International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Reducing CPU Utilization & Improving Failover Time in Dual Controller SDN (Software Defined Network) Environment

S. Annie Christila[1], R. Sivakumar[2]
[1]Research Scholar,
Department of Computer Science, CHRIST (Deemed to be University)
Bengaluru, Karnataka 560029, India.
e-mail: annie.s@res.christuniversity.in
[2]Department of Computer Science,
CHRIST (Deemed to be University), Bengaluru, Karnataka 560029, India.
e-mail: sivakumar.r@christuniversity.in

Abstract— A salient component of the network is failover. Largely, customers are accessing dual path controller software defined network (SDN) environment. Subsequently, when one controller fizzles out the network communication can persist as a result of the other controller. There will be no interruption in the network communication due to the failover setup which will be effective in case of failure of a controller. In the absence of failover, the network communication will be obstructed. This could be by cause of the system resources, system itself or routing policies or Distributed Denial of Services. Generally, controller will support many switches. When SDN controller collapses controller will not be in sync with switches. After few attempts switches will failover to the secondary SDN controller. During this shutdown time the network communication will be obstructed. Failover algorithm is the key to solve this issue and will facilitate the synchronization of the information with SDN controllers and enhance failover time. Apparently, communication between both SDN controllers and switches is enhanced as well as reinforces the time of synchronization due to multiple SDN controllers. CPU utilization is another problem in SDN environment as the new inflow packets need to be forwarded to control plane to find the action that need to be taken. Processing at both ends and having one to one mapping between new incoming connection requests and the packet from data plane to control plane increases the CPU utilization and there by more power consumption.

Keywords – SDN (Software Defined Network), NIC (Network Interface Card), VM (Virtual Machine), VEA (Virtual Ethernet adapter), LPAR (Logical PARtition), CC (Cloud Computing)

I INTRODUCTION

When it comes to Software Defined Networks (SDN) cloud computing (CC) and Software Defined Networks (SDN) have made significant impact at a pivotal moment of SDN both in the industry and academia wider range of acceptance in network community. Moreover, this compelling exploration has attracted many researchers to map out SDN-related network security solution [1]. The scope of SDN-related solutions has indeed allured more interest as their adoption in area networks grows far and wide. The technologies allow developers to directly manage, program and control network resources over the SDN controller [2]. In order to attain a more reliable, programmable, stable and flexible network services, the SDN would isolate the control plane from date plane. The new and recent studies show that SDN in conjunction with edge computing can be implemented in the Internet of Things (IoT) like ubiquitous and smart city healthcare [3].

Edge controllers of SDN play a vital function in gathering of the network data from forwarding gadgets to retain an extensive view of local network by way of applying logically centralized management of local data plane. As per the flow table, it is through switches responsible as forwarding gadgets that data in data plane are forwarded. OpenFlow protocol was broadly employed in the control as well as data planes.

On the one hand you have CC that upholds the substantial ideas of attention differentiated with traditional computing methods and on the other hand SDN paves way to security solution [5]. It is for this complementary effect both CC and SDN have been embellished basically in the industry and scholarly community due to their indispensable features [6].

Even though the two innovations were making several novelties that transform both industry and research however security was the crux of these two technologies. Due to their primary function of resolving security problems CC and SDN have drawn so many authors who made an extensive research

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

[7]. Consequently, the research works show that investigation works uncovered various security threats that are directed against CC and SDN over several network elements.

Hence, security was viewed as the main hindrance to the development of CC and SDN. SDN is disposed to Distributed-Denial-of-Service (DDoS) attacks because of its centralized control structure [8]. Due to these malicious packets with spoofing addresses that are sent to switches can simply cause flow table overflow and buffer fullness as switches in the data plane contain limited sources. Moreover, controller saturation occurs due to packet flooding on controller resulted by security issue whereby witches were forces transfer several packets in messages to controller for flow requests [9]. Thus, DDoS assaults result in network collapse, and flow recognition was indispensable for the security of SDN networks. Many approaches were implemented as classifiers for flow detection in SDN [10]. The efficiency of various techniques can affect the efficiency of DDoS defence in SDN.

In order to ensure a smooth and efficient controlling the dual controllers are configured with Cloud SDN (Software Defined Network) so that in case of a failure of one controller the other can take over [16]. No matter where the problem arises from be it system resources or any other routing/policy algorithms users can succeed by configuring dual SDN Controller in their environment. Through the intermediate devices to connect to network one switch will control both SDN controllers. This will allow the switches to reach out to the SDN controllers to facilitate the necessary action as per the flow of data received.

For instance, traffic is detected outside host A, initiated by LPAR this will traffic will be forwarded to the primary SDN controller by the switch connected to primary SDN. Primary SDN controller then will determine the necessary action to be taken according to the type of traffic detected. As a follow up, switch will update its forwarding table and take action. As this is dual SDN controller environment the same information will be updated to the secondary SDN Controller periodically. This will ensure both SDN controllers maintains the information about hosts/policies. Control elements are responsible for more than 1000 data elements. The following diagram is an example how SDN is configured in the data centre.

II. PROBLEM STATEMENT

Dual controllers' system in Cloud SDNs (Software Defined Network) is monumental as they are configured with dual controllers. As a safety measure to protect and ensure guiding/controlling of communication two controllers reinforce each other to prevent blackout. Be it system resource or any other routing/policy algorithms- related problems users can resolve them by configuring dual SDN Controller in their environment [16]. Through separate switches, both SDN controllers will be connected to other intermediate devices to connect to network. After the configuration of the dual SDN Controller user then needs prioritize any one of the controllers which then becomes primary SDN controller while the other one becomes automatically secondary SDN Controller.

All the LPARs will establish communication to outside by reaching out the primary SDN Controller. Whenever there is a detection of traffic both SDN Controllers will be synchronized in such a way that the dual SDN Controllers, primary as well as secondary will determine the necessary action to be take and send the response. This dual SDN controllers' system will help maintain the information about hosts/policies.

At times, one might encounter network connectivity issues due to SDN controllers vulnerable to DDoS attacks wither because of network issues, system resource issues or operating system issues in this case there will over flooding of incoming packets especially new flow which then causes network connectivity issues.

Furthermore, how do we receive large packets in a better way to gain better network performance? Here is another salient feature of multiple SDN controller due to which communication between both SDN controllers is efficient and the time it takes to sync between both controllers is efficient. This is one of the main studies addressed in this research paper. A maximum transmission unit (MTU) MTU is an inherent property of the physical media associated with the network protocol. For example, a maximum transmission unit (MTU) for Ethernet is 1500 bytes. In many situations network applications sends small messages across the network as these packets are received by the receiver it processes each packet and sends to network layer. The role of network layer then is to verify this packet and sends to transport layer which in return checks the validity of this packet (including checksum verification) and wakes up the application that is waiting for the data. In recent days network adapters are used in order to support large receive functionality in which it receives more than one packet for the TCP connection and sends all those packets to upper layer at once. However still the network layer and transport layer need to process those packets before giving it the application. This paper will addressee more efficient way of receiving large packets in effective way.

CPU utilization is another problem in SDN environment as the new inflow packets need to be forwarded to control plane to find the action that need to be taken. For every new inflow

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

of TCP/ICMP/UDP packets the packet need to forwarded from data plane to control plane and once the response is received the appropriate action need to be taken. This consumes more network bandwidth at the same time more processing time in control plane. This method increases the CPU utilization and there by more power consumption.

III. EXISTING METHOD

Dual controllers' system in Cloud SDNs (Software Defined Network) is monumental as they are configured with dual controllers. As a safety measure to protect and ensure guiding/controlling of communication two controllers reinforce each other to prevent blackout. The following challenges are presented:

1. For every inflow of packets there is an equal packet generated and then switch send packet to controller. For example, if 500 new inflow packets comes in, it generates 500 packets.
2. Consumption of the network bandwidth and resources for process all the packets.
3. Encryption and decryption of the data of switch and controller are a costly operation.
4. Vulnerability to DDoS attacks will lead to the following problems:
    - Increased resource usage (memory and CPU)
    - Causing packet buffer overflow and packet drops
    - OFA (Open Flow Agent) dropping the new packets.
    - In some cases, increase in connection establishment time (TCP retransmission timers are in seconds 1.5, 3, 6…)

Let us now take a look at the existing method. When TCP connection gets established LPAR will be sending TCP SYN packet. Once received, the packets will be forwarded to Primary SDN by the switch. The necessary action will be determined by SDN controller for witch to take for the purpose packet inflow. Once the switch gets a response weather to forward or to reject the packet switch then will update the forwarding table and send the pocket or reject it if the command to do so. At times, one might encounter network connectivity issues due to SDN controllers vulnerable to DDoS attacks wither because of network issues, system resource issues or operating system issues in this case there will over flooding of incoming packets especially new flow which then causes network connectivity issues. After few attempts Switch then will finally give up and on contacting primary SDN it will start accepting the response from secondary SDN.

Similarly, the communication between both SDN controllers it sends single packet with length as MTU. In many situations network applications sends small messages across the network. When these packets are received by the receiver it processes each packet and sends to network layer. Network layer verifies this packet and sends to transport layer. Transport layer checks the validity of this packet (including checksum verification) and wakes up the application that is waiting for the data. When SDN controller has big data that need to be sent to other controller the time it will take in TCPIP stack in both sender and receiver is considerably high. The following diagram explains the failover SDN configuration.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

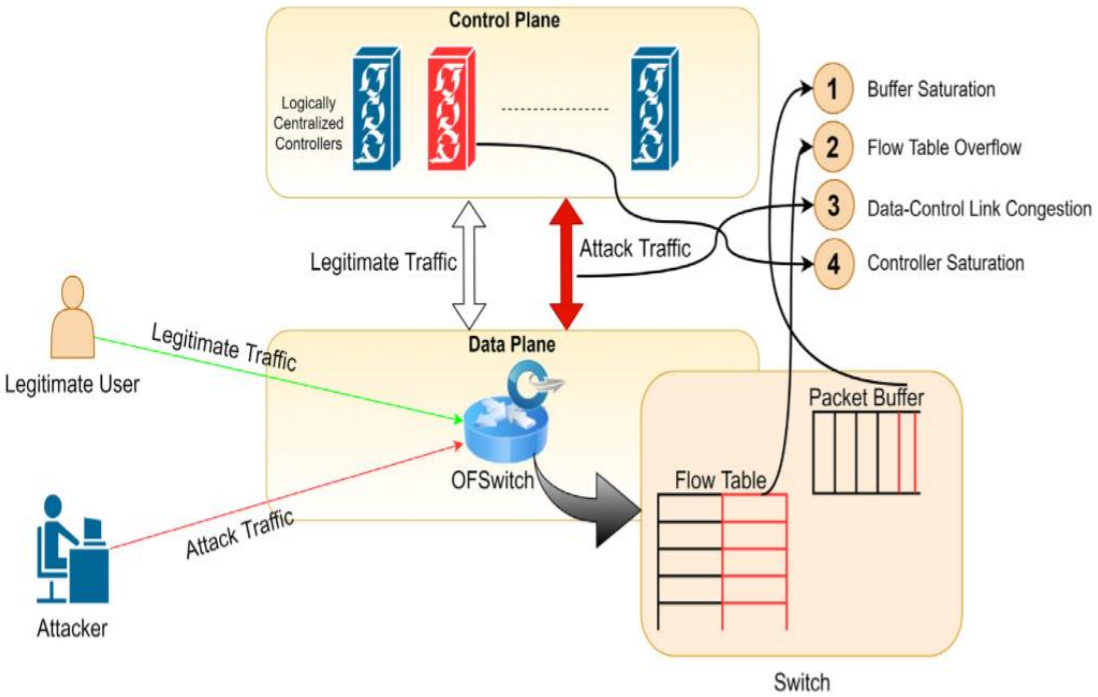


Figure 1. Failover Configuration - Dual SDN Controller setup

IV PROPOSED METHOD

Failover is a procedure by w hich a system automatically transfers control to a duplicate system when it detects a fault or failure. In this paper, an algorithm to achieve SDN controller fast failover is proposed. All the routing/flow tables, policy entries will be stored/calculated on primary controller and synched with secondary SDN controller equally. It is to avoid blackout in case of failure of a controller. To avoid this, EtherChannel will be created in the switches and each switch will be connected to each controller primary as well as secondary. Both the ports will receive new data flow from switches and consequently so do both SDN controllers. Secondary SDN controller will however populate the routing table/policy updates based on the new packet. Switch will turn to secondary SDN controller after three attempts of non-response. Through OpenFlow protocol switch will send communication to secondary controller through OpenFlow and point out the action to be taken. Switch will make few attempts to reach secondary SDN controller and request it to move as primary SDN controller. As we have already synced all the information including routing tables, forwarding tables, policy information and security configuration, the secondary SDN controller will easily take over.

Algorithm

1) There will be an exchange of heartbeats between both SDN Controllers (SDN) during bring up phase will exchange heartbeats.
2) Through separate switches, both SDN controllers will be connected to other intermediate devices to connect to network. After the configuration of the dual SDN Controller user then needs prioritize any one of the controllers which then becomes primary SDN controller while the other one becomes automatically secondary SDN Controller. Decisions and sending it to data plane will be done by Primary Controller.
3) EtherChannel will be created in the switches and each switch will be connected to each controller primary as well as secondary. Both the ports will receive new data flow from switches and consequently so do both SDN controllers. Secondary SDN controller will however populate the routing table/policy updates based on the new packet.
4) There will be a periodical exchange of all the information routing table / policies and security parameters between both the SDN controllers as well as the forwarding table & configuration settings information (Source MAC, Source IP, Destination MAC, Destination IP).
5) Maintain the exchange of heartbeat information between both SDN Controllers

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

6) Secondary SDN Controller will take effect and learn about primary SDN controller's start building the routing table/security policy updates.

7) When there is no response from primary controller, switch will attempt 3 more times before turning to secondary SDN controller.
8) Switch will make few attempts to reach secondary SDN controller and request it to move as primary SDN controller. As we have already synced all the information including routing tables, forwarding tables, policy information and security configuration, the secondary SDN controller will easily take over.
9) As we have already synced all the information including routing tables, forwarding tables, policy information and security configuration, the secondary SDN controller will easily take over.

Moreover, through this paper a faster communication between switch and SDN controllers as well improvement in communication speed which results less consumption of electric power are equally achieved. This method also proves less CPU consumption, better network by less energy consumption. The first one coalesces the packet from switch and sends as single packet the controller. This helps easy processing and also saves lot of CPU cycles.

A. Adapter input interrupt path

1. In Open Flow Switch, on receipt of each packet flow table is checked to take specific action.

2. If there is matching entry present in the flow table the specific action (forward/deny) is taken.

3. If there is no matching entry, it is new flow. Packet header information will be added to the specific common memory. For example If packet is received on the port 4, packet header information will be written in the block allocated for port 4.

4. Set the bit in meta data field. (If memory allocated for port 4 is updated, then 4^{th} bit in the meta data field will be set to 1)

5. If packet processing timer is not running start the packet processing timer (10ms).

B. Packet Processing Timer Routine

7. Check the meta data (first 4 bytes of the packet). If it is 0 no new inflow detected in this interval. Go to step 10. If meta data is not 0 then some new flow arrived in this interval.

8. Form the packet using the specific common memory and send it to controller. (Adapter interrupt path has filled packet header information for every new flow in this memory)

9. Initialize the specific common memory to 0.

10. Start the packet processing timer (10ms) again. Repeat steps 7,8 & 9 upon the timer expiry.

C. Processing on Controller

1. On controller side, based on the meta data it will process the blocks. If bits 1 and 3 are set in meta data only those fields

2. corresponding to port 0 and port 2 are processed. Reply will be sent back to data plane

With this proposed algorithm,

- 32 Requests packet can be sent together. This will reduce the traffic on the control channel.
- Controller can process these requests in parallel and can send cumulative response.
- The same algorithm can even be modified to accommodate 80 requests together there by reducing further traffic on control channel.

Most importantly, the requests can be processed in parallel and on receiving the cumulative response we can free up upto 80 buffers used for storing the packets.

The second approach utilizes the large receive functionality and there by processing the incoming requests much quicker. A network device can have configurable socket option to enable large receive from the network adapters.

Meta Data	Packet Header info from port 0	Packet Header info from port 1	..	..	..	..	Packet Header info from port 6
Packet Header info from port 7	..	..	..	..	..	..	Packet Header info from port 14
Packet Header info from port 15	..	..	..	..	..	..	Packet Header info from port 22
Packet Header info from port 23	..	..	..	..	..	..	Packet Header info from port 31

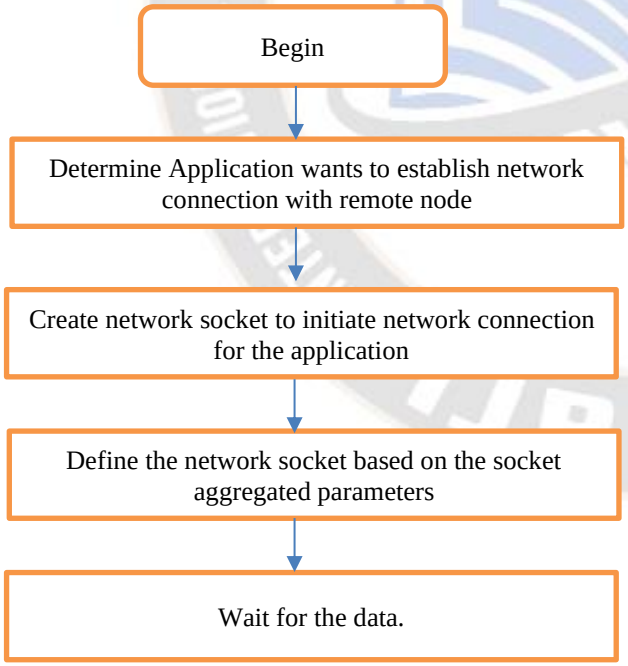
Figure 2. Coalesced Packet format

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

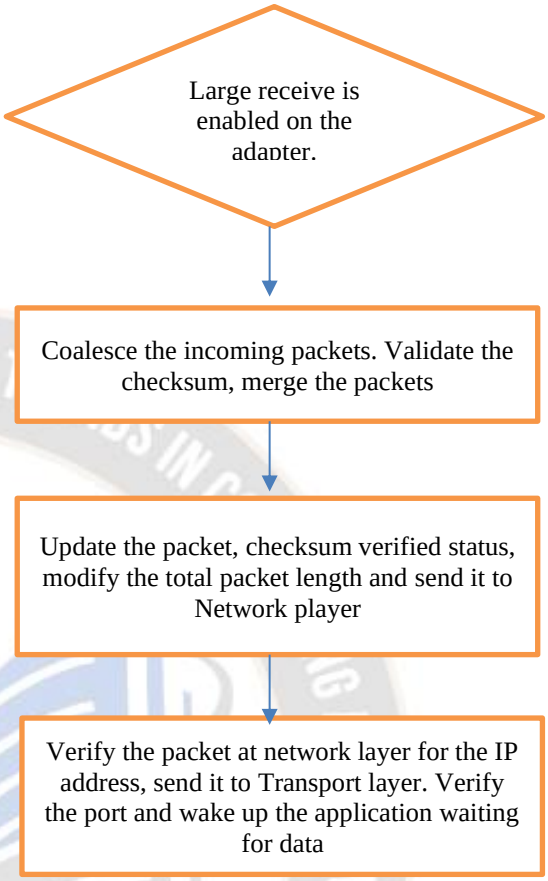
Applications create network socket to have communication across application on different node. Socket option parameters are determined for the network socket associated with the application. In case of large receive adapters receive many packets, aggregate them and send it to upper layers. It is found that, based on said aggregating the plurality of packets, whether at least one aggregation limit associated with the socket aggregation parameters is reached. An aggregate packet is provided to a network/transport layers to perform basic verification and send it to application.

In many situations network applications sends small messages across the network as these packets are received by the receiver it processes each packet and sends to network layer. The role of network layer then is to verify this packet and sends to transport layer which in return checks the validity of this packet (including checksum verification) and wakes up the application that is waiting for the data. In recent days network adapters are used in order to support large receive functionality in which it receives more than one packet for the TCP connection and sends all those packets to upper layer at once. However still the network layer and transport layer need to process those packets before giving it the application. This paper will addressee more efficient way of receiving large packets in effective way.

Application Flow



Data flow from network adapter



For example, large receive provides significant performance benefits in systems with Gigabit Ethernet NICs. It has ability to receive huge data and coalescing before sending it to network adapter. When this is enabled and if network layer and transport layer process each and every packet before passing to application it consumes more CPU utilization at network layer and transport layer. Even application gets one packet by one. As a result we are not able gain the performance we got from large receive functionality of NIC.

The above flow chart explain the data flow from network interface card to application. As large receive functionality of the adapter coalesce the packets, if we enable the feature we propose through this paper, checksum of all the packets are verified and packets to every tcp connection is merged. Once the time is over the packet is sent to network adapter. Network adapter checks the IP details and pass it to transport layer. Transport layer just verifies the port and wakes the application that waits for the data. Using this approach CPU time in network layer and transport layer is saved. Thus getting better network throughput and energy saving.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

V RESULTS

During this section, the experimental outcome of the improved failover time & Reducing CPU utilization in Dual SDN environment approach is tested by simulating high volume of new connection requests and ping requests. It is SDN detailed database produced by employing mininet emulator. The existing starts with creating 10 topologies in mininet but switches are related to single Ryu controller. The network simulation runs intended for benign ICMP and TCP traffic and malicious traffic that has been collected in TCP Syn attack and ICMP attack. The database holds 10000 instances with 2 classes as depicted in Table 1. We have also generated different data set with samples 50000, 100000 and 200000 and tested.

TABLE I Details on Dataset

Class	No. of Samples
TCP Connection	9000
ICMP Requests	1000
Number of Samples	**10000**

We have tested using Mininet and generated new connection flow. The following figure shows the new inflow packets from SDN data plane to controller plane for the incoming new TCP connections. It is evident that for every new connection there is one new inflow packet gets generated and sent to controller. Once response received from controller the appropriate action is taken in the data plane.

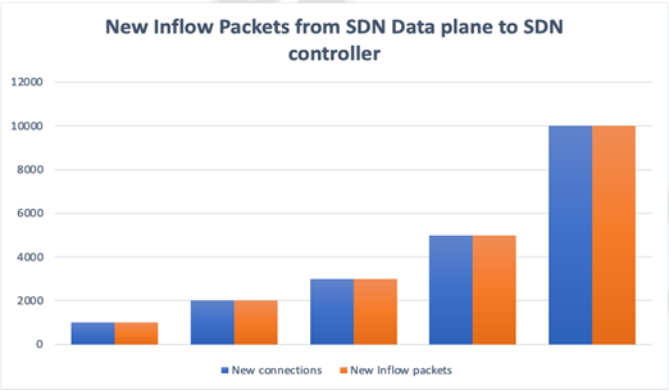


Figure 3. Inflow packets from SDN data plane

The following figure shows the inflow packet that gets generated to control plane reduced by 90%

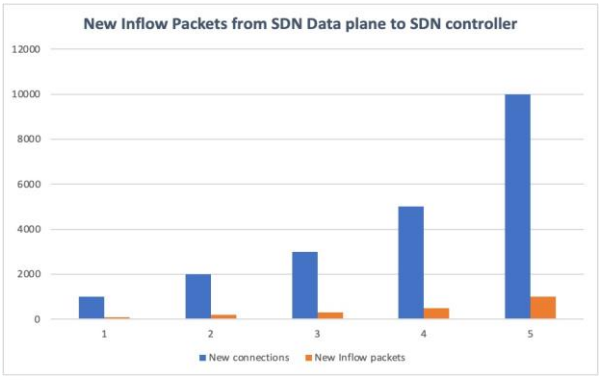


Figure 4. Inflow packets from SDN data plane using new algorithm

Now we look at how this helps us in improving the performance. We will be measuring the delay in establishing connections. For the exercise we have generated thousands of new connections and measured how long it takes on average to establish the connection.

CPU utilization is measured using the existing method and new algorithm. Test is performed with various different data sets. First 10000 new connection is simulated and the CPU utilization is measured. In the next test 20000 new connection is simulated and the CPU utilization is measured. The same test is performed with 50000, 100000 and 200000 new connections. When less number of connections exists CPU utilization is very low in the existing algorithm. However when number of connection increases the CPU utilization is also increasing. In case of new algorithm initially CPU utilization is high that is due to additional overhead in waiting/processing multiple packets together. However when more number of connection exists the new algorithm need not process so many header information. The following diagram explains how CPU utilization is reducing when more number of new networking connection established.

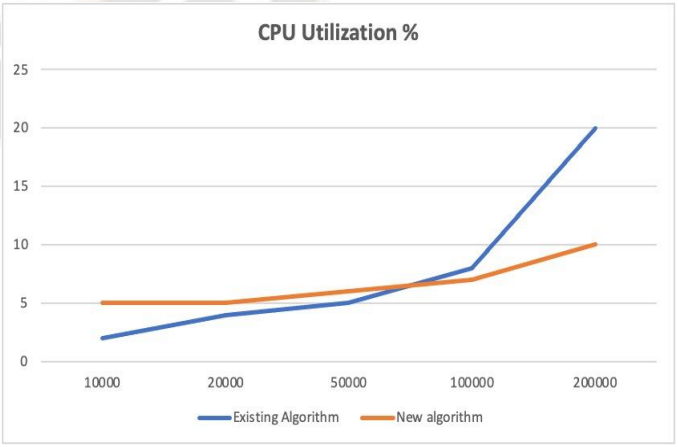


Figure 5. CPU Utilization

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

Time taken to establish the connection is measured using the existing method and new algorithm. Test is performed with various different data sets. First 10000 new connection is simulated and the time taken to establish the connection is measured. In the next test 20000 new connection is simulated and the time taken to establish the connection is measured. The same test is performed with 50000, 100000 and 200000 new connections. When less number of connections exists time taken to establish the connection is very low in the existing algorithm. However when number of connection increases the time taken to establish the connection is also increasing. In case of new algorithm initially time taken to establish the connection is high that is due to wait time at data plane before sending the connection requests to control plane and also on additional overhead in waiting/processing multiple packets together. However when more number of connection exists the new algorithm need not process so many header information. The following diagram explains how time taken to establish the connection is reducing when more number of new networking connection established.

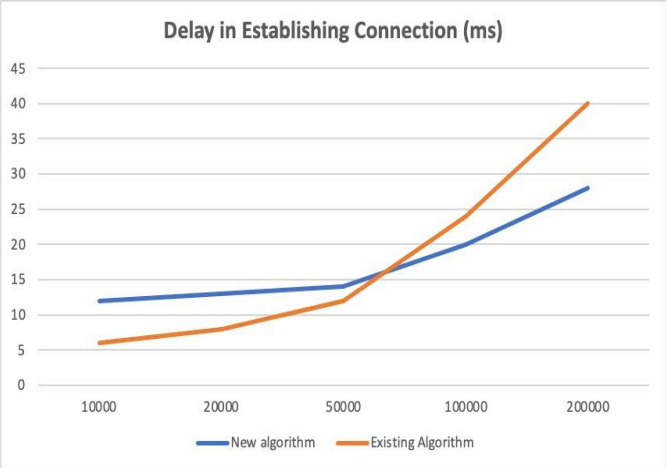


Figure 6. Delay in establishing connection.

VI CONCLUSION

The main purpose of this proposed method is to enhance the failover time between primary SDN controller and the secondary SDN controller. an algorithm to achieve SDN controller fast failover is proposed. All the routing/flow tables, policy entries will be stored/calculated on primary controller and synched with secondary SDN controller equally. It is to avoid blackout in case of failure of a controller. To avoid this, EtherChannel will be created in the switches and each switch will be connected to each controller primary as well as secondary. Both the ports will receive new data flow from switches and consequently so do both SDN controllers. Secondary SDN controller will however populate the routing table/policy updates based on the new packet. Switch will turn to secondary SDN controller after three attempts of non-response. Through OpenFlow protocol switch will send communication to secondary controller through OpenFlow and point out the action to be taken. Switch will make few attempts to reach secondary SDN controller and request it to move as primary SDN controller. As we have already synced all the information including routing tables, forwarding tables, policy information and security configuration, the secondary SDN controller will easily take over.

In the other two improvement ideas, implementing configurable socket aggregation option to enable large receive in a network device are described herein. It is determined, at the network device, to create a network socket for an application of the network device to initiate a network connection for the application. Socket aggregation parameters are determined for the network socket associated with the application. The network socket is defined based, at least in part, on the socket aggregation parameters. A plurality of packets received from the network adapter for the TCP connections are aggregated. It is determined, based on said aggregating the plurality of packets received from the network adapters, whether at least one aggregation limit associated with the socket aggregation parameters is reached. An aggregate packet is provided to a network layer where it verifies the IP address and checksum which intern provides to transport layer. After port and checksum verification the packet is sent to application. As almost 40 packets are getting coalesced into single packet, lot of CPU time is saved both in network and transport layer. This results in better network throughput and less CPU consumption. Moreover, through this paper a faster communication between switch and SDN controllers as well improvement in communication speed which results less consumption of electric power are equally achieved. This method also proves less CPU consumption, better network by less energy consumption. The first one coalesces the packet from switch and sends as single packet the controller. This helps easy processing and also saves lot of CPU cycles.

REFERENCES

[1]. S. Kautish, R. A and A. Vidyarthi, "SDMTA: Attack Detection and Mitigation Mechanism for DDoS Vulnerabilities in Hybrid Cloud Environment," in *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6455-6463, Sept. 2022, doi: 10.1109/TII.2022.3146290.

[2]. K. Srinivasan, A. Mubarakali, A. S. Alqahtani, and A. Dinesh Kumar, "A survey on the impact of ddos attacks in cloud computing: Prevention, Detection and mitigation techniques," *Intelligent Communication Technologies and Virtual Mobile Networks*, pp. 252–270, 2019.

[3]. M. Revathi, V. V. Ramalingam, and B. Amutha, "A machine learning based detection and mitigation of the DDOS attack by using SDN Controller Framework," *Wireless Personal Communications*, 2021, pp.1-25.

[4]. K. Bhushan and B. B. Gupta, "Distributed denial of service (ddos) attack mitigation in Software Defined Network (sdn)-based cloud computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 1985–1997, 2018.

[5]. A. Maheshwari, B. Mehraj, M. S. Khan, and M. S. Idrisi, "An optimized weighted voting based ensemble model for DDoS attack detection and mitigation in SDN environment," *Microprocessors and Microsystems*, Vol. 89, p.104412, 2022.

[6]. I. A. Valdovinos, J. A. Pérez-Díaz, K.-K. R. Choo, and J. F. Botero, "Emerging ddos attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and Future Directions," *Journal of Network and Computer Applications*, vol. 187, p. 103093, 2021.

[7]. K. Bhushan and B. B. Gupta, "Detecting DDoS Attack using Software Defined Network (SDN) in Cloud Computing Environment," 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN), 2018, pp. 872-877, doi: 10.1109/SPIN.2018.8474062.

[8]. D. Yin, L. Zhang and K. Yang, "A DDoS Attack Detection and Mitigation With Software-Defined Internet of Things Framework," in IEEE Access, vol. 6, pp. 24694-24705, 2018, doi: 10.1109/ACCESS.2018.2831284.

[9]. O. E. Tayfour, and M. N. Marsono, "Collaborative detection and mitigation of distributed denial-of-service attacks on software-defined network," *Mobile Networks and Applications*, vol. 25, no. 4, pp.1338-1347, 2020.

[10]. S. Batool, F. Zeeshan Khan, S. Qaiser Ali Shah, M. Ahmed, R. Alroobaea, A. M. Baqasah, I. Ali, and M. Ahsan Raza, "Lightweight statistical approach towards TCP syn flood ddos attack detection and mitigation in SDN environment," *Security and Communication Networks*, vol. 2022, pp. 1–14, 2022.

[11]. Richard Stevens; "TCP/IP Illustrated Volume 1" 2nd Edition, Prentice Hall,1999

[12]. William Stallings, Cryptography and network security: Principles and practice, Prentice Hall, Upper Saddle River, New Jersey, 2003

[13]. Network adapters and the features supported – By Cavium/QLogic

[14]. Network Performance and Security by Chris Chapman

[15]. RFC 903 RARP Protocol - https://tools.ietf.org/html/rfc903

[16]. S. Annie Christila, Improving Failover time in Dual VIOS Network Virtualized Environment

[17]. Hai Lin, Lucio Correia, Mel Cordero, Rodrigo Xavier, Scott Vetter, and Vamshikrishna Thatikonda - IBM PowerVM Virtualization

[18]. Gary R. Wright (Author), W. Richard Stevens - TCP/IP Illustrated, Vol. 2: The Implementation

[19]. Kumar Reddy, "Network Virtualization".P

[20]. Network Adapter Specification LSO feature – Intel