ARID ZONE JOURNAL OF ENGINEERING, TECHNOLOGY & ENVIRONMENT

AZOJETE - Centre for Satellite Technology Development Special Issue:

Space Science and Technology for Sustainable Development

AZOJETE, June, 2019. Vol. 15(SP.i2):242-254

Published by the Faculty of Engineering, University of Maidiguri, Maidiguri, Nigeria.

Print ISSN: 1596-2490, Electronic ISSN: 2545-5818

www.azojete.com.ng

ORIGINAL RESEARCH ARTICLE

DEVELOPMENT OF A MODIFIED EAST-WEST INTERFACE FOR DISTRIBUTED CONTROL PLANE NETWORK

E. A. Adedokun* and A. Adekale

Department of Computer Engineering, Ahmadu Bello University, Zaria, Nigeria

ARTICLE INFORMATION

Submitted: 10 May, 2018

Revised: 01 September, 2018

Accepted: 05 September, 2018

Keywords:

Software-defined networks

Distributed control plane

East-West interface

High Availability

Multi-domain

Inter-controller Communication.

ABSTRACT

The East-West Interface is important in achieving communication in a distributed control plane network such as a Wide Area Network (WAN); to enable scalability and distribution of the control plane. In this paper, a Modified Communication Interface for Distributed Control Plane (mCIDC) was developed to ensure communication in WANs. The mCIDC interface allows the synchronization of different modules in the controller to enable consistent high availability and efficient communication among controllers in the East-West Interface needed for Software Defined Network (SDN) to scale in a WAN environment. The modified-CIDC (mCIDC) is developed based on the Communication Interface for Distributed Control Plane (CIDC) and implemented on top of Floodlight Controller using the ISyncService module. The performance of the mCIDC and CIDC was compared using captured Transmission Control Protocol (TCP) packets, TCP errors and inter-controller communication overload (ICO). The results indicated that for Claranet_2; mCIDC showed a better performance in minimizing number of Captured TCP Packets, TCP Errors and ICO by 26.55%, 17.89%, and 19.35% respectively when compared with CIDC, while for Claranet_3; 15.82%, 21.60% and 29.25% for Captured TCP Packets, TCP Errors and ICO respectively, when compared with CIDC. This shows that the mCIDC ensures communication by transmitting the necessary required packets (information) among controllers with reduced TCP errors and fewer overloads

1.0 Introduction

Traditional networks (networks that require manual configuration) are faced with many challenging concerns (Chen *et al.,* 2015) such as: implementing high-level policies in network, manually configuring each network device using low-level configuration and meeting up with the dynamic nature of today's applications. Software Defined Network (SDN) is a new networking paradigm that addresses the limitations of traditional networks by simplifying network management, enable innovation and evolution of networks (Kreutz *et al.,* 2015). SDN concept involves separating the control function from networking devices to a single point of control. The SDN architecture decouples the network control in the control plane and the forwarding functions in the data plane enabling the abstraction of the network for applications and network services. It transfers the intelligence from the traditional network devices to a centralized control plane and enables network programmability. SDN architecture is vertically split into three functional layers (or planes) (Jarraya *et al.,* 2014): Infrastructure Layer (Data Plane), Control Layer and Application Layer. The SDN Controller in the Control plane interacts with these functional layers through the Northbound Application Programming Interface (API), Southbound API and East/Westbound API. This APIs enable communication across the SDN architecture, making the architecture dynamic, manageable, cost-effective and adaptable (Rao *et al.,* 2016).

SDN was initially designed for testing new protocols in campus networks, but thereafter several works have shown that this concept is suitable for several types of networks such as data centers, Wireless networks, and Wide Area Network (WAN) (Benamrane *et al.,* 2017). The benefits of SDN can be achieved in multi-domain networks such as WAN by distributing the SDN control plane. The distributed control plane addresses the single point of failure (SPOF) problem; which makes SDN architecture highly vulnerable to attacks (Kreutz *et al.,* 2013). It also addresses the challenges of scalability and performance. The distributed control plane involves the use of multiple controllers and this is divided into: logically centralized and logically distributed control planes (Blial *et al.,* 2016). Large networks such as WAN with complex infrastructure and protocols use the logically distributed control plane. The logically distributed control plane does not need extensive synchronization between controllers for global network view. It enables each controller to manage its own domain and distribute necessary data to other controllers (Benamrane *et al.,* 2017). This control plane can be designed vertically (hierarchical) or horizontally (flat).

Communication between the multiple controllers in the distributed control plane is of primary importance (Benamrane *et al.,* 2017). This communication addresses scalability and ensures that requirements of SDN in WAN such as global knowledge, real-time monitoring, decision making, policy updating and fined-grained control (Liu and Lil, 2015) are achieved. The communication between the controllers at the

control plane is handled by the East-West API (or Interface). To identify and provide common compatibility and interoperability between different controllers, it is necessary to have a standard east/westbound interface (Kreutz *et al.,* 2015). There is no standard for the East-West Interface (Jarraya *et al.,* 2014). In this paper; a modified east-west interface based on the Communication Interface for Distributed Control Plane (CIDC) by (Benamrane *et al.,* 2017), was developed. The mCIDC provides high availability using the floodlight IsyncService, enabling consistent and efficient communication in the East West Interface.

1. Literature Review

In addressing the challenges faced by the physically centralized SDN control plane architecture such as SPOF, scalability and limited processing capacity, the distributed control plane architecture is used. This architecture consists of the logically centralized and logically distributed approaches (Benamrane *et al.,* 2017). In the logically centralized approach, the controllers collaborate to manage the network and have the same view of the network using the same shared database. In the logically distributed approach, each domain is managed by its controller and can share only some useful information with the other controllers to achieve some services such as the topology view.

Most of the distributed control plane architectures in literature with a logically centralized approach such as Onix, Hyperflow, Elasticon and ONOS, currently cannot manage inter-domain flows between SDN domains (Wibowo and Gregory, 2016). To achieve a scalable control plane in large distributed multi-domain networks (such as WAN), the logically distributed control plane architectures are proposed. Some literatures suggest that the east-west interface communication is very important in achieving scalability in multiple controllers such as Software Defined Networking Interface - SDNi (Yin *et al.,* 2012) and East-West Bridge - EW Bridge (Lin *et al.,* 2013). The East-West Interface is the communication channel that exists solely for distributed SDN controllers (Oktian *et al.,* 2017); providing inter-domain communications. However, there is no standard for the east-west interface.

Lin *et al.* (2013) designed a high-performance mechanism called East-West Bridge (EW Bridge), for heterogeneous SDN domains to exchange network view in multi-domain network. The EW Bridge is enabled in the controllers/network operating systems (NOSes) for each SDN domain by adding network virtualization, EW Bridge and Link Layer Discovery Protocol (LLDP) extension to guarantee communication between controllers. Messages are delivered for update by the publish/subscribe system and each database stores the entire topology locally. EW Bridge focuses on enterprise, datacenter and intra-domain networks. Inter-domain networks are not considered.

Phemius *et al.* (2014) proposed an extensible Distributed SDN Control plane (DISCO) for WAN and overlay networks. The DISCO controller is composed of intra-domain

Corresponding author's e-mail address: wale@abu.edu.ng

and inter-domain. The intra-domain monitors the network and manages flow prioritization, while the inter-domain manages communication with other DISCO controllers. An Extended Database in each controller is used to store network topology information from both domains. The inter-domain enables the exchange of aggregated network-wide information using Messenger and Agents. The Messenger is implemented using the Advanced Message Queuing Protocol (AMQP). The Messenger Module builds channels between neighboring controllers while each Agent publishes and consumes message through the messenger. DISCO uses four main agents (Connectivity, Monitoring, Reachability and Reservation) to support network-wide functionalities and ensure consistency.

Benamrane *et al.* (2017) implemented Communication Interface for Distributed Control plane (CIDC) was implemented for inter-controller communication using a logically distributed architecture. The CIDC interface was implemented in a WAN, and used by each controller in the network to synchronize its stats and services with neighboring controllers; for inter-controller communication. The role of each controller in the network was customized with three communication modes: Notification, Service and Full. The CIDC Interface shares necessary data among controllers based on the desired communication mode.

3. Materials and Methods

3.1 Materials

The mCIDC was implemented on top of Floodlight controller. The mCIDC modules and ISyncService were developed in Java. The experimental emulation was carried out on a 64-bit Windows 8 operating system, Intel Core i5 CPU- 3.20GHz with 8GB RAM.

3.2 Description of mCIDC

mCIDC followed the concept of the CIDC interface by Benamrane *et al.* (2017). The mCIDC considered the logically distributed control plane architecture with multiple controllers. The mCIDC interface is composed of four essential modules: Consumer, Producer, DataUpdater and DataCollector. The modules are connected and communicate with the core elements of the controller. Each controller plays the role of a consumer for external events (events outside its network) and Producer for local events (events within its network). The mCIDC uses the Netty framework to reduce resource consumption. The interface uses the Notification (the Producer notifies all remote controllers when changes occur in its domain) and Full (the Producer shares all events and services) communication modes. The mCIDC uses an event-driven paradigm (actions are made by events). The controller is active, when new events such as packet-in are received. Events are determined by event listeners and observers in the controller. Figure 1 shows the flowchart of mCIDC.

Corresponding author's e-mail address: wale@abu.edu.ng

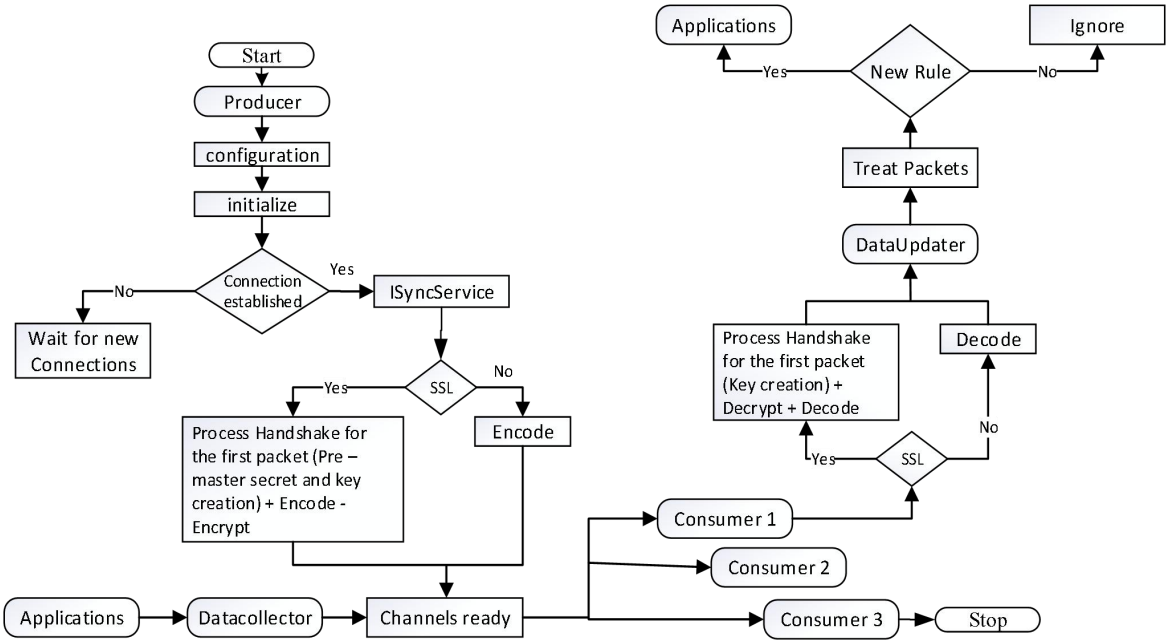

Figure 1: Flowchart of mCIDC

3.3 mCIDC Operation

The mCIDC used the ISyncService to synchronize updates between controller's state by providing access to updates published by all other modules in the controller in an efficient manner. The controller starts up initializing the default modules and states such as forwarding and topology manager. The Producer starts by reading the configuration to recognize the IP addresses of the neighboring controllers. The Producer uses the ISyncService and initializes connection with the consumers, and joins each consumer that respond to a list of connections. State information such as topology and link discovery are shared using ISyncService. The channels are configured and ready for any new events from the Data-Collector. When changes occur (new events are detected), the Data-Collector sends network status to all connected Consumers to notify remote controllers that a local status has changed and to synchronize data between controllers. The Data-Updater receives external data from the Consumer and updates the system.

3.4 Testbed and setup

The network environment was emulated on three Virtual Machines (VMs), where each VM represents a WAN domain. The VMs are emulated using Ubuntu 16.04 LTS. The VMs emulate the Claranet network topology in Figure 2, obtained from the Internet Zoo Topology, using Mininet. The VMs are emulated using Graphical Network Simulator 3 (GNS3), and each VM contains the local network composed by Open Virtual Switch (vSwitch), to which one virtual host is attached, and an SDN controller. The Controllers form a full mesh on connecting with one another. With each controller managing one domain, the link between switches and hosts is set to 1 Gbps for the bandwidth and 30 ms for the one-way delay (Benamrane *et al*, 2017).

Corresponding author's e-mail address: wale@abu.edu.ng

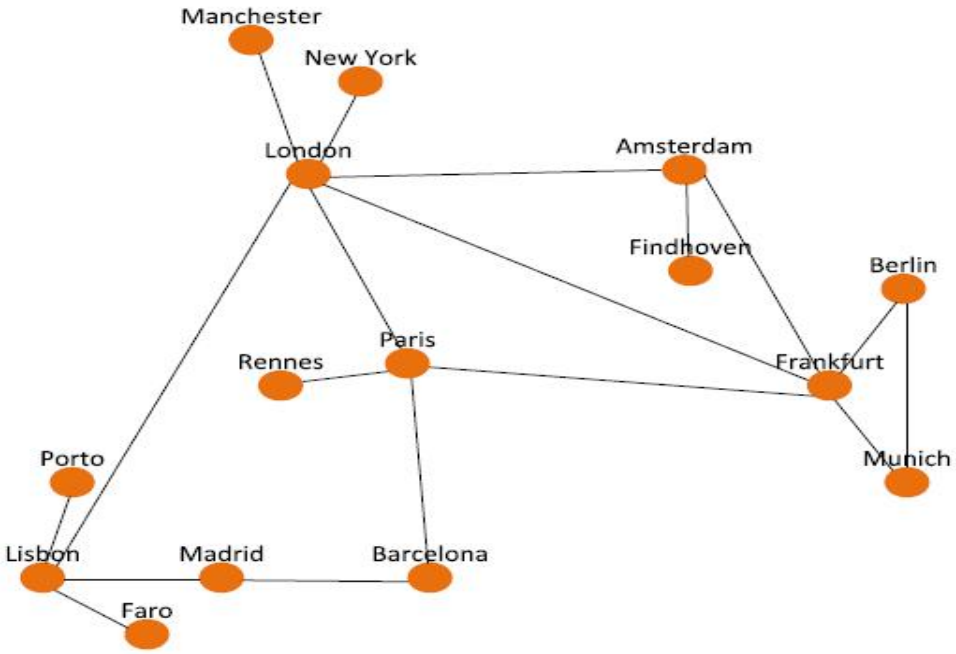

Figure 2: Claranet Network Topology (Benamrane et al. 2017)

3.5 Experimental Description

The VMs are powered "ON" from the GNS3 interface, starting all three VMs. Each VM represents a WAN domain. Each domain controller is launched. The network topology in each domain is started using Mininet. OpenFlow 1.3 is used as the southbound protocol. After the successful establishment of OpenFlow connections between the nodes (hosts and switches) and the controllers, then new traffic is injected into the network using the Iperf tool, for a default time of 10 seconds. This is achieved by simulating a scenario where switches must request their controller for new events processing, by fixing one of the emulated hosts as a server and the remaining hosts as clients. New flow from the client passes through the switch, sending a packet-In message to its controller. The controller installs forwarding rule in the switch flow table. These operations result in network changes, causing the controller to synchronize with its neighboring controllers (Benamrane *et al.* 2017). Wireshark analyzer was used for network analysis, the experimental description was carried out on CIDC and mCIDC using notification mode. The experiment was carried out on different Claranet network topology size as shown in Table 1.

Table 1: Network Topology

TOPOLOGY	PSEUDO-NAME	DOMAIN	NODES PER DOMAIN	NODES
Claranet	Claranet_2	2	15	30
Claranet	Claranet_3	3	15	45

Corresponding author's e-mail address: wale@abu.edu.ng

3.6 Performance metrics

Synchronization metrics such as TCP Packets Captured, TCP errors and Inter-controller Communication Overload (ICO) that could affect response time and controller performance in a distributed architecture are considered as performance metrics. The performance metrics are obtained from the Wireshark Analyzer. These performance metrics are used to evaluate the necessary data to synchronize information.

i. TCP errors

These are the TCP errors during conversations between client and server. These errors include: Out-of Order, Dup Ack, Lost Segment, Fast Retransmission, Windows Update and so on.

ii. Captured TCP packets

This is the TCP Packets captured using Wireshark Analyzer, between clients and server. The TCP Packets represents the necessary information transmitted across different controller domains.

iii. Inter-controller Communication Overload (ICO)

This is the total rate of bidirectional traffic (Kbit/s) exchanged between controllers, during transmission.

4. Results and Discussion

In this section; simulation results for the Claranet network topology are provided. For each metric, we obtained results for the CIDC and mCIDC, using the Notification mode. Claranet topology with different sizes was used to test, and compare the performance of CIDC and mCIDC.

4.1 Captured TCP Packets

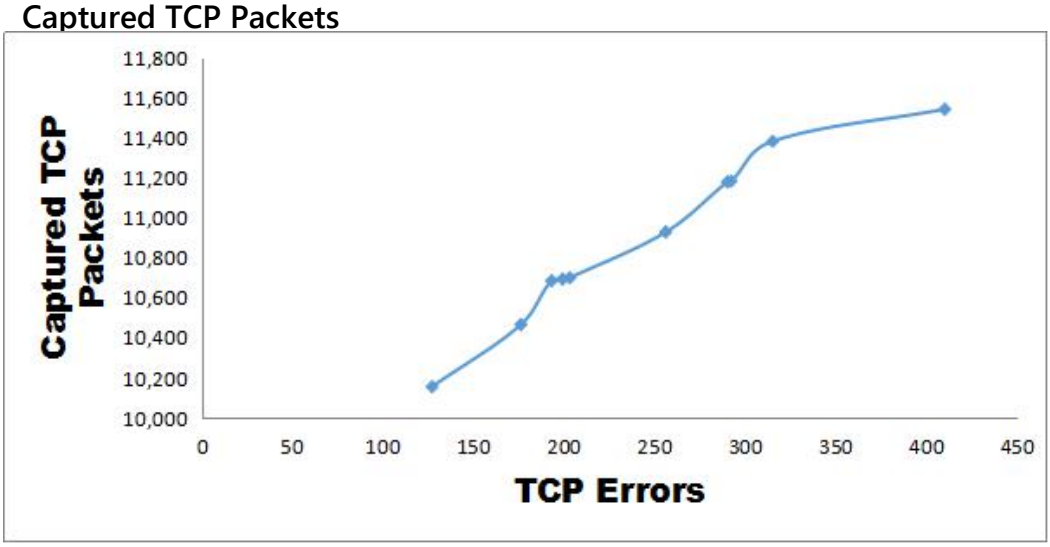

Figure 3: Captured TCP Packets against TCP Errors for CIDC Claranet_2

Corresponding author's e-mail address: wale@abu.edu.ng

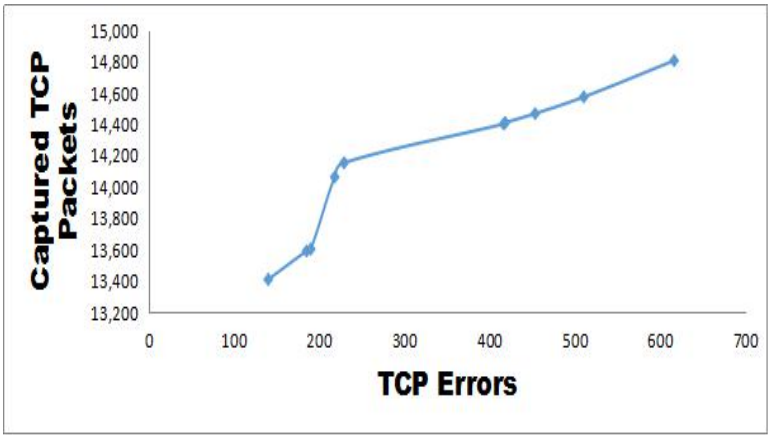

Figure 4: Captured TCP Packets against TCP Errors for CIDC Claranet_3

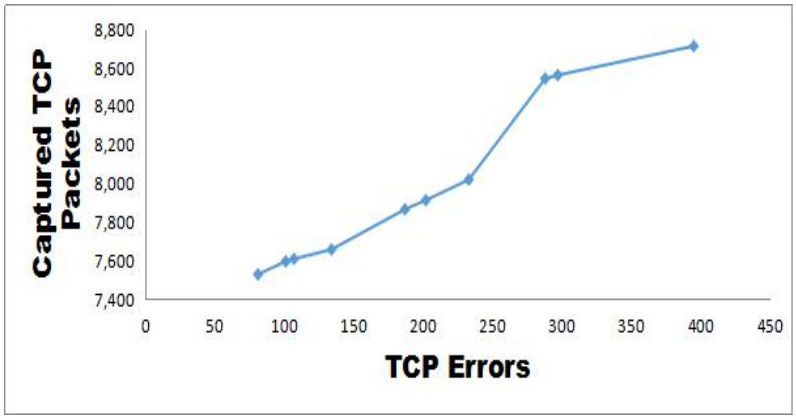

Figure 5: Captured TCP Packets against TCP Errors for mCIDC Claranet_2

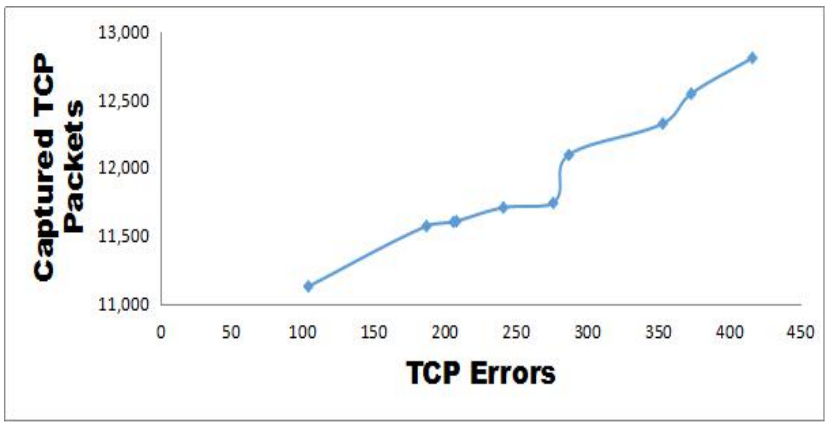

Figure 6: Captured TCP Packets against TCP Errors for mCIDC Claranet_3

It was observed from Figure 3 to 6 that the number of packets increases from Claranet_2 to Claranet_3; showing that the number of packets is related to the number of nodes, the more connected nodes in the topology, the more packets are exchanged. It is also observed that the captured TCP Packets increases as the TCP errors generated increases from Claranet_2 to Claranet_3 in both CIDC and mCIDC. Comparing CIDC (Figure 3 and 4) and mCIDC (Figure 5 and 6); the number of captured TCP packets in mCIDC is less than that of CIDC with a percentage decrease of 26.55% and 15.82% for Claranet_2 and Claranet_3 respectively. This decrease

Corresponding author's e-mail address: wale@abu.edu.ng

indicates that the ISyncService in mCIDC requires less packet exchange for communication.

4.2 Inter-Controller Communication Overload (ICO)

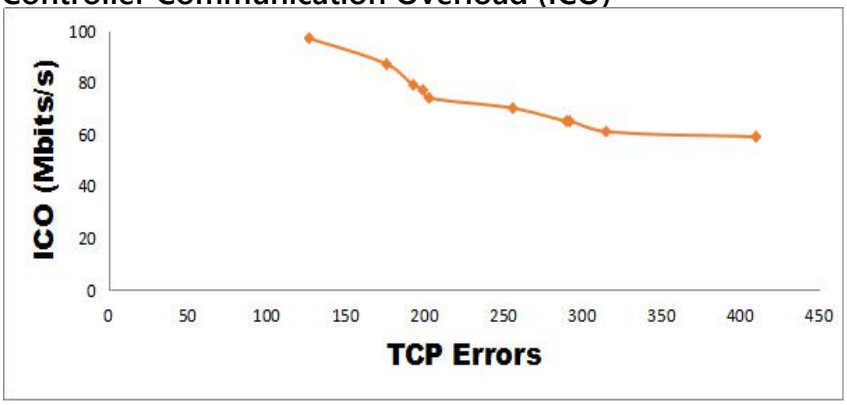

Figure 7: ICO against TCP Errors for CIDC Claranet_2

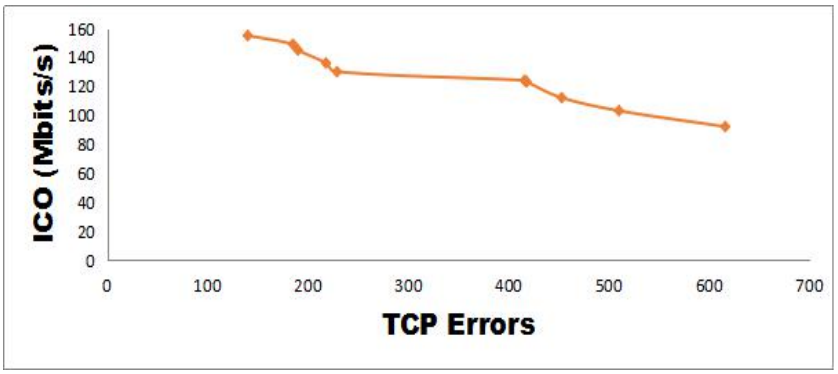

Figure 8: ICO against TCP Errors for CIDC Claranet_3

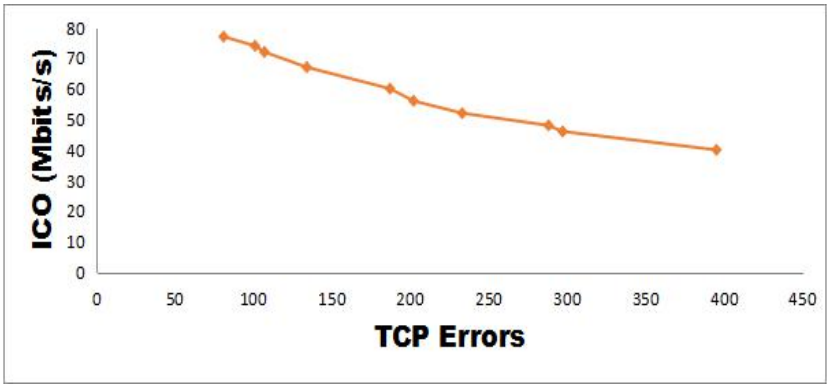

Figure 9: ICO against TCP Errors for mCIDC Claranet_2

Corresponding author's e-mail address: wale@abu.edu.ng

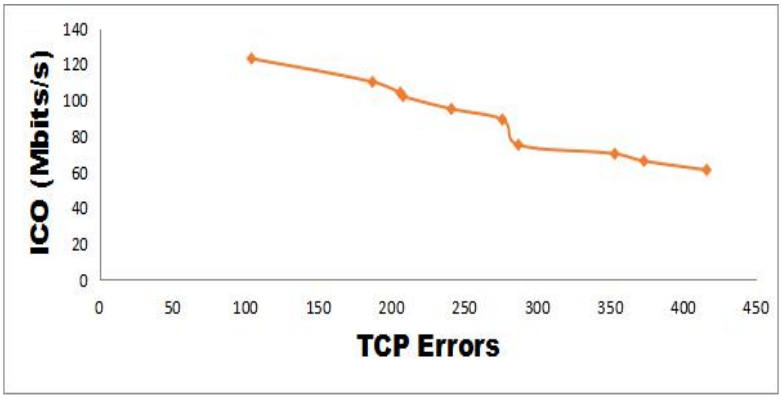

Figure 10: ICO against TCP Errors for mCIDC Claranet_3

4.3 The ICO as shown in Figure 7 to 10 indicates that as the number of TCP errors generated increases, the ICO decreases for both Claranet topologies in CIDC and mCIDC. This decrease is due to the congestion of the TCP Window pipe; making it difficult for exchange of information (traffic) between controllers. The ICO in mCIDC (Figure 9 and 10) is less than that of CIDC (Figure 7 and 8), with a percentage decrease of 19.35% and 29.25% for Claranet_2 and Claranet_3 respectively. mCIDC generates less ICO during multi-domain communication across the WAN.

4.4 TCP Errors

The TCP errors as shown in Figure 3 to 10 shows that as the TCP Errors generated increases; the Captured TCP Packets increases while the ICO decreases for both CIDC and mCIDC. The TCP Errors in mCIDC is less than CIDC with a percentage decrease of 17.89% and 21.60% for Claranet_2 and Claranet_3 respectively. This shows that mCIDC enables sharing of necessary data among controllers with less generated errors; providing efficient network communication.

5. Conclusion

The East-West Interface is very important in large SDN networks or SDN based multi-domain networks with multiple controllers (such as WAN). The main purpose of East –West Interface is to synchronize states for high availability, by enabling communication between groups or federations of controllers; this is achieved through monitoring /notification capabilities and exchange of data between controllers. The mCIDC is a modified East-West Interface developed for WAN, achieving high availability across multiple controllers in the control plane, for efficient communication. The mCIDC improves on the CIDC in the experiments, showing that the mCIDC establish high availability with decrease in captured TCP packets and ICO as compared to CIDC. This will improve communication, policy updating and decision making in multi-domain networks (WAN).

Corresponding author's e-mail address: wale@abu.edu.ng

Acknowledgment

We would like to thank Nigeria Liquified Natural Gas Limited (NLNG) for provision of Computers and a Laboratory to carry out this study.

References

Benamrane, F. and Benaini, R. 2017. An East-West interface for distributed SDN control plane: Implementation and Evaluation. Computers and Electrical Engineering, 57: 162-175.

Blial, O., Ben Mamoun, M. and Benaini, R. 2016. An overview on SDN architectures with multiple controllers. Journal of Computer Networks and Communications, Article ID 9396525: 8.

Chen, J., Zheng, X. and Rong, C. 2015. Survey on software-defined networking. In: Qiang W., Zheng X., Hsu CH. (eds) Cloud Computing and Big Data. CloudCom-Asia 2015. Lecture Notes in Computer Science, Springer.

Cham, N.Y. Jarraya, Y., Madi, T. and Debbabi, M. 2014. A survey and a layered taxonomy of software-defined networking. IEEE communications surveys and Tutorials, 16(4):1955-1980.

Kreutz, D., Ramos, F. and Verissimo, P. 2013. Towards Secure and Dependable Software-Defined Networks. In Proceedings of the second ACM SIGCOMM Workshop on Hot Topics in software defined networking, Hong Kong, China - August 12 - 16, 103(1): 14-76.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S. and Uhlig, S. 2015. Software-defined networking: A comprehensive survey. arXiv preprint arXiv:1406.0440.

Lin, P., Bi, J. and Wang, Y. 2013. East-West Bridge for SDN network peering. In: Su J., Zhao B., Sun Z., Wang X., Wang F., Xu K. (eds) Frontiers in Internet Technologies. Communications in Computer and Information Science, 401: 170 -181. Springer, Berlin.

Liu, S. and Li, B. 2015. On scaling software-defined networking in wide-area networks. Tsinghua Science and Technology, 20(3): 221-232.

Oktian, Y., Lee, S., Lee, H. and Lam, J. 2017. Distributed SDN Controller System: A Survey on Design Choice. Computer Networks, 121: 100 – 111.

Phemius, K., Bouet, M. and Leguay, J. 2014. Disco: Distributed Multi-Domain SDN Controllers. In 2014 Network operations and management symposium (NOMS) IEEE: pp.1–4.

Rao, A., Auti, S., Koul, A. and Sabnis, G. 2016. High availability and load balancing in SDN controllers. International Journal Trend Research and Development, 3 (2): 310-314.

Wibowo, F. and Gregory, M. 2016. Software Defined Networking Properties in Multi-Domain Networks. In 2016 26th International Telecommunication Networks and Applications Conference (ITNAC), New Zealand: pp. 95-100. IEEE

Corresponding author's e-mail address: wale@abu.edu.ng