INT-SDN: Evaluation of various P4 parameters using optical telemetry having reconfigurable data plane on 40 Gbps line rate

Harminder Singh[§], Changcheng Huang[§], Mathieu Sicard-Gagne[†], Gauravdeep Shami[†], Marc Lyonnais[†], Dmitri Fedorov[†] and Rodney Wilson[†]
[§]Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada
[†]Ciena, CTO Team, Ottawa, Canada
Email: harmindersingh@cmail.carleton.ca, huang@sce.carleton.ca, {msicard, gshami, mlyonnai, dfedorov, rwilson}@ciena.com

***Abstract*— Ethernet packet communication is the old established protocol of data transfer in networks. Ethernet devices transmit the data from a source to a receiver while the receiver has no statistics regarding the transmission channel. In this era of global transmission where the upcoming communication technology will be equipped with self optimized transmission under which there will be minimum errors. The need to gather information related to the transmission path arises. The primary objective of this research paper is to evaluate the different P4 parameters on optical media. P4 enabled NIC's, corresponding FPGA Images and traditional Ethernet switches were used for the results. INT-SDN P4 packets were transmitted and the comprehensive test traces were captured and examined. DPDK was used which provides the foundation for P4 packet forwarding. Data, as well as the control plane were fully configurable in the hardware setup which resulted in speedier data transmission and also enhanced the packet efficiency at higher data rates. Metrics such as latency, throughput, data rates and efficiency were accumulated, compared and analyzed on P4 based optical network having a 40 Gbps data transmission rate. Moreover, the inside story of the optical channel was gathered which will provide further insights related to the P4 transmission medium.**

***Keywords*— Programming Protocol Independent Packet Processors (P4), In-Band Network (INT), Software Defined Networking (SDN), Network Interface Card (NIC), Data Plane Development Kit (DPDK), Field Programmable Gate Array (FPGA), Application Specific Integrated Circuit (ASIC), Transmission Control Protocol (TCP).**

I. INTRODUCTION

As specified in Gartner's report [1], the number of communicating devices will reach 20.4 billion and that of the IoT implements will touch 30.7 billion by 2020. In accordance with this massive expansion in the number of hardware implementations, the need to raise the bandwidth will arise with respect to time. Therefore, a traditional TCP/IP packet should be modified to accommodate the large amount of information that needs to be successfully transmitted and efficiently received. Moreover, the packet size has to be raised and an addition of more headers will be needed with time. To process the larger volume of packets, particular packet handling mechanisms will also be required to maintain the overall efficiency of the optical network. INT-SDN [2] is the possible solution to this future network complication as SDN is bound to deliver the data in a more economical way while having easier network implementations. Additionally, SDN is also capable to provide centralized provisioning of the optical networks.

Data Plane, Control Plane and the Management Plane are the three vital components of any networking system. While data plane carries the user information, the control plane consists of routing information as well as of signaling components and the administrative instructions are in the management plane. SDN creates a divergence in the data and control plane providing more flexibility to the network management by transferring the control plane from hardware device to the software domain.

The concept of OpenFlow was introduced to remove the communication barrier [3] between hardware devices of different manufacturers and to create a single platform of networking irrespective of any vendor-specific physical hindrance. OpenFlow consists of a flow tables, some controllers and a common protocol. The protocol was designed to match the packet headers across multiple layers and also to get the TCP/UDP statistics of the forwarded packets. However, with respect to time and globalization, the need to elevate the packet size arises and therefore, the number of packet headers also expanded.

P4 is a protocol independent programming language [4] that enables the administrator to have full control over the data plane. P4 provides modification and enhancement to the traditional Ethernet topology which uses the TCP/IP concepts for the data forwarding mechanism. The communication through the old established Ethernet terminology only provides the transfer of information from source to sink, and there is no collection of metadata from the communication channel. Metadata refers to the ingress timestamp, egress timestamp, queue occupancy of the packets along with the switch ID's of all the intermediate devices through which packets have gone through. This metadata adds more details to the packets and will assist in tracing the path of the packets. Moreover, the timestamp of the packet will provide the time at which the P4 packet enter and leave a particular device, which will aid in calculating the latency of the overall packets.

In P4 communication the metadata is called as INT collection which corresponds to the calculation of queue occupancy and hop latency of the packets. The metadata is inserted by the devices very much similar to the push mechanism on the stacks, in which the most recently added metadata will appear at the top of the stack and will be clearly visible at the output side. This metadata is the driving force of the system which could be used to feed the future Artificial Intelligence (AI) as well as to the Machine Learning (ML) algorithms for diversified analysis.

978-1-7281-2794-1/19/$31.00 ©2019 IEEE

II. ADVANTAGES OF P4

Unlike TCP/IP packets which have a fixed number of headers, P4 packets can have an adjustable number of headers depending on the required application. These additional headers in P4 are cited as INT headers which collects the in-band network telemetry and delivers this stored information to the receiver. In short, P4 focuses on the Information-Centric Networking [5] which revolves around gathering the intended information related to the optical channel as well as to the overall system.

Implementing P4 over the existing Ethernet devices without hindering the present network setup is a major concern while running P4. However, Service-Centric Networking [6] is the unique property of P4 which empowers P4 to run over the traditional TCP/IP protocol. This service-centric attribute validates the communication without altering any pre-established settings with TCP/IP header arrangement.

III. ARCHITECTURAL CONFIGURATIONS

- **DPDK**

Multithreading processing mechanism works with minimum errors up to line rate of 10 Gbps. Therefore, specialized packet processing techniques are required to cover up the data rate gap while transferring information. There are multiple packet processing software available such as DPDK, NetMap, PF_RING_DNA, etc. As some of the exclusive hardware devices which can process higher data rate are not economical to use, hence, open-source software applications are being implemented nowadays.

The DPDK performs three main tasks: Storing the packets by allocating memory space, managing the generated P4 packet queue and buffer management of the packets. Where queue management is performed by *librte_ring* DPDK library by providing a ring structure called as *rte_ring*.

When there is a massive burst of packets, the need to fasten the processing rate arises within the communicating software. DPDK is a software application consisting of a large number of libraries that accelerate the packet processing rate [7] by eliminating the use of Linux Kernel stack (Figure. 1). In this way, the application implemented in user space communicates directly with the physical networking device. DPDK behaves as a user place directory while using the *pthread* library.

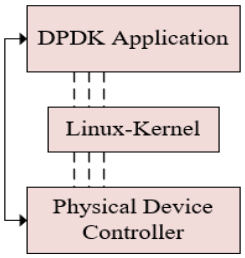

Figure 1. Kinux by-passing by DPDK.

EAL (Environment Application Layer) in DPDK holds the relationship between hardware and DPDK memory space. Hugepages is the corresponding memory space that is being used by the DPDK and Linux kernel for the resource allocations. The EAL is also responsible for avoiding the panic condition when a single application is required to use multiple physical as well as software resources at once. The basic function of EAL is to initialize resources such as PCI Devices, physical consoles, timers, etc. Execution instances are created by EAL which provides smoother access to the physical cores. The initialization is performed after building and making the application.

- **P4 Switch**

In our research work, Inventec switch is implemented and used. The switch is P4 enabled L2/L3 programmable white box which performs packet forwarding for the optical network. It has a bi-directional throughput of 6.4 Terabits, latency of 675 nanoseconds (or 0.675 microseconds), memory of 8 GB and the packet buffer capacity of almost 22 MB. The switch has 64 physical ports that can perform on 100 GE high-end data rates. The main advantage of the Inventec switch lies in the dynamic buffer management system. This system prevents the collision of heavy TCP/IP traffic bursts [8] between the transmitting and the receiving nodes of the network implementation. On top of that, in terms of network security concerns the switch provides Accounting and Authorization facility along with ACL Permit/Deny and IPv6 ACL realizations and enforcements.

- **8700 Ethernet Switch**

An Ethernet switch acts as a centralized device that gets data from multiple ports and provides them with full available bandwidth. Ciena 8700 Multi-Port 100 Gbps Ethernet Switch is used while creating the P4 Testbed on the optical telemetry. This switch aggregates the incoming data traffic and diverges that into corresponding physical ports for the next destinations. Multi-Protocol Label Switching (MPLS), Zero-Touch Provisioning (ZTP) and Segment Routing are some of the switch attributes which provide advanced functionality and high-end data executions.

- **P4 NIC Card**

A P4 NIC is capable of generating P4 packets, which consists of modified Ethernet headers along with some additional header fields. Two Alpha-Data 40 Gbps P4 NIC's are implemented in our Testbed setup which creates and receives the P4 packets over the optical network. C/C++ header files and libraries are used which provide API's for controlling the reconfigurable computing planes [9] for the practical applications.

A specific FPGA image is flashed over the NIC's using Xilinx Vivado software in order to make the NIC's P4 operational. That FPGA image consists of the bitstreams that are having the Match-Action Tables (MAT). The image could be modified in accordance with the desired output while considering the type of application implemented. These tables act as the guiding principles and are required to perform specific tasks on the received P4 packets. These tasks may include shortening the header or adding metadata to the particular header fields. In traditional NIC's there is no provision of header modifications, however, P4 communication provides multiple options to generate a flexible data packet.

IV. P4 TESTBED SETUP

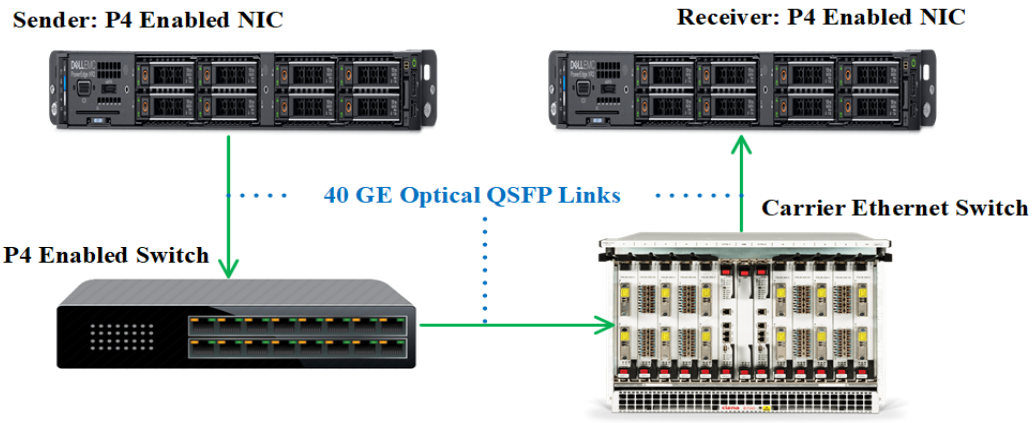

Figure 2. P4 Testbed Architecture.

(i.) P4 Packet Trace

The P4 packet is generated at the sender side using the P4 enabled NIC. The NIC is capable of producing the packets of the desired length while having specific packet headers. The process of initializing the packets begins with defining the desired number of headers in the sample application. Where sample application is used to create and store the P4 packet formations which could be altered at any stage in the communication process. The packet then travels through the 40 GE optical QSFP link and reaches the P4 enabled Inventec switch. Then the packet is forwarded to the 8700 carrier Ethernet switch. This switch just passes the packets to the P4 enabled NIC, which serves as the receiver side.

While traveling through the Inventec switch, the metadata is inserted to the P4 packet headers. This metadata is referred to as the ingress as well as the egress timestamp and the switch ID. The packet has been provided with enough storage to smoothly collect and store the metadata in the specific INT headers. Then this collected metadata is delivered to the receiver NIC which then displays the desired information included in the INT headers.

For the exact realization and proper execution of the INT headers two vital procedures are followed at the NIC's: Parsing and deparsing. These two mechanisms are implemented at both the sender and receiver nodes of the Testbed architecture in order to maintain a flow in system performance.

(ii.) Parsing

In this P4 Testbed (Figure 2), uncomplicated FPGA images are used to avoid any unnecessary handling of matching algorithms. There are basically two types of rules for the matching tables: ternary and exact. Hash based lookup tables are also being implemented depending on the type of header extractions at the output side.

Parsing is the process in which the INT headers in the P4 packets are matched with the tables stored in the FPGA load and the complete procedure is called as the match-action tables. Under the parsing stage, the specified INT headers are identified by looking into the FPGA load. A proper sequential state and transition method is followed and can be cited under Finite State Machines (FSM's) for further clarifications as shown in Figure 3. The FPGA image contains the bitstream and the same image is implemented on the sender as well as on the receiver side for proper synchronization of the packet header extraction.

(iii.) Deparsing

After identifying the particular INT headers for further processing, deparsing is the next stage where the actual action is carried out. As compared to parsing, the deparsing mechanism is more complex to implement in the actual scenarios. The deparser is always in the activity mode being ready to perform and complete the end tasks at the P4 packet.

The deparser can add or delete the INT headers and these functions are based on the information provided by the parser side. In short, the deparser is capable and responsible for performing three functions on the packet:

- **Packet Integration**

Under this procedure, the deparser can merge two P4 packets as ordered by the parser. The parser provides the information onto which two headers are to be combined in order to combine all the data into a single INT header to be delivered to end node.

- **Packet Extending**

In this P4 Testbed, packet integration is not implemented considering the type of complexity and amount of maintenance associated. However, extending and compression are realized on practical grounds. When metadata is inserted into the packet header by the intermediate nodes and the size of the overall packet grows automatically, then this mechanism is called as packet extending. With each subsequent interaction of the P4 packet with hardware devices, the INT header expands in terms of size.

- **Packet Compression**

Packet compression refers to the deletion of some data from the INT header in order to shorten the size of the packet header. In some scenarios, where the need to remove some unnecessary information arises then this packet compression is implemented. In this P4 Testbed (Figure 2), the packet compression is implemented but not widely used as the focus is to gather the metadata while employing the packet extender functionality.

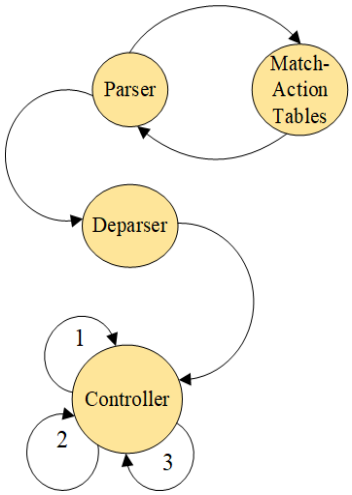

Figure 3. Packet abstraction using Finite State Machine (FSM). Where 1 corresponds to packet buffer, 2 belongs to packet drop/ delete and 3 refers to the packet out.

While considering the packet processing conditions and the implementation of control flows the realization is much complex on FPGA's as compared to that of the ASIC's. The reason lies behind the excess amount of data that the P4 packet carries in form of the metadata collected from each corresponding networking nodes. Tree structured algorithms and branch conditions are used in FPGA's while executing the packet contained in the sample P4 application.

V. P4 PACKET STRUCTURE

The traditional Ethernet packets have 7 header fields named as: Preamble, Start Frame Delimiter (SFD), Destination Address (DA), Source Address (SA), Length, Payload and Cyclic Redundancy Check (CRC). Whereas the P4 packet also has 7 headers but with the addition of specific INT headers:

(i.) IDMA Metadata (Header Size: 8 Bytes)

This header doesn't carry any metadata rather this indicates the starting of the P4 packet. This header is included to differentiate the TCP/IP packet from the P4 packet.

(ii.) Ethernet (Header Size: 14 Bytes)

This header is created by combining three Ethernet packet headers: Destination MAC Address, Source MAC Address and the Length header. The challenge in P4 networking is to maintain the old established Ethernet protocols and make them compatible with the P4 methodology, therefore, the three Ethernet headers have been carried out while implementing P4 networking in actual scenarios.

(iii.) IPV4 (Header Size: 20 Bytes)

Three types of information is contained in this header: Sender IP Address, Receiver IP Address and the length of the IPV4 header.

(iv.) TCP (Header Size: 20 Bytes)

This header holds the transmission protocol standard that the packet needs to follow while travelling in the network. Two types of flags are used for the proper packet transmission and they are SYN and ACK flags. Where the SYN is used to maintain the synchronization between the two connected nodes and the ACK flag is used to acknowledge the sender that the packet containing information has been successfully received.

(v.) INT Shim (Header Size: 4 Bytes)

The critical difference between the Ethernet and the P4 packet starts from this header onwards. INT (IN-Band Network Telemetry) Shim header is introduced to the Ethernet packet to modify it from Ethernet to P4 communication. The INT shim header is adjustable in terms of header size because this is the header which gathers all the metadata from the communicating intermediated devices. This header contains the ingress as well as egress timestamps of the P4 packet, switch ID of the device, hop latency and queue occupancy at the NIC side. Every P4 enabled device adds its own metadata and the size of INT shim header goes on expanding with respect to time.

(vi.) INT Header (Header Size: 8 Bytes)

This INT header was introduced in order to have proper details of the P4 packet transmission path and to trace the hops that the packet has gone through. This header gives directions to the packet to reach the end node successfully without being lost in the channel.

(vii.) Payload (Header Size: 231 Bytes)

The actual information to be transmitted by the sender node is contained in the payload header. With respect to the amount of data carried out by this header, the size of the payload is given with maximum space i.e. 231 bytes.

- **Packet Forwarding.**

The implementation and realization of the P4 language is target and protocol independent [10] which means the administrator doesn't need to clarify the specifications of each and every intermediate device to the P4 packet. Once the destination address is known to the packet then with the assistance of INT header the P4 packet can successfully reach the end node without any hassle, provided all the devices that the packet is going to interact with are P4 enabled. The primary objective of the P4 transmission is to gather the metadata and to transfer the information from one node to another without any transmission nuisance. All other headers except the payload are dumped once the packet reaches the destination address and then the data is delivered successfully to the receiver end. The end node can view and analyze the received statistics and can trace the packet pathway through the hop ID's stored in the metadata provided by the INT shim header. This helps in tracing the path for the P4 data packet.

VI. PSEUDO ALGORITHM

Pseudo Algorithm 1: Parser-Deparser Functionality

```
1  StartTime → Packet_Ingress.nanoseconds()
2  EndTime → StartTime + Packet_Egress.nanoseconds()
3  Set PacketSize (256 Bytes) for TestSet_TX
4  LineRate (Potentially Available) → 100 GE
5  DataRate (Implemented) → 40 GE
6  Login into the TestSet using valid credentials
7  if PacketBurst() is available then
8  |   Start Transmitting Packets
9  |   PacketBurst() at P4_Enabled_Inventec_Switch_RX
10 |   Buffer Packet at Parser
11 |   Match Packet_Headers with FPGA_Tables
12 |   if MAT = Packet_Extender then
13 |   |   Forward Packet to Deparser
14 |   |   Add Metadata
15 |   end
16 else
17 |   if MAT = Packet_Compressor then
18 |   |   Forward Packet to Deparser
19 |   |   Remove particular Headers
20 |   end
21 end
22 foreach P4 packet generated at NIC do
23 |   Forward PacketBurst() to P4 enabled switch_RX
24 |   Forward PacketBurst() to 8700 Ethernet switch_RX
25 |   PacketBurst() at P4_Enabled_Sink_NIC_RX
26 |   Call New PacketBurst()
27 end
```

Algorithm 1 shows the logic implemented while undergoing the parser-deparser mechanism at the realized P4 NIC's. 40 GE is the practical data rate used i.e. the packets are transferred from sender to receiver at the rate of 40 Gbps and the optical QSFP links are also 40 Gbps enabled. The packets are sent in the burst size of 512 packets at a single instance [line 7]. The instructions in the FPGA bitstream is compared with the headers of the ingress P4 packet and according to the resultant guidelines, particular tasks are performed on the packet. The packet could be extended [line 12] by adding corresponding metadata into it, or it could be compressed [line17] by removing some of the unnecessary headers. The compressor function is implemented to reduce the burden of size length on the overall packet magnitude. When the compressor is realized then the packet gets more space to gather and carry additional route information. The complete algorithm works by initializing the P4 packets at the source side [line 22], then forwarding that packet to the next hops: P4 enabled switch and Carrier Ethernet switch [line 23-24]. And in the final stage, the packet reaches the destination at the sink side and the new call for the next burst of packets is generated [line 26].

VII. PERFORMANCE EVALUATION

Two P4 enabled Alpha-Data KU3 NIC's are implemented and configured on the different Dell servers. An appropriate FPGA image having the mechanism of match-action tables is flashed on both the servers using Vivado software. The sender NIC (installed on the server) is connected to the P4 enabled Inventec switch, which is further linked to the Ciena 8700 Ethernet carrier switch. The carrier switch is then attached to the receiver NIC (installed on the server). 40 GE optical QSFP links are used to connect all the hardware devices. The realization of the P4 NIC's is achieved using Centos 7.5 OS having kernel version as 3.10.0-693.el7.x86_64. The latest DPDK version of 19.02 is used to access all the updated packet libraries.

- The P4 protocol overhead is calculated as:

$$\frac{P4\ Packet\ Size - Packet\ Payload\ Size}{P4\ Packet\ Size} = \frac{305-231}{305} = 24\%.$$

The overall P4 overhead comes out to be 24%. This overhead will reduce with expansion in the packet size.

- Protocol efficiency of the P4 technology:

$$\frac{Packet\ Payload\ Size}{P4\ Packet\ Size} = \frac{231}{305} = 76\%.$$

The efficiency of the P4 protocol used for the transmission comes out to be 76%.

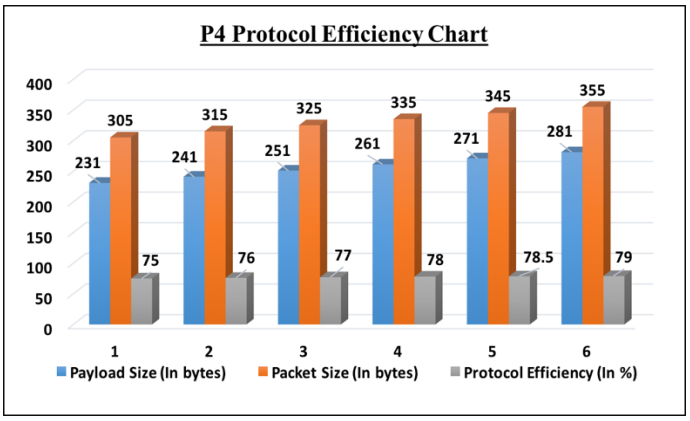

Chart 1. P4 Protocol Efficiency.

- Throughput is calculated as follows:

Efficiency × Net bit rate = 75% × 40 Gbps ≅ 30 Gbps.

With protocol overhead of 24%, efficiency of 75%, data rate of 40 Gbps, the maximum network throughput comes out to be 30 Gbps when 40 GE optical QSFP are implemented for network connections.

- Latency

In P4 Testbed, the consistent latency of 360 microseconds (or 0.00036 seconds) is observed against different time instances. This is a healthy indication that the system is delivering quality output as the packets are being processed and forwarded to the

next stage with minimum delay. Lower the packet latency, the higher will be the efficiency.

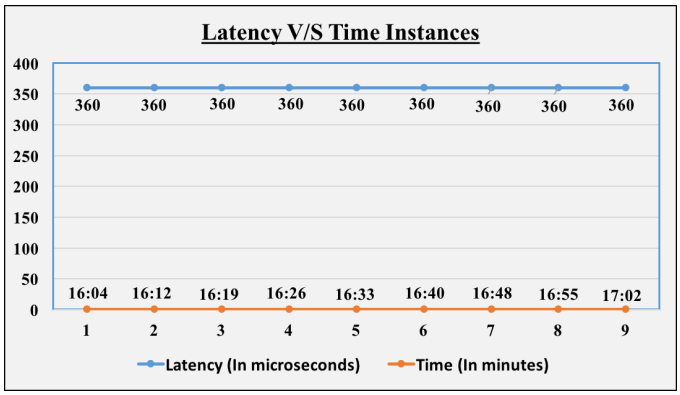

Chart 2. Latency V/S Time Instances.

- Queue Status

The queue status is the total number of P4 packets in the row waiting for further instructions. The average number of egress queue status is taken into consideration and it is observed that there is 24200 number of packets at various time occurrences. The queue size remains the same which depicts that the buffer is diligently performing its task of storing the specified amount of P4 packets. The queue size will gradually increase with the augmentation of packets. More buffer capacity as well as extra cores and queues will be required at the NIC's whenever the need to accommodate a large number of packets emerge.

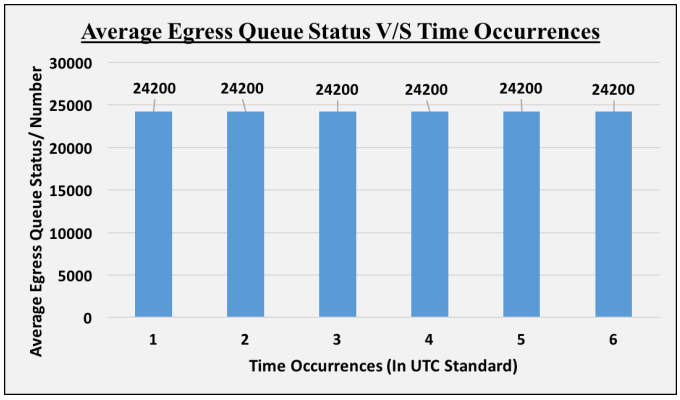

Chart 3. Average number of P4 packets in the egress queue versus various time instances.

VIII. RESULTS AND DISCUSSION

75% of the P4 protocol efficiency is gained when the packet size of 305 bytes along with 231 bytes of payload is transmitted over the 40 GE data rate. It is observed that when the packet size is increased by a significant value of 10 bytes then there is a subsequent rise in the overall efficiency by almost 1%. Minimum latency of 360 microseconds is observed at particular time periods with the same packet size of 305 bytes. With this latency, the future implementation could be performed on the 5G networks which require ULLC (Ultra Low Latency Calculation). As the developed P4 Testbed is having very low latency and significantly high throughput, therefore, 5G network scenario could be implemented on this Testbed for future realizations.

IX. FUTURE TASKS

Increasing the data rate from 40 GE to 100 GE is one of the essential endeavors to be implemented, in order to fulfill the bandwidth requirement of future users. For the actual realization of 100 GE data rate, there are some prerequisites to be met. Firstly, 100 GE P4 enabled NIC's are required to be installed, then there is the need for particular FPGA images to be flashed on those NIC's. Corresponding optical QSFP's are required which can carry 100 GE P4 packet rate. Unnecessary P4 packet headers that are not required for the application could be removed to save memory and to give more space to the INT sections to gather and store more metadata from the intermediate communicating nodes. As the P4 Testbed is having ULCC (Ultra Low Latency Calculation), therefore the Testbed could be used to test the 5G network scenarios. IoT networks may also be connected with the realized communication devices to have divergent applications. Traffic management algorithms incorporating Machine Learning (ML) and Artificial Intelligence (AI) could be merged with the present architecture in order to have divergent results.

REFERENCES

[1] Gartner, "Forecast: Internet of things - endpoints and associated services, worldwide, 2016," Gartner, Tech. Rep., January 2017.

[2] Jonghwan Hyun, Nguyen Van Tu and James Won-Ki Hong, "Towards Knowledge-Defined Networking using In-band Network Telemetry", *IEEE- NOMS (Network Operations and Management Symposium)*, pp. 1-7, 2018.

[3] Anatoliy Malishevskiy et al., "OpenFlow-based Network Management with Visualization of Managed Elements", *IEEE- Third GENI Research and Educational Experiment Workshop*, pp 73-74, 2014.

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese and D. Walker, "P4: Programming Protocol-Independent Packet Processors", *CRC*, pp. 88-94, July 2014.

[5] Salvatore Signorello, Jerome Francois, "NDN.p4: Programming Information-Centric Data-Planes", *IEEE NetSoft Conference (NetSoft)*, pp. 384-389, 2016.

[6] T. Braun, A. Mauthe, and V. Siris, "Service-centric networking extensions," *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, pp. 583–590, 2013.

[7] Abdulrahman Almohaimeed and Abu Asaduzzaman, "Introducing Edge Controlling to Software Defined Networking to Reduce Processing Time", *IEEE- CCWC (Annual Computing and Communication Workshop and Conference)*, pp. 585-590, 2019

[8] F. Paolucci, F. Cugini and P. Castoldi, "P4-based Multi-Layer Traffic Engineering Encompassing Cyber Security", *Optical Society of America*, 2018.

[9] Abbas Yazdinejad, Ali Bohlooli, Kamal Jamshidi, "P4 to SDNet: Automatic Generation of an Efficient Protocol-Independent Packet Parser on Reconfigurable Hardware", *IEEE- ICCKE (International Conference on Computer and Knowledge Engineering),* pp. 159- 164, October 2018.

[10] Han Wang, Robert Soule, Huynh Tu Dang, Ki Suh Lee, Vishal Shrivastav, Nate Foster, Hakim Weatherspoon, "P4FPGA: A Rapid Prototyping Framework for P4", *ACM Symposium on SDN Research conference*, pp. 122-135, 2017.