An Efficient Rerouting Approach in Software Defined Networks

Md Israfil Biswas
School of Computing
Ulster University
Northern Ireland, UK
mi.biswas@ulster.ac.uk

Sally McClean
School of Computing
Ulster University
Northern Ireland, UK
si.mcclean@ulster.ac.uk

Philip Morrow
School of Computing
Ulster University
Northern Ireland, UK
pj.morrow@ulster.ac.uk

Bryan Scotney
School of Computing
Ulster University
Northern Ireland, UK
bw.scotney@ulster.ac.uk

Gerard Parr
School of Computing Sciences
University of East Anglia
Norwich, UK
g.parr@uea.ac.uk

***Abstract*— This paper illustrates an efficient traffic rerouting solution in Software-Defined Networks (SDN) by monitoring the network status periodically. The proposed approach provides a rerouting solution by first calculating the link utilization for available paths and then rerouting the flow to the least delay path among the available paths. The traffic rerouting solution is considering the network condition to prevent the switch overutilization and congestion while any new flow arrives. The proposed method is implemented by using ONOS controller and Mininet emulator. The proposed algorithm in the controller predicts the utilization and delay on the link to calculate how much load to be rerouted if the average link utilization exceeds the threshold level. Hence, this method will proactively avoid congestion by adding flows, monitoring the parameters and prevent the unbalanced distribution after rerouting as our experimental results show.**

Keywords*— SDN, Network Monitoring, OpenFlow, congestion control)

I. INTRODUCTION

Datacentre networking with the cloud is becoming an apprehension of top priority because of the high availability and security of computer networks. Hence, the prodigious trust in the services of computer networks is essential for both users and service providers. However, the Internet has dynamically changed over the last decades and despite such a tremendous success, the increasing dependability on the Internet has created concerns for future.

Software-Defined Networks (SDN) have become the new paradigm with the most promising and popular advantages of the Internet; like global visibility, openness, vendor independence and programmable networking devices incorporating Network Function Virtualization (NFV) [1]. SDN with OpenFlow [2] protocol leads to various innovative traffic engineering techniques with its intelligent controller and a programmable data plane that are integrally flexible and customizable. Virtualization capabilities over geographically isolated and transparent resource provisioning [3] also shows these advantages.

SDN has a faster deployment with programmable networking elements in Control and Forwarding Plane. The control deals with the necessary protocols like OSPF, BGP and manages topology to exchange information so that the data plane can forward packets, resulting in end-to-end connectivity. Hence, the data plane is primarily focused on the switching between the data paths of the routing architecture that decides what to do when a packet is received on its inbound interface. Fig.1 shows the routing planes operations in SDN environment.

Efficient routing algorithms are essential for traffic monitoring, especially during live Virtual machine (VM) migration [4] to find the shortest paths in a network and lead the traffic through the route. SDN helps to optimize the traffic by analysing, predicting and regulating the transmitted data. However, among many problems in the current routing system is forwarding of packets along non-optimal routes by over-utilizing some links while leaving other links idle. Reactive forwarding by the Open Network Operating System (ONOS) controller is the default packet forwarding mechanism to forward packets whenever a new flow arrives at the switch [5]. This reactive forwarding method sends a copy of the first packet header from a new flow to the controller and then the controller installs a forwarding rule to the switch whenever the new flow arrives there.

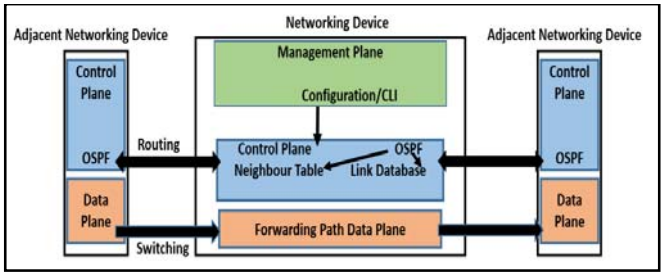

Fig. 1. Routing Planes Operation in SDN

978-1-5386-7275-4/18/$31.00 ©2018 IEEE

Hence, without awareness of traffic condition and Quality of Service (QoS) parameters, making a route decision can lead to inefficiency and is a major drawback of the reactive forwarding method, which could result in throughput degradation and increased flow completion time.

A. Related Works

This section briefly reviews some basic notions related to traffic engineering techniques and discuss their aspects with limitations.

A forwarding algorithm is proposed in [6] that separates the elephant flows into mice and distributes them across multiple paths. The routing algorithm is based on label-based forwarding in a round-robin manner. However, this type of algorithm requires overhead bytes. Hence, the packet header increases linearly with a path to implement the policy. Our current method is based on path utilization and estimated delays of each path instead of round-robin to split traffic load.

A modified version of Penalizing Exponential Flow-splitting (PEFT) is proposed in [7] to handle dynamic DCN traffic. The optimal routing paths are calculated considering the incoming and outgoing traffic volume across all the associated ports of each switch. This method results in packet reordering problem during the splits and transfers of the flow through unequal cost path. The local traffic parameters are collected during runtime at switches that help to make packet forwarding decisions on hop-counts. Hence, the delay produces heavy load on switches.

Automatic Re-routing with Loss Detection (ARLD) [8] is a method enabled by SDN and the OpenFlow protocol while packet drops occur in a congested network. Once packet loss is detected, the node itself is removed from the topology. This makes all other paths through this node unavailable during alternate route computation. The controller reroutes to another link during any packet loss at a particular link declared as a bottleneck link. Hence, the controller updates switch flow tables with the reroute information for further traffic to reroute through the updated route until the flow table entry expires. Thus, this is a reactive method initiating rerouting and alternative route computation once network congestion and packet drop happen. The problem with such method is that it does not consider the existing network utilization and may choose all the available paths for the reroute. Therefore, the rerouting can again introduce congestion, if the new alternate path is already a fully utilized link and degrades the whole network performance or QoS. The main limitation of such a reactive method is, as the network is already congested, a good number of packets already gets dropped by the time controller which computes and applies alternative route to flow tables.

In a proactive approach like our proposed method involves computation of primary paths and avoid packet drops resulting in additional resource reservations and flow table entries which is an overhead to the SDN controller.

S. Song et al. [9] proposed a proactive method by changing the topology predicting congestion in advance and monitoring the network status periodically. The algorithm is designed in such a way that the SDN controller will reroute to a new path bypassing a switch that is over-utilized by crossing the threshold. The controller recovers to the original route while the utilization of congested switch is decreased to 50%. However, the major limitation of such an approach is that it may reroute the current flows to a long delay path and can lead to worsened congestion condition.

Our goal in this paper is to provide an efficient rerouting mechanism, we are interested in applications that provide accurate, real-time information about the path utilization. However, when it falls below an established threshold, addresses the SDN controller to re-route network traffic to a backup route with least delay. Our results show that the developed open source application can effectively perform these tasks in cooperation with an ONOS controller.

In summary, this paper is unique in the following aspects:

- The proposed rerouting approach in SDN environment reduces Flow Completion Time (FCT) for large flows by sending flows over less utilized and least delay paths.
- The path reconfiguration experiments show the benefit of the proposed method as more frequent path reconfigurations can increase the number of packet drops which leads to retransmissions and finally degrade the application performance. The proposed approach can reduce the number of packet drops for large flows and improve the network performance.
- Experiment results show that the proactive approach to transport solution with SDN maintains the total number of flows as low as possible hence, avoid congestion and makes the algorithm fast with reduced overhead.
- The bandwidth utilization experiment shows that the proposed proactive approach can have better utilization of the associated switches.

The rest of the paper is organized as follows: Section II describes Packet Rerouting approaches in SDN. Section III presents the proposed transport rerouting solution for SDN. Section IV describes the evaluation of the proposed method with the experiment setup and results. Finally, section V provides some conclusions and a view for future work.

II. Packet Rerouting in SDN

Internet Service Providers (ISPs) and Datacentres (DCs) are facing challenges with the rapid growth of the Internet that require dynamic topology adjustment (rerouting to backup paths) to ensure stable and secure connectivity. However, this may cause an expensive system with a number of demerits. Therefore, SDN is used for the solution of effective dynamic rerouting.

In SDN, the network control function is separated from the forwarding elements and helps the control function into an individual centralized control. One of the important control function is routing control that contains the knowledge of switches, routing information and network status.

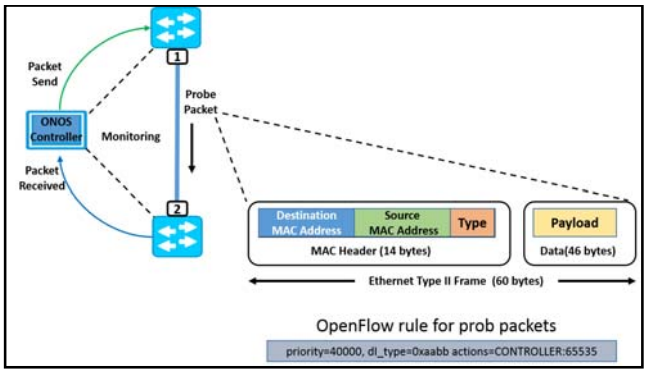

Fig. 2. Illustration of OpenFlow link Monitoring

SDN manages routing algorithms separately while keeping data flows running on original network paths with the centralized controller. The simple and flexible routing control with the capability of adaption during the change in networking state make SDN tremendously popular.

SDN uses the OpenFlow Protocol that sets rules for routing by installing a small piece of OpenFlow firmware in a switch and giving access to flow tables. In a flow table, each flow entry is associated with an action, to instruct the switch how to process traffic from the flow. A Secure Channel that connects the switch to a remote controller, permitting commands and packets for communication between a controller and the switch using the OpenFlow protocol. The OpenFlow Switch provides an easier and more convenient way for researchers to test new proposals by specifying the OpenFlow Protocol through which entries in the flow table can be defined externally. Many switch vendors have therefore started to support OpenFlow for these features.

SDN with the centralized management control of a network is responsible for building, displaying the network topology with the capability of network programming on devices like the switch. ONOS with its potentiality of dynamic rerouting to support customized infrastructure effectively manages the entire network with stability. Hence, ONOS makes it a controller choice for many services as it focuses on performance aspects and clustering to increase the availability and scalability.

ONOS is popular for its default applications that directly interact with the controller with a certain level of network abstraction and can serve as tools for network monitoring, control, and analytics[10]. Usually, ONOS is run in a Virtual machine (VM) with developing applications that provide accurate real-time information about connection quality (required for the link monitor), and, in the case where it falls below an established threshold, address the SDN controller to re-route network traffic to a backup route. Hence rerouting is connected to network monitoring, which in turn is directly linked to the ONOS controller.

As shown in Fig.2, the monitoring part is periodically generating test L2 packets and sends them to a switch indicating send to a specific port. The packets are transmitted to the specified port and received by the neighbouring switch. On the other hand, the received probe packets are then returned to the controller. Hence, the quality of the link is measured by the ratio of received and returned packets.

III. The Proposed Rerouting Approach

The purpose of this section is to describe the proposed rerouting approach in SDN. To achieve this goal, the architecture is designed considering path utilization calculation and finally forwards the packet to the least delay paths that help to avoid congestion. Hence, the proposed Utilization Delay Aware (UDA) forwarding algorithm first computes all possible paths from source to destination sorted by hop-count of each path as shown in **Algorithm 1**. For each path, the number of links associated with its utilization is computed to represent the cost of each link in the path. Here, the link cost is calculated for each path to understand how much load can be accepted without that path becoming congested.

In this approach, we have considered 80% of the link utilization as a threshold to reroute the path to calculate the link cost and then calculated least delay path to avoid the congestion. For end-to-end delay calculation of each less utilized path from the pathlist is measured by sending out probe packets from the controller. As shown in the algorithm, we get total delay *TotalDelay* for each path and the average delay *AvgDelay* is calculated. After comparing the average end-to-end delays of available paths, the flow is shifted to the least delay path to avoid congestion and reduce flow completion time.

Algorithm 1: Packet Forwarding through UDA

```
1: TotalDelay =0
2: LeastDelayPath=Max
3: if the LinkLoad exceed the threshold then
4:    RerouteLoad← LinkLoad - (0.8 * bandwidth)
5:    Find LessUtilized paths in the path-list
6:    for j ∈ 1, 2 … RerouteLoad do
7:        rerouteFlow← RequestedFlows [j]
8:        flowRate ← rerouteFlow
9:        if flowRate >= RerouteLoad then
10:                       grantedPath ← LessUtilizedPath
11:               if grantedPath != NULL then
12:                       rerouteFlow to grantedPath
13:                       update the granted load of grantedPath
14:                       RerouteLoad ← 0
15:                       break
16:               end if
17:       end if
18:       Find available shortest paths in the path-list
19:       for each grantedPath in path-list do
20:               for i ∈ 1, 2 … No_of_Flows (N) do
21:                       Calculate Delay for p
22:                       TotalDelay = TotalDelay +Delay [i]
23:               end for
24:       AvgDelay= (TotalDelay/ Number of Flows);
25:       LeastDelayPath=min(AvgDelay, LeastDelayPath);
26:       end for
27:       Update the granted load to LeastDelayPath
28:   end for
29: else
30: route the flow as usual (no rerouting)
31: end if
```

The SDN controller calculates the link utilization and delays periodically by sending a message to a switch. Timestamps are used to check the time from the very beginning when the

switch responds with the total bytes passed through its ports. Hence, the controller stores and uses this information to calculate port utilization and delay and thus helps to predict link overutilization and congestion. Our approach is able to communicate with connected switches and query their flow tables and fetch their flow entries. This information contains flow statistics and flows identifier. When any of switches become over-utilized, this method queries its flow information and sorts them by their periodic utilization calculation. When a flow is rerouted to the selected path, the acceptable load of that path is updated. Paths computed by this calculation is used in forwarding the packets.

Using the flow statistics information, the method can predict congestion. If any port utilization is more than a threshold, for example, more than 80% the controller predicts congestion. Using the UDA forwarding, the controller must be able to shift the load *RerouteLoad* to another path. Where *RerouteLoad* is the total load which must be rerouted from the congested link to eliminate the congestion.

The method calculates the average transmission rate of the out port. The controller iteratively reroutes the flows from the active path to the proper backup path. To do this in each iteration a flow is chosen and rerouted to the best backup path capable of accepting that flow's load. This helps selection of the flows to reroute while maintaining the total number of flows as low as possible to make the algorithm fast and reduce the overhead. This also helps on the ports of over-utilized, by decreasing the flow and reducing the congestion. If the *RerouteLoad* value is zero, all of the flows in the congested switch are checked. Hence, this method will help to skip the paths which are already over-utilized and preventing acceptance of new load. It also checks the path utilization and delays when new flow arrives and uses the shortest path for that flow. If the utilization is near the threshold, the path is skipped, and flow is rerouted to a less utilized path considering the next shortest path. Therefore, this method uses network information during the periodic updates.

IV. Experiment Results and Analysis

We evaluated the proposed congestion aware mechanism in a Leaf-Spine topology (as depicted in Fig. 3) using Mininet emulator [11] and compared with reactive forwarding method in ONOS. The benefits of a congestion-aware algorithm, using metrics such as Completion time, packet drops, can be verified by comparing with the legacy network.

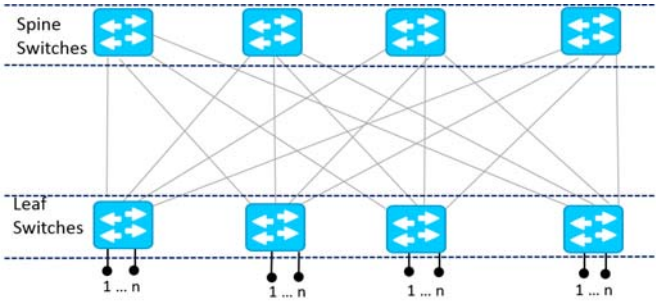

Fig. 3. Illustration of a leaf-spine topology

A. Experiemnt Setup

The first set of experiments validates the proposed routing mechanism compared to reactive forwarding in the leaf-spine topology, where both short flows and long flows are used to find the benefits. In the second set of experiments, we have evaluated the performance of the proposed rerouting algorithm for link utilization using long flows in a use case scenario.

The evaluation is done in an Intel(R) Core(TM) i7-4770CPU 3.40 GHz CPU with Memory of 16 GB machine. The network topology is created using Mininet version 2.2.1 and open flow version 13. Because of performance, high-level abstractions and API, we have used the ONOS controller, version 1.8 for the experiments. The iperf tool [13] is also used to generate traffic and VMs with Wireshark traffic analyser to generate graphs. All the experiments are done over 1000 runs with 95% confidence intervals. The VMs are created considering iperf TCP packets.

B. Configuration details and Results

The server-client model is used in this experiment, considering two Hosts and two sets of flows are considered depending on the VM sizes and link speed. The short flows are with a threshold values less than link speed of 10 Mbps with a packet sampling rate of 1 in 10 packets.

TABLE I. Configuration parameters: Short and Long Flow FCT Experiment

Parameters	Value
Leaf switches	4
Spine switches	4
Host to Leaf switches Bandwidth	1Gbps
Leaf to Spine switches Bandwidth	1Gbps
Window Size (Short Flow)	200 kB
Window Size (Long Flow)	2 MB
VM Sizes (Short Flow)	2 to 10 MB
VM Sizes (Long Flow)	200 to 1000 MB
Traffic	TCP

The other type of flows is long flows with a threshold value of 100Mbps with a packet sampling rate of 1 in 100 packets. Table 1 shows the configuration details for the Leaf-Spine topology. In the first experiment of this set, we compared the Average Flow Completion Time (FCT) for both short and long flows. The FCT of a flow is the time difference between the time when the first packet of a flow leaves the source and the time when the last packet of the same flow arrives at the destination [12].

Fig.4 shows the comparison result between the proposed UDA forwarding and reactive forwarding for short flows with various tiny VMs. The result shows no significant change in the average FCT for short flows. However, Fig.5. Shows improvement using UDA forwarding compared to reactive forwarding method using long flows. The figure shows average FCT reduction of 2.3% -19.6% using the proposed method. This is because the proposed method reroutes the large flows to the least utilized and least delay path while the reactive forwarding method only uses the shortest paths for all traffic flows.

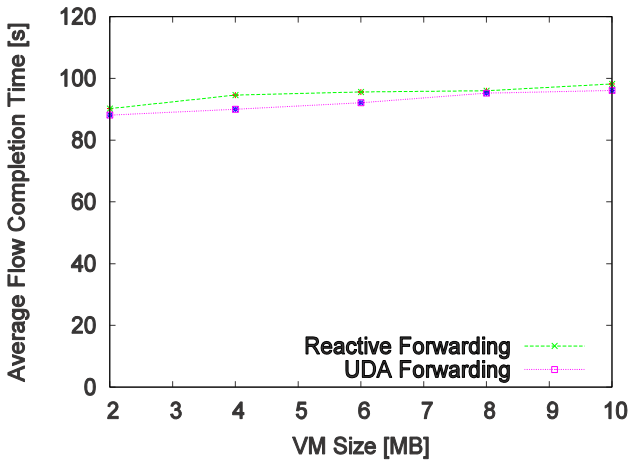

Fig. 4. Short Flow: Comparison of Average Flow Completion time with various VM sizes

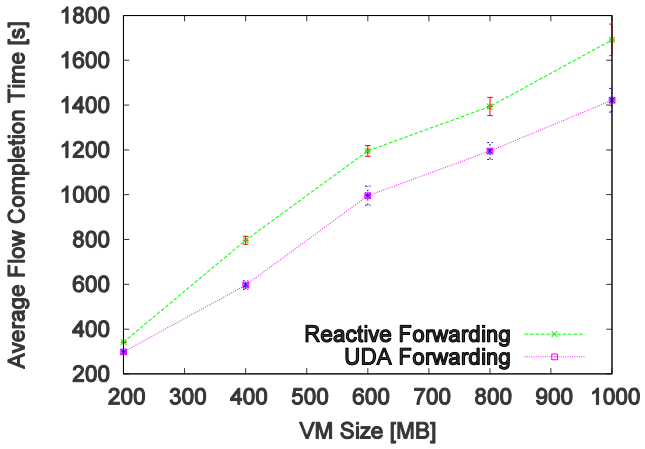

Fig. 5. Long Flow: Comparison of Average Flow Completion time with various VM sizes

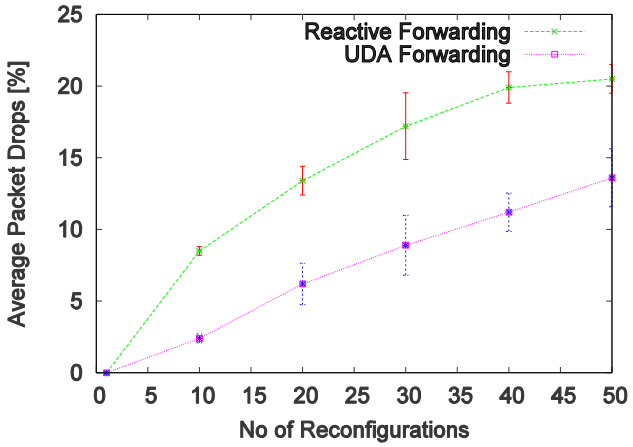

Fig. 6. Long Flow: Comparison of Average packet Drops

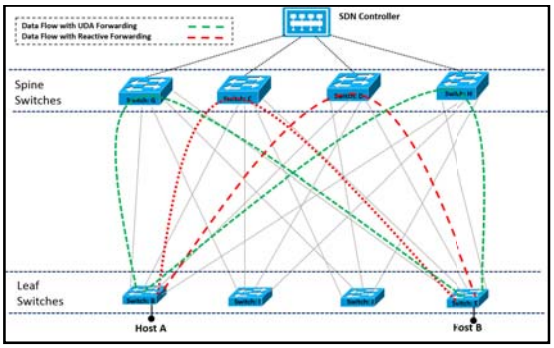

Fig. 7. Use case Scenario : Bandwidth Utilization using the proposed UDA Forwarding Approach.

Frequent path reconfigurations in a network can increase the number of packet drops, which lead to retransmissions and finally can degrade application performance. However, the frequency of path reconfiguration is usually enlarged by severe variation of traffic. Few research studies have addressed the adverse influence of frequent reconfiguration as well as the mechanisms to avoid it.

Therefore, path reconfiguration is introduced in the second experiment with background traffic to match varying traffic demand in order to check the QoS of the network. As we could not find any significant improvement for the short flows in our previous experiments, in this scenario, the proposed UDA forwarding is compared with the reactive forwarding for large flows during path reconfigurations.

In this experiment, iperf UDP packets are used with a constant bit rate of 100 Mbit/sec to compare the UDA forwarding with the reactive forwarding by calculating the average packet drops for a period of 10 sec. Table II shows the details configuration parameters. According to our experimental results as shown in Fig.6 the growth of reconfiguration frequency leads to higher dropped packet counts using reactive forwarding, whereas using the proposed UDA forwarding shows improvement by reducing 1% - 8% of the packet drops. This is because the ONOS controller uses less utilized and least delay paths for packet fording and avoids congestion with fewer packet drops compared to reactive forwarding method.

C. Use case Scenario

Fig. 7 shows a use case scenario for bandwidth utilization using UDA forwarding compared to Reactive forwarding in a network from Host A to Host B. The Scenario is considered during two sets of flows with background UDP traffic that contains 30% of the link capacity. The Details of the configuration parameters can be found in Table II.

Using Reactive forwarding, the first set of flows is chosen through the Leaf switch (B) and Spine switch (C), with random interval 1-2s. The second set of flows are sent through the Leaf switch (B) and Spine switch (D). In this scenario, using the UDA approach, the controller first selects the least delay path and calculates the bandwidth utilization.

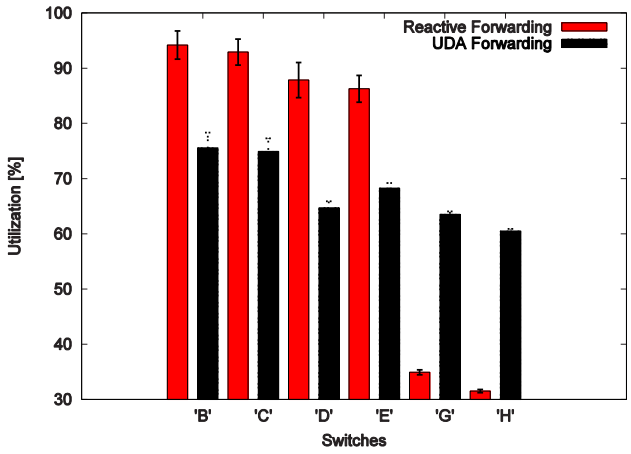

Fig. 8. Utilization Comparisons between UDA and Reacive Forwarding

If the utilization is less than the threshold level, the controller forwards the packet. Here, the controller selects Spine switch (G) for the first set of flows and leaf switch (B). For the second set of flows, it selects Spine Switch (H) with leaf Switch (B) to reach Host B. Finally, the SDN controller updates the link information periodically for next transmission.

Fig.8 shows the overall effect of the proposed UDA forwarding approach, where all the switches better utilize its bandwidth during the transfer compared to reactive forwarding method that over-utilizes some of the switches. Hence, the overutilization using reactive forwarding may reduce the overall application performance by higher packet drops.

TABLE II. CONFIGURATION PARAMETERS: LONG FLOW UTILIZATION EXPERIMENT

Parameters	Value
Leaf switches	4
Spine switches	4
Host to Leaf switches Bandwidth	1Gbps
Leaf to Spine switches Bandwidth	1Gbps
Traffic	TCP
First Set of Flows (VM1)	600MB
Second Set of Flows (VM2)	400 MB
Interval between Flows	1-2 sec
Background Traffic	UDP, 30%

V. CONCLUSION AND FUTURE WORK

Although SDN is in the most challenging technology domain, it is still popular with a lot of business because of the adaptive solutions without any expensive investments. The paper presented an efficient rerouting algorithm for congestion control by monitoring link utilization and considering least delay path in SDN environment. Any new flow is routed to the shortest path normally considering that the path is not over-utilized or congested. However, if the threshold level is crossed for a certain switch, our method bypasses that switch by rerouting to another switch. It chooses the minimum number of flows possible for resolving the congestion and reroutes them to the less utilized and least delay path. From the experimental results, our investigation of the proposed method optimizes the network performance with lower FCT and fewer packet drops for large flows than the reactive forwarding method.

For future directions, our research will focus on reducing the overhead while measuring network statistics for network monitoring through smart algorithms as operations overhead grows according to the scale and complexity of the network. Our work will also explore on more useful SDN monitoring applications by good use of programming languages and APIs that will help researchers to develop various monitoring mechanisms for achieving more flexible, adaptive, and high-level control characteristics in SDN monitoring.

ACKNOWLEDGMENT

This research is supported by the 'Agile Cloud Service Delivery Using Integrated Photonics Networking' project funded under the US-Ireland Programme NSF (US), SFI (Ireland) and DfE (N. Ireland). The authors would also like to thank British Telecom (BT) and Invest NI for partly supporting this research under the BT Ireland Innovation Centre (BTIIC).

REFERENCES

[1] Nakao, A.: Network virtualization as foundation for enabling new network architectures and applications. IEICE Trans. commun. 93(3), 454–457 (2010).

[2] ONF, OpenFlow Switch Specification Version 1.5.0, https://www.opennetworking.org/images/stories/downloads/sdnresources/onfspecifications/openflow/openflow-switchv1.5.0.noipr.pdf, December 2014.

[3] M. I. Biswas, G. Parr, S. McClean, P. Morrow and B. Scotney, "SLA-Based Scheduling of Applications for Geographically Secluded Clouds", 1st workshop on Smart Cloud Networks & Systems (SCNS'14), December 3-5 Paris, France, 2014.

[4] M. I. Biswas, G. Parr, S. McClean, P. Morrow and B. Scotney, " A Practical Evaluation in Openstack Live Migration of VMs using 10Gb/s Interfaces", The Second International Workshop on Education in the Cloud-EC2016, 29 March-2 April 2016, Oxford, UK.

[5] ONOS. [Online] Availible: http://github.com/opennetworkinglab/onos

[6] S. Hegde, S. G. Koolagudi, S. Bhattacharya, "Scalable and fair forwarding of elephant and mice traffic in software defined networks," Computer Networks, vol. 92, pp. 330–340, December 2015.

[7] F. P. Tso and D. P. Pezaros.,"Improving Data Center Network Utilization Using Near-Optimal Traffic Engineering", IEEE Transactions on Parallel and Distributed Systems, 2013.

[8] S. M. Park, S. J. and J. Lee,"Efficient routing for traffic offloading in Software-defined Network", Procedia ComputerScience 34 : 674- 679, 2014.

[9] S. Song, J. Lee, K. Son, H. Jung, 1. Lee, "A congestion avoidance algorithm in SDN environment", IEEE International Conference on Information Networking (ICOTN), pp. 420-423, 2016.

[10] Pang-Wei Tsai; Chun-Wei Tsai; Chia-Wei Hsu; Chu-Sing Yang, "Network Monitoring in Software-Defined Networking: A Review", IEEE Systems Journal, 2018.

[11] "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet." [Online]. Available: http://mininet.org/

[12] F. Carpio, A. Engelmann, A. Jukan, "DiffFlow: Differentiating Short and Long Flows for Load Balancing in Data Center Networks," in Proc. IEEE GLOBECOM, vol. 10, pp. 1–6, April 2016.

[13] iPerf - The network bandwidth measurement tool, https://iperf.fr/.