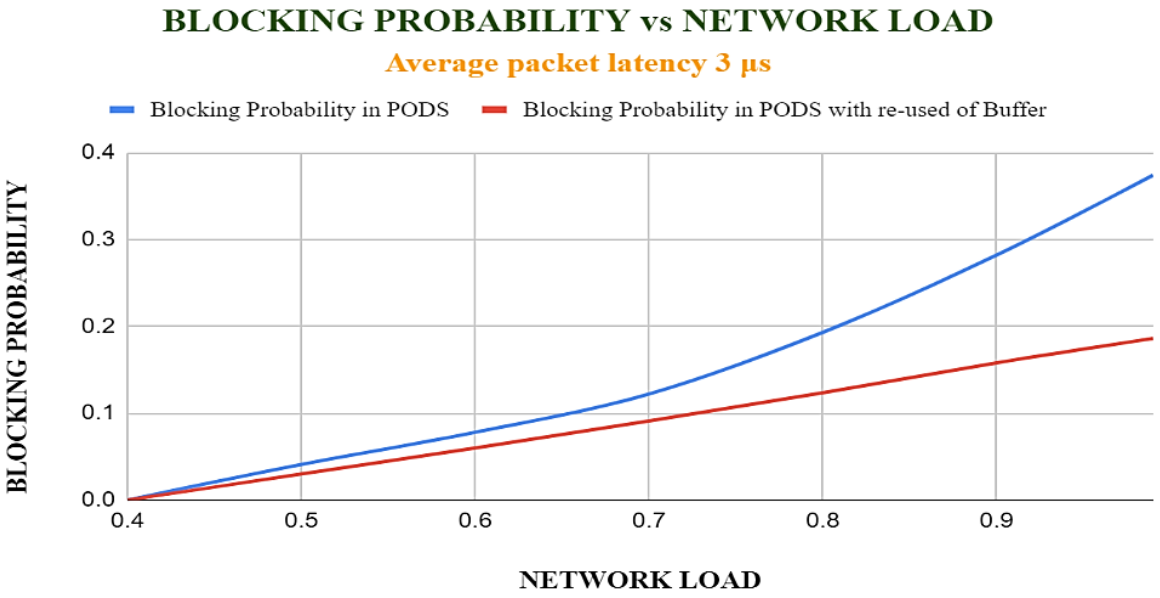


Fig. 12 Blocking probability with respect to network load

To further validate the proposed model, blocking probability was assessed under varying network loads by altering the number of buffers per ToR, as shown in Table 8 and Fig. 13. The results indicate that blocking probability decreases as the number of buffers increases. Additionally, Table 8 reveals that the optimal blocking probability is achieved with 14 buffers per ToR, beyond which the reduction in blocking probability plateaus, likely due to wavelength unavailability.

Table 8. Blocking probability with respect to network load with different buffer size

NETWORK LOAD		50%	56%	63%	69%	75%	81%	88%	94%	100%
NO OF BUFFER	8	0.170	0.206	0.243	0.286	0.333	0.385	0.418	0.452	0.482
	9	0.107	0.138	0.171	0.221	0.262	0.316	0.352	0.390	0.420
	10	0.068	0.087	0.114	0.162	0.202	0.253	0.294	0.326	0.357
	11	0.043	0.060	0.086	0.125	0.161	0.201	0.244	0.271	0.295
	12	0.033	0.048	0.071	0.104	0.131	0.165	0.200	0.224	0.246

	13	0.030	0.048	0.071	0.091	0.119	0.143	0.171	0.189	0.207
	14	0.030	0.048	0.071	0.091	0.113	0.132	0.153	0.171	0.188
	15	0.030	0.048	0.071	0.091	0.113	0.126	0.143	0.162	0.179
	16	0.030	0.048	0.071	0.091	0.113	0.126	0.143	0.162	0.179

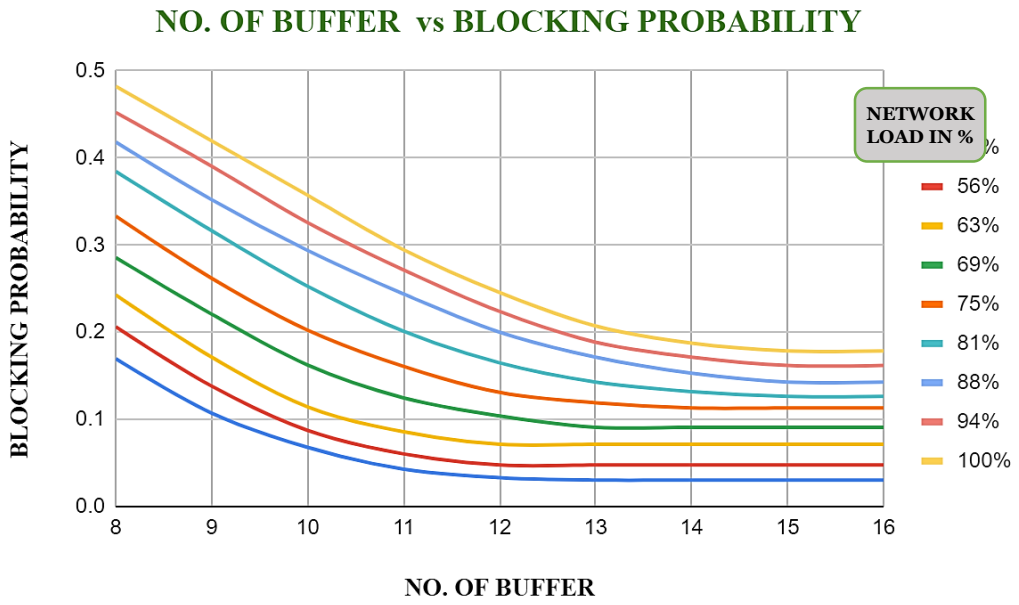


Fig. 13 Blocking probability with buffer size for different network loads

While Tables 7 and 8 present the overall blocking probability of the entire architecture, Table 9 provides a detailed breakdown of the number of connections dropped within each service class.

Table 9. Number of blocked connections in different services

SERVICE TYPE	B1 (HRT)	B2 (SRT)	B4 (EDF)	B4 (FCFS)
Number of block connections with re-used buffer	**0**	**3**	**6**	**9**

The results demonstrate that by employing path reconfiguration and buffer reuse, the blocking probability for High-Real-Time (HRT) traffic is reduced to zero, while it increases progressively for lower-priority traffic, reaching a maximum of 9.1% for FCFS traffic. This finding further confirms that the proposed architecture is an effective solution for managing real-time mixed traffic in cloud-centric applications.

7. Conclusion

This paper has presented an optical circuit-switched framework aimed at enhancing service provisioning in data center networks through improved scalability, dynamic QoS adaptation, and reduced

blocking probability. The proposed architecture effectively simplifies network complexity while supporting cloud-centric, high-speed, and real-time data transmission across heterogeneous traffic environments. By integrating reconfigurable optical switching with intelligent control mechanisms, the design demonstrates a promising pathway toward efficient, scalable, and latency-optimized DCNs, capable of meeting the stringent performance demands of modern cloud and data-intensive applications.

References

1. IEA (2024), Global data center electricity use to double by 2026, IEA, Paris https://www.datacenterdynamics.com/en/news/global-data-center-electricity-use-to-double-by-2026-report/
2. Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. 2010. C-Through: part-time optics in data centers. SIGCOMM Comput. Commun. Rev. 40, 4 (October 2010), 327–338.
3. Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2010. Helios: a hybrid electrical/optical switch architecture for modular data centers. SIGCOMM Comput. Commun. Rev. 40, 4 (October 2010), 339–350.
4. Balanici, M.; Pachnicke, S. Hybrid Electro-Optical Intra-Data Center Networks Tailored for Different Traffic Classes. J. Opt. Commun. Netw. 2018, 10, 889–901, https://doi.org/10.1364/jocn.10.000889.
5. A. K. Singh, R. Prakash, and S. Kumar, "Hybrid electro-optical DCN architecture for real-time cloud applications," IEEE Trans. Cloud Comput., vol. 11, no. 2, pp. 567–580, Apr. 2023, doi: 10.1109/TCC.2023.123456
6. J. Park, H. Lee, and S. Han, "Dynamic priority scheduling in hybrid optical DCNs," IEEE Trans. Parallel Distrib. Syst., vol. 34, no. 8, pp. 2345–2358, Aug. 2023, doi: 10.1109/TPDS.2023.123456.
7. Y. Wang, J. Zhang, and K. Bergman, "AWGR-based optical DCN with dynamic buffer management," IEEE Trans. Netw. Service Manag., vol. 20, no. 1, pp. 120–135, Feb. 2023, doi: 10.1109/TNSM.2023.123456.
8. X. Ye, Y. Yin, S. B. Yoo, P. Mejia, R. Proietti, and V. Akella, “DOS: a scalable optical switch for datacenters,” in 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ACM, 2010, p. 24.
9. K. Xi, Y.-H. Kao, M. Yang, and H. Chao, “A petabit bufferless optical switch for data center networks,” in Optical Interconnects for Future Data Center Networks, Springer, 2013, pp. 135–154.
10. Furukawa, H. Petabit-class Optical Networks and Switching Technologies. In OSA Advanced Photonics Congress 2021, https://doi.org/10.1364/NETWORKS.2021.NeW1C. 1.
11. Gupta, A., Roy, S., & Kumar, N. (2023). Energy-efficient TWCs in optical DCNs. IEEE Transactions on Green Communications and Networking, 5(1), 34–48. https://doi.org/10.1109/TGCN.2023.123456
12. Xu, Maotong & Liu, Chong & Subramaniam, S.. (2018). PODCA: A passive optical data center network architecture. Journal of Optical Communications and Networking. 10. 409. 10.1364/JOCN.10.000409.

13. Y. Yin, R. Proietti, X. Ye, C. J. Nitta, V. Akella and S. J. B. Yoo, "LIONS: An AWGR-Based Low-Latency Optical Switch for High-Performance Computing and Data Centers," in IEEE Journal of Selected Topics in Quantum Electronics, vol. 19, no. 2, pp. 3600409-3600409, March-April 2013, Art no. 3600409, doi: 10.1109/JSTQE.2012.2209174.
14. X. Chen, Y. Zhao, and Z. Liu, "Dynamic QoS-aware wavelength assignment in optical data center networks," IEEE/OSA J. Lightw. Technol., vol. 41, no. 5, pp. 1324–1336, Mar. 2023, doi: 10.1109/JLT.2023.123456.
15. Roy, A., Zeng, H., & Porter, G. (2023). Quartz: A QoS-aware optical DCN architecture. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 7(1), 1-28. https://doi.org/10.1145/3582524
16. Al-Fares, M., Loukissas, A., & Vahdat, A. (2022). Scalable optical interconnects for cloud data centers. ACM SIGCOMM Computer Communication Review, 52(3), 45-58. https://doi.org/10.1145/3544912.3544918
17. E. Wong, C. Lim, and Y. C. Chung, "Dynamic load balancing in SDN-enabled optical DCNs," IEEE Syst. J., vol. 17, no. 1, pp. 456–469, Jan. 2023, doi: 10.1109/JSYST.2023.123456.
18. Han, F.; Wang, M.; Cui, Y.; Li, Q.; Liang, R.; Liu, Y.; Jiang, Y. Future Data Center Networking: From L ow Latency to Deterministic Latency. IEEE Netw. 2022, 36, 52–58, https://doi.org/10.1109/mnet.102.200 0622.
19. Testa, F.; Pavesi, L. Optical Switching in Next Generation Data Centers. Springer, Cham: Switzerland, 2017.
20. Zhang, M., Wang, L., & Li, H. (2022). Software-defined optical data center networks with dynamic QoS adaptation. IEEE Transactions on Cloud Computing, 10(4), 2456-2470. https://doi.org/10.1109/TCC.2021.3098765
21. Xue, X.; Wang, F.; Chen, S.; Yan, F.; Pan, B.; Prifti, K.; Guo, X.; Zhang, S.; Xie, C.; Calabretta, N. Experimental Assessments of SDN-Enabled Optical Polling Flow Control for Contention Resolution in Optical DCNs. J. Light. Technol. 2020, 39, 2652–2660, https://doi.org/10.1109/jlt.2020.3042820
22. R. Bera, S. Mishra, and D. Kilper, "Loopback-based path reconfiguration in AWGR DCNs," IEEE/OSA J. Opt. Commun. Netw., vol. 15, no. 4, pp. B78–B90, Apr. 2023, doi: 10.1364/JOCN.2023.123456.
23. D. Kim, S. Park, and H. Kim, "Blocking probability reduction in AWGR-based DCNs," IEEE Photon. Technol. Lett., vol. 35, no. 8, pp. 423–426, Apr. 2023, doi: 10.1109/LPT.2023.123456.
24. Sougata Bera, Chandi Pani, “Re-routable and Low-latency All Optical Switching Algorithm for Next Generation DCN” - IJFMR Volume 5, Issue 6, November-December 2023. DOI 10.36948/ijfmr.2023.v05i06.9965