Comparing OpenFlow and NETCONF when Interconnecting Data Centers

Thomas Kunz and Karpakamurthy Muthukumar
Systems and Computer Engineering
Carleton University
Ottawa, Canada
tkunz@sce.carleton.ca

Abstract— **SDN technology has been applied to a range of different networks, ranging from Ethernet services to large cloud environments. More recently, interest has turned towards extending programmability of Optical Transport Networks (OTN). In the SDN architecture, SBIs are used to communicate between the SDN controller and the switches or routers in the network. In this paper, we deploy OpenFlow and NETCONF as SBIs in managing BoD across interconnected data centers over OTN. More specifically, we use these protocols to communicate between an OpenDayLight controller and two BTI7800 network elements that interconnect the data centers. We present experimental results for both BTI's YANG-based NETCONF implementation and our port of OpenFlow for a number of use cases. Our results show that NETCONF is faster and requires fewer control message. However, OpenFlow offers better bandwidth utilization over the interconnecting link.**

Keywords— SDN, SBI, OpenFlow, NETCONF, data centers, Bandwidth-on-Demand

I. INTRODUCTION

Recent years have witnessed an unprecedented growth in the number of data centers being built by large cloud service providers. To provide flexible and reliable services at the global scale, cloud service providers have deployed multiple data centers in different geographical areas, often spanning continents that are interconnected via private high-speed backbone networks, offering hundreds of Gbps or tens of Tbps bandwidth [9].

The inefficiency of current inter data center backbone networks stems from three aspects [9]. First, the lack of effective control techniques prevents the efficient use of the network resources. With no coordination, each application or service now can send however much traffic whenever it wants, irrespective of the current network load. As a result, the bandwidth needs to be over-provisioned in the network to be able to handle the superimposed peak demand. In reality, with a little coordination, the demand for bandwidth could be reduced by postponing the delivery of delay-tolerant services to off-peak periods. Second, the widely varying performance requirements of applications are usually ignored. For example, interactive applications (e.g. web search) are delay-sensitive, while background applications (e.g. data synchronization between data centers) are throughput-sensitive. Third, it is known that traffic engineering with traditional distributed routing protocols (e.g. link state) is suboptimal in most cases. Distributed approaches are often inflexible and hardly lend themselves to the deployment of sophisticated resource sharing principles such as fair bandwidth sharing between services with priorities or multi-path forwarding to balance application traffic in response to link failures and demand changes.

The emerging Software Defined Networking (SDN) technique has recently been used for inter data center traffic management to address the above-mentioned inefficiencies of traditional approaches. Transport networks are evolving to be more and more automated and driven by software to minimize the operational costs and to provide new services and applications in a quicker and more efficient manner [14]. Several transport technologies such as Dense Wavelength Division Multiplexing (DWDM) and Reconfigurable Optical Add Drop Multiplexer (ROADM) etc. are available for interconnecting the data centers, each of which provides various transport optical features [7]. Moreover, management, configuration or debugging procedures remain a daunting task in data centers mainly because of multiple vendor-specific proprietary assets such as switches/routers requiring their own proprietary procedures rather than a simple and unified process. Similarly, traffic management and policy enforcement can become important and critical issues, as data centers are expected to continuously achieve high levels of performance [6].

In this paper, we compare OpenFlow and NETCONF, two protocols that serve as South Bound Interfaces (SBIs) in handling Bandwidth-on-Demand (BoD) requests across the data centers that are interconnected with the help of an optical backbone and managed using SDN. The protocol performance is evaluated in a testbed comprising two BTI7800 devices interconnecting two data centers via a 100 Gbps optical interconnect. Depending on the availability of resources and the nature of traffic demands, the SDN controller dynamically determines the flow rules and communicates those rules to the network elements using the implemented SBIs.

The paper is organized as follows. Section II reviews related work, Section III describes our testbed. Various use cases and experimental results are discussed in Section IV and we conclude the paper in Section V.

II. RELATED WORK

This section reviews the literature related to both managing BoD across interconnected data centers using SDN as well as the use of SDN in the field of optical networks.

978-1-5090-6501-1/17/$31.00 ©2017 IEEE

A. Managing Bandwidth on Demand Across Interconnected Data Center using SDN

In [16] the authors proposed a network driven transfer mode for cloud operations in a carrier SDN environment at layer 2 and layer 3 with the help of a label switched path.

In [8] the authors described the BoD in an evolved multilayer, SDN-based cloud services model for the WAN. The IP layer devices are being controlled using OpenFlow and a proprietary interface (JunOS), while the optical transport layer devices are controlled using CLI.

The proposed solution in [5] relies on a framework called DynPaC (Dynamic Path Computation), which can provide resilient L2 services taking into account the bandwidth and VLAN utilization constraints. The authors demonstrate BoD service provisioning in layer 2 devices using OpenFlow as communication protocol to manage the layer 2 devices. The data centers are connected using the optical network interface at layer 0 and layer 1, managed using a proprietary control interface. The optical layer device does not exhibit dynamicity.

In [9] the authors presented a utility-optimization-based joint bandwidth allocation for IP-based inter data center communication with multiple traffic classes that handles priorities between traffic classes and explicit consideration of the delay requirement that meets their end-to-end communication. The authors demonstrate BoD service provisioning and management of layer 2 devices using OpenFlow as communication protocol.

The authors of [13] propose an SDN-enabled optical transport architecture that meshes seamlessly with the deployment of SDN within the data centers. The proposed programmable architecture abstracts a core transport node into a programmable virtual switch that leverages the OpenFlow protocol for control. With appropriate extensions to OpenFlow, they discuss how SDN provides programmability and flexibility to packet optical data center interconnects.

In summary, most related works with respect to BoD in an SDN environment focuses on introducing flexibility and manageability at higher layers in the network. The optical network infrastructure is either treated as fixed physical links [5][16] or have very minimal capability for adding a flow [8] by using a non-SDN protocol. The authors primarily focus on how SDN helped them to manage data center interconnections at the IP layer.

B. SDN in the Field of Optical Network

Transport networks are evolving to be more and more automated and driven by software to minimize the operational costs and to provide new services and applications faster and more efficiently. In [7] the author claims that, if one wants to extend the SDN approach to the physical photonic layer, the SDN controller must take the analogue nature of the optical transmission into account. In the transport networks the physical impairments of light paths in optical networks can limit the connections that are feasible using SDN.

In [7][11] the authors propose protocol extensions that can capture, process and set power levels in the optical networks. The opportunity to adjust the channel power levels on the fly could allow for the introduction of new optical traffic which would not otherwise be feasible, all the while ensuring that the existing connections remains unaffected.

Francesco Paolucci et al. [12] discuss building a software defined elastic optical network that offers a high degree of flexibility in enabling dynamic configurable light path provisioning and re-optimization. The authors proposed to use the NETCONF protocol to achieve a high degree of convergence and limit the number of utilized protocols.

Marcos Siqueira et al. [15] proposed a Software Defined Optical Transport Network (SD-OTN) architecture. The SDN controller provides a NETCONF/REST interface for plugging in control applications to perform specific tasks by taking advantage of the network abstraction. By modeling these network entities using the NETCONF modeling language YANG, the authors state that they gain the possibility to export or transform the data needed for their configuration, as well as to model their interconnections and restrictions.

The authors of [17] propose the flexi grid optical network as the solution for managing inter data center networks. The environment is composed of an enhanced Software Defined Networking (eSDN) control architecture designed for the application scenario. The authors implemented an extended OpenFlow protocol to support optical switching of spectrum randomly from 50 GHz to 400 GHz based on traffic requests.

In summary, the transport networks are evolving to be more and more automated and driven by software to minimize the operational costs and to provide new services and applications in a quicker and more efficient way [7]. The authors in [12][15] propose NETCONF as an interface for managing the optical network. On the other hand, ONF is working on OpenFlow extensions for optical networks. The authors in [17] have demonstrated optical spectrum switching using a proprietary extended OpenFlow interface.

C. Motivation

Software Defined Optical Transport Networks (SD-OTN) is a recent extension of SDN, which initially focused on IP-based networks [5] [9]. The SBI plays a vital role in managing the data plane devices in an SDN environment. The ONF suggests OpenFlow as SBI to manage the network devices in an SDN environment. As reviewed above, OpenFlow can be implemented over an optical network by defining optical extensions [17] for supporting the optical data plane devices. Others suggested NETCONF as an active protocol for managing optical data plane devices [13][16]. Yet there appear to be no studies directly comparing the use of NETCONF and OpenFlow as SBIs. Therefore, we compare and provide a quantitative and qualitative analysis of both these protocols operating in an optical SDN environment managing BoD across interconnected data centers.

III. TESTBED DESCRIPTION

To evaluate both protocols, we implemented a number of test cases/use case scenarios in a testbed. In our testbed, two BTI7800 network elements interconnect two data centers,

managing a 100G OTU4 link, as shown in Figure 1. The BTI devices communicate with and are controlled by an OpenDayLight (ODL) SDN controller that implements the BoD application. The BTI7800 is a layer zero and layer one optical device that offers data center interconnection as a service. In an SDN environment, the BTI7800 is treated as a data plane device. On the data center side, the BTI7800 supports 12*10 G links that support 12 users of 10 G bandwidth for user traffic each. A Spirent traffic generator acts as user traffic source. The optical router receives the input traffic of 120 G but can only multiplex and forward traffic of 100 G on the OTU4 link interconnecting the two data centers. The SDN controller has to meet the BoD requirements by granting bandwidth to more essential user traffic and dropping other users whose requirement cannot be satisfied.

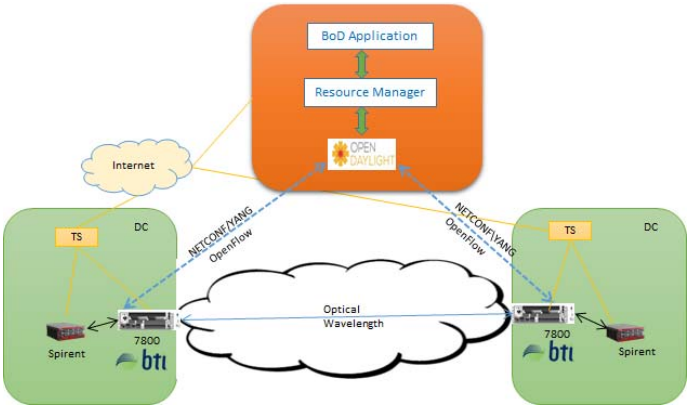

Fig. 1. Network Testbed

Our research is mainly focused on the BTI7800 network elements that interconnect the data centers. Natively, ODL implements both OpenFlow and NETCONF as SBIs, the BTI7800 also already implements NETCONF. BTI also provides a range of YANG [3] models that describe various capabilities of the edge device. These models form the basis for exchanging control messages between the BTI7800 and the SDN controller.

The BTI7800 natively does not support OpenFlow, nor does OpenFlow specification V1.3 released by the ONF, designed to support IP layer devices [1], accommodate any optical layer information to support the BTI7800 optical network elements. Possible transport extensions to OpenFlow are published by the ONF OTWG under TS-022 [2]. To solve this issue we started with OpenvSwitch, an open source library supporting OpenFlow V 1.3. We modified the open source OpenVSwitch code to support the required optical extension, which in our case is the need to establish and tear down optical cross-connects within the device. We then ported this modified version onto the BTI7800 devices. The modified BTI OpenVSwitch plugin helps in creating ports and flow connections that are referenced using OpenFlow protocol messages and managed by an OpenFlow controller such as OpenDayLight (ODL).

To support our required optical extensions, we re-interpreted specific fields of the protocol messages. The controller is aware of the ports available within the device. Now the flow rules for rerouting the packets within the device have to be communicated. The OpenFlow flow modification message [1] is implemented to support the BTI7800 device as follows. In the BTI7800, the nature of the flow is to link two ports, known as cross-connect. The information about the port in the flow modification message uniquely identifies each port and is used to specify the port that needs to be cross-connected. The flow modification information has to contain two end port and their operation (add, modify, delete). The first port ID is carried as a field in the basic message, the second message is carried in the optional instruction field.

IV. Scenarios and Testbed Results

We studied the protocol performance using a range of use cases/scenarios. The use cases are specifically designed to handle BoD for users across data centers with the help of SDN at the optical layer and switching. These use cases are:

- High Priority Flow (20 G) Request for User B
- High Priority Flow (20 G) Request for User A
  - When no low priority flow exists
  - When the lower priority flow exists
- Two Users' Request for High Priority Flow, Mutual Sharing

In addition, we defined and studied use cases where the bandwidth request made from the user(s) is large depending on the scenario. The use cases help us in comparing both protocols when a large number of sub-flows or cross-connects are modified. Finally, we also studied the performance of both protocols in a stress test scenario, when multiple applications try to modify the same data plane device though the control plane simultaneously at the same time. These uses cases are:

- Bandwidth Request
  - Elephant Request (40 G)
  - Emergency Request (100 G)
- Stress Testing

A. Evaluation Metrics

In all cases, the behavior of both protocols as SBIs is observed. The following metrics are considered.

Time: An efficient protocol should exhibit a fast and reliable response. The time taken to complete the required set of configurations by both SBIs are compared. The time period for each use case differs depending on when the system attains the steady state. A use case reaches the steady state when all the configurations are committed, the traffic flows end-to-end in all the links that are active and the system remains in a stable state.

Bandwidth Utilization: The research is focused on handling BoD requests across the interconnected data centers managed using the SDN SBIs. One of the key aspects for adopting SDN in the current network management is because it provides a fast and efficient way to manage the networks. Therefore, resources can be optimally used. For a specific use case the bandwidth utilization is calculated over a period of time from when a use case is triggered by the user until system attains steady state. All the experiments start with the system operating in a defined initial state. New user traffic demands additional bandwidth and the SDN controller decides to service the bandwidth requests based on the nature of the traffic and availability of the resources. The SDN control plane communicates the changes through the implemented SBIs and waits until the system

completes the requirement(s). Once the requirement(s) are completed, the system returns to its initial state. The time period for both SBIs in each use case is equal: if one of the implemented SBIs attains the steady state faster than the other, the environment is maintained in the steady state until the environment implementing the other SBI also attains the steady state. In steady state, if all links are active, the bandwidth utilization is always 100 %.

Control Messages: The ODL SDN controller manages the data plane devices via the implemented SBIs. The SBIs communicate the flow modification information as control messages. Each SBIs handles the messages differently based on the nature of the protocol, which leads to difference in message size and number of messages. An efficient protocol should require less control signaling and therefore induce a lower network overhead.

B. Initial Setup

The initial environment shown in Figure 2 is used for all use cases. Both data centers are identical in network topology and are interconnected with the help of the OTU4 optical link. In both data centers the BTI7800 device acts as an edge node. Each data center has two users that are connected with the BTI7800.

Both data centers have two users, user A and user B respectively. Each user owns a potential bandwidth of 6*10 G connections patched with the BTI7800. The OTU4 100 G link connects both optical routers across the data center, and it is capable of multiplexing user traffic on the optical link connecting data centers. Each user has a dedicated bandwidth of 40 G each, constituting 80 G out of 100 G bandwidth (blue and green links). The sub flows or cross-connects 1 to 4 are termed as the base flow of user A and represented with green links. The sub flows or cross-connects 5 to 8 are termed as the base flow of user B and represented with blue links. One of the users, "user B", is assigned the remaining 20 G for low priority traffic (yellow color links). The sub flows or cross-connects 9 and 10 are termed as the low priority flow of user B and represented with yellow links.

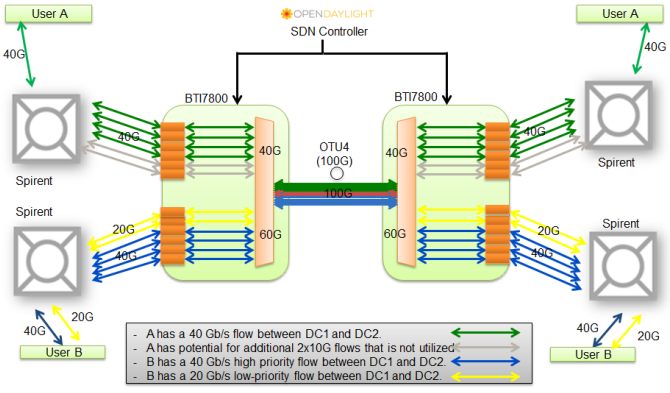

Fig. 2. Initial Environment

The links (arrows) represented within the BTI7800 are known as cross-connects or sub flows. They help in multiplexing/bridging any 10 of 12 (*10 G) ports on to the OTU4 link. Based on the user traffic, the SDN controller signals which 10 ports need to be cross-connected to the OTU4 link.

In the interconnected data centers, for the traffic to flow from one end to another, it is essential to provision the end-to-end tunnel. The application layer first provisions the data center at one end, but on the other end no connection is provisioned yet. Therefore, the traffic cannot reach the other end. Next, the application layer begins to provision the data center at the other end. As a result an end-to-end tunnel is created for each sub flow and eventually traffic begins to flow end-to-end. The BTI7800 YANG model handles cross-connects as a single dataset. The controller does not allow to access them as individual cross-connects and is forced to access them as a set of cross-connects. Using the implemented NETCONF SBI, the list of cross-connects that need to be provisioned are specified, the BTI7800 device receives the request and iteratively processes each cross-connect information specified in the list. At the end of the request it returns the status back to the controller. OpenFlow, unlike NETCONF, operates on each sub flow individually. The application layer provisions both data centers for each flow separately. Eventually we see traffic flowing across data centers. As each flow is provisioned separately, the amount of time the link is not utilized is lower with OpenFlow, compared to NETCONF. The time taken to create or tear down the cross-connect(s) are specific to the BTI7800 devices. The authors of [4] report a hardware time of 7 seconds to setup a cross-connect in an extended optical OpenFlow environment, the hardware setup time varies for different devices based on their performance and functionality.

C. High Priority Bandwidth Request by User A

We start with the environment in the initial state discussed above. User A has high priority traffic and requires additional bandwidth. The low priority traffic of user B loses its bandwidth and user A is assigned this bandwidth to satisfy the high priority traffic demand. On completion of the high priority traffic demands, the environment returns back to the initial state. The controller will release the bandwidth occupied by the high priority traffic by user A and assign the flows back to the low priority traffic of user B. Both users A and B have dedicated bandwidth of 40 G each, and the 20 G of lower priority traffic is assigned to user B. The SDN controller, using the implemented SBI, will signal the BTI7800 network element to drop the bandwidth pertaining to the low priority traffic and will establish the bandwidth for the high priority traffic belonging to user A for a time period of 8 seconds.

The experiments begins at time t = 10 sec. At time t = 10 sec the application layer signals the resource manager regarding the bandwidth request for the high priority flows from user A. Two cross-connects need to be torn down and two new cross-connects are added. In NETCONF and OpenFlow, at t = 23 sec and t = 25 sec, both sub flows are provisioned. It takes 13 seconds and 15 seconds for NETCONF and OpenFlow respectively to provision the high priority request. NETCONF takes 6.5 seconds to provision one data center, it takes 0.5 second to delete both cross-connects and 3 seconds to add a single cross-connect. Using OpenFlow, it takes 3 seconds to add a cross-connect, and 0.75 seconds to delete a single cross-connect. The provisioning of a cross-connect takes longer

hardware setup time because an optical tunnel between the two interconnecting ports is created. During the deletion of a cross-connect, the entry is removed from the internal flow table and then ports are shut down, resulting in a reduced hardware setup time. The high priority traffic is serviced for 8 seconds. We then return back to the initial state by dropping the two high priority sub flows and adding the two low priority sub flows.

Time: The OpenFlow protocol takes longer compared to NETCONF because each sub flow modification is processed separately. The resource manager communicates which particular sub flow needs to be modified one after another. The implemented OpenVSwitch algorithm posts the cross-connects modification as request message within the queue and the kernel module processes the requests one by one. The cross-connects that need to be deleted and added are posted at both ends one after another. There is a possibility that the cross-connect modification request is posted on the queue before the kernel completes processing of the predecessor modification request. In NETCONF, the resource manager communicates the set of cross-connects the need to be provisioned and the NETCONF agent within the BTI7800 identifies the cross-connects that are modified by comparing the information with the existing provisioned cross-connects.

Bandwidth Utilization: The NETCONF scenario has a bandwidth utilization of 92.78 % and the OpenFlow scenario bandwidth utilization of 95.85%. The OpenFlow interface deletes a sub flow and adds a sub flow and it repeats the procedure for the number of sub flows to be modified. The bandwidth utilization for both protocols are calculated from the start of experiment t = 10 and until t = 53 sec when both scenarios remain in the steady state for a few seconds.

D. Control Messages

The cross-connects that need to be provisioned are specified as a set of cross-connects in a single NETCONF flow modification message. The message has a constant size of is 2612 bytes. As our testbed involves two BTI7800 devices, two control messages of 2612 bytes are communicated to both BTI7800 devices. The response message from a BTI7800 device has a size of 52 bytes.

OpenFlow communicates the flow modification messages depending on the number of cross connects that need to be modified. The size of the OpenFlow flow modification message from the controller to the BTI7800 device is 206 bytes and the response is 66 bytes. If existing cross-connects are to be modified, this requires additional messages to deleted them first. For a full reconfiguration of all 10 cross-connects, a total of 20 messages are communicated to a single BTI7800 device, for a total size of 4120 bytes. The 20 responses add up to another 720 bytes of network traffic, received at the controller.

E. Summary of Results

A detailed discussion and analysis of each use case can be found in [10], we summarize the overall results here only, due to the space limitations. One key performance differentiator we observed is the time it takes to modify (add or delete) cross-connects. Table 1 summarize the time taken by both SBIs to modify different numbers of cross-connects for all use cases. Table 2 summarizes the key performance metrics of all our use cases.

V. Conclusions and Future Work

In the era of intense high bandwidth demand growth, and unpredictably shifting traffic patterns, operators need their transport networks to become dynamically programmable in order to offer new services and to satisfy the traffic demand without over-provisioning the network resources. SDN-based transport optical networks allow the resources to be governed by policy management, enabling the network operators to transform their transport networks to function efficiently and also increases operational agility. We evaluate two popular SBIs, NETCONF and OpenFlow, in managing the BoD across the transport optical interconnect. The behavior of both implemented SBIs is different based on the nature of the protocol.

TABLE I. CROSS-CONNECTS MODIFICATION TIME

Number of Cross-Connects Provisioned	**NETCONF (sec)**		**OpenFlow (sec)**	
	Add	Delete	Add	Delete
1	3	0.5	3	0.75
2	6	0.5	6	1.50
4	12	1	12	3
6	18	1	18	4.50
10	31.5	1	30.5	7.50

TABLE II. USE CASES RESULTS

Use case	***NETCONF***			***OpenFlow***		
	Time (sec)	Link (%)	# msgs	Time (sec)	Link (%)	# msgs
Initial	63	32.0	2	61	53.9	10
High Priority Flow (20 G)						
User B Request	-	100	-	-	100	-
	User A Request					
No Low Priority Flow Exists	31	94.4	4	34	93.4	8
Lower Priority Flow Exists	45	92.78	4	49	95.9	16
User A and B	31	95.6	4	33	95.7	8
Elephant Request						
Elephant Request	104	77.8	6	117	92.9	48
Emergency Request						
Emergency Request	144	59.6	6	169	92.4	80

The results are specific to BTI7800 environment. NETCONF references the BTI YANG model to represent and communicate the information from the SDN controller. BTI's YANG-based NETCONF is faster, and more efficient in terms of the number of control messages communicated between the controller and the BTI7800 device, and reduces the complexity of the resource manager. On the other hand, OpenFlow accesses each cross-connect individually; therefore, it efficiently handles the bandwidth by minimizing the sub flow idle time, but the protocol is slower and the number of control messages increases

proportionally to the amount of flow rules modified. If it is acceptable for the link to be idle for a few seconds, NETCONF is a better protocol. OpenFlow is a better protocol if bandwidth being idle is an issue and can compromise with time and the network overhead.

An increase in the number of cross-connects that need to be provisioned for each data center will affect the performance of the OpenFlow protocol as the control messages and the time taken to handle each message increases with the number of cross-connects that need to be modified. In case of NETCONF, the control message indicating the modified set of cross-connects will still be a single message, however, the time taken to process the message at the edge device will be linear in the number of flows that need to be modified.

Some suggestions are presented here to enhance this work in the future. Having a protocol that can access a set of cross-connects and each cross-connect individually based on the use case is a best approach. Aggregation of OpenFlow flow rules is a possible solution to overcome the issues (time and number of messages communicated between the controller and the device) faced by the implemented OpenFlow protocol. The OpenFlow standard and ODL controller do not currently support aggregation of flow rules. On the other hand, flow table size and aggregation of flow rules in the core switches is a relatively new topic of SDN.

Parallelization of the control messages issued from the SDN ODL controller to the BTI7800 network elements located at the sender and receiver interconnected data centers will speed up the process. Issuing the cross-connect information to both the BTI7800 network elements at the same time by introducing multithreading in the resource manager will help to reduce the time for each use case by 50%. The synchronization between both threads will become important for maintaining a stable system, in particular in cases where requests succeed at one end of the connection but fail at the other end.

ACKNOWLEDGMENT

The authors would like to thank BTI Systems (now Juniper Networks) and particularly Peter Landon for supporting this work and providing access to the testbed

REFERENCES

[1] OpenFlow Switch Specification, Version 1.3.0 (Wire Protocol 0x04), June 25, 2012, ONF TS-006. (https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf)

[2] Optical Transport Extensions, Version 1.0, March 15, 2015, ONF TS-022. (https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/ Optical_Transport_Protocol_Extensions_V1.0.pdf)

[3] YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), IETF, Request for Comments: 6020, Category: Standards, ISSN: 2070-1721.

[4] Mayur Channegowda; Reza Nejabati; Dimitra Simeonidou, "Software Defined Optical Networks Technology and Infrastructure: Enabling Software Defined Optical Network Operations", IEEE, Pages: A274 - A282, 23 October 2013.

[5] Robert Doverspike et al., "Using SDN Technology to Enable Cost-Effective Bandwidth on Demand for Cloud Services", IEEE/OSA Journal of Optical Communications and Networking, Pages: A326 - A334, Febuary 2015.

[6] Kreutz; F. Ramos; P. Esteves Verissimo; C. Esteve Rothenberg; S. Azodolmolky; S. Uhlig, "Software Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, Pages: 14 - 76, January 2015.

[7] Anna Lidia Soso; Gianmarco Bruno; Jeroen Nijhof , "DWDM Optical Extension to The Transport SDN Controller", Fotonica AEIT Italian Conference on Photonics Technologies, Pages: 1-4, May 2015.

[8] Alaitz Mendiola et al., "Multi-Domain Bandwidth on Demand Service Provisioning Using SDN", 2016 IEEE NetSoft Conference and Workshops (NetSoft), Pages: 353 – 354, June 2016.

[9] Jason Min Wang; Ying Wang; Xiangming Dai andBrahim Bensaou," SDN-based Multi-Class QoS-Quaranteed Inter Data Center Traffic Management", Cloud Networking (CloudNet), IEEE 3rd International Conference, Pages: 401 - 406, October 2014.

[10] Karpakamurthy Muthukumar, "Analysis of OpenFlow and NETCONF as SBIs in Managing the Optical Link Interconnecting Data Centers in an SDN Environment." Masters's Thesis, Carleton University, Ontario, Canada, January 2017

[11] Vijoy Pandey, "Towards Widespread SDN Adoption: Need for Synergy Between Photonics and SDN Within the Data Center", IEEE Photonics Society Summer Topical Meeting Series, Pages: 227-228, July 2013.

[12] Francesco Paolucci; Andrea Sgambelluri; Nicola Sambo; Filippo Cugini; Piero Castoldi; "Hierarchical OAM Infrastructure for Proactive Control of SDN-Based Elastic Optical Networks", 2015 IEEE Global Communications Conference (GLOBECOM), Pages: 1 – 6, December 2015.

[13] S. A. Shah; J. Faiz; M. Farooq; A. Shafi; S. A. Mehdi, "An Architectural Evaluation of SDN Controllers", 2013 IEEE International Conference on Communications (ICC), Pages: 3504 – 3508, June 2013.

[14] Abhinava Sadasivarao; Sharfuddin Syed; Ping Pan; Chris Liou; Inder Monga; Chin Guok and Andrew Lake, "Bursting Data Between Data Centers: Case for Transport SDN", IEEE 21st Annual Symposium on High-Performance Interconnects, Pages: 87 - 90, August 2013.

[15] M. Siqueira and J. Oliveira, "An Optical SDN Controller for Transport Network Virtualization and Autonomic Operation", Globecom Workshops (GC Wkshps), IEEE, pages: 1198-1203, December 2013.

[16] L. Velasco; A. Asensio; J. L. Berral; A. Castro; V. López, "Towards a Carrier SDN: An Example for Elastic Inter-Datacenter Connectivity", Optical Communication (ECOC 2013), Pages 1-3, September. 2013.

[17] Yongli Zhao; Jie Zhang; Hui Yang and Xiaosong Yu, "Data Center Optical Networks (DCON) with OpenFlow Based Software Defined Networking (SDN)", Communications and Networking in China (CHINACOM), Pages: 771 – 775, August 2013.