

Flow Optimization in Multipath Network using Maximum Flow Algorithm to Improve the Bandwidth Usage in SDN

MSc Research Project
Cloud Computing

Vivek Medleri Hire Math
Student ID: x18147062

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Vivek Medleri Hire Math
Student ID:	x18147062
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	23/04/2020
Project Title:	Flow Optimization in Multipath Network using Maximum Flow Algorithm to Improve the Bandwidth Usage in SDN
Word Count:	5806
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	26th May 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Flow Optimization in Multipath Network using Maximum Flow Algorithm to Improve the Bandwidth Usage in SDN

Vivek Medleri Hire Math
x18147062

Abstract

Traditional network architecture is distributed and complicated compared to the Software Defined Networking (SDN) that modifies the distributed network architecture to a centralized one and makes the management of the network easy. The problem arises in the network flow of the multipath network and finding the maximum flow among the source node and destination node. But, the present network situation in the data centers which uses the multipath topology needs a maximization in the flow to improve the bandwidth usage. This research presents to find the maximum flows and avoiding augmented paths (where we can call this as finding the path barrier) in the multipath network and forwarding the network traffic flow from source to the destination. In this project, a Dinic's max flow algorithm is used to find the maximum flows and maximize the bandwidth usage in the software defined network. The Mininet network emulator and RYU controller is used in the implementation. The outcome from the investigation of this research removes network congestion and complications which improves the network bandwidth usage.

1 Introduction

The next generation of network infrastructure can be defined as Software defined networking (SDN), it gives the network more control over the flexibility and efficiency. The software defined network has two planes, the control plane and the data plane. These two planes are separated from each other, the network operators will have the ability to control which packets are transmitted over the paths in the network Yoshioka et al. (2017). To simplify and improve the management and programmability of the networks, a lot of attention has been given to SDN and this has led in real deployment. For example, Google have implemented in their data center. They have used SDN OpenFlow-based network to interconnect all of its data centers all over the world which eases in the performance and flexibility in traffic engineering functions, this gives them nearly 100 percent link utilization. These aspects provide a great benefit in migrating to software defined networks (Hu et al.; 2015).

Generally, a SDN will have number of OpenFlow-enabled switches (OF-switches) and SDN centralized controllers. From various networks the OF-switches will forward the traffic, like the cellular networks and IP data networks. To the assigned SDN controller

the responsible OF-switch will send the routing request when there is a newly generated data flow. The control message is received by the controller and the optimal path routing to the destination OF-switch is calculated and along the optimal path the routing tables of switches is set up. Dedicated in-band control and out-bound control are the two types of control channels. The out-band control creates links between its responsible controllers and each OF-switch using a dedicated network. On the other hand, the in-band control shares the same forwarding infrastructure which allows data messages and control. The in-band control is cost effective compared to the out-bound control (Lin et al.; 2018).

The SDN controller are defined by the flow-rules and are installed at the switches used by ternary content-addressable memory (TCAM). Due to more energy consumption and high cost the availability of TCAM at a switch is limited. Due to the limited TCAM, the number of flow rules that are inserted are also limited to a switch. Three different strategies were proposed by the researches like exact-match, hybrid and wildcard to place flow rules at the switches. These three strategies serve for different purposes, in exact-match, an individual rule is associated with each flow and therefore there is an increase in network visibility. The wildcard-based strategy has few flow rules and are associated with multiple rules. The last strategy called the hybrid strategy is considered as a combination of both the exact-match and the hybrid. Accordingly, to generate the packet messages, more flow rule installations are requested, which makes the controller overhead to increase (Bera et al.; 2019).

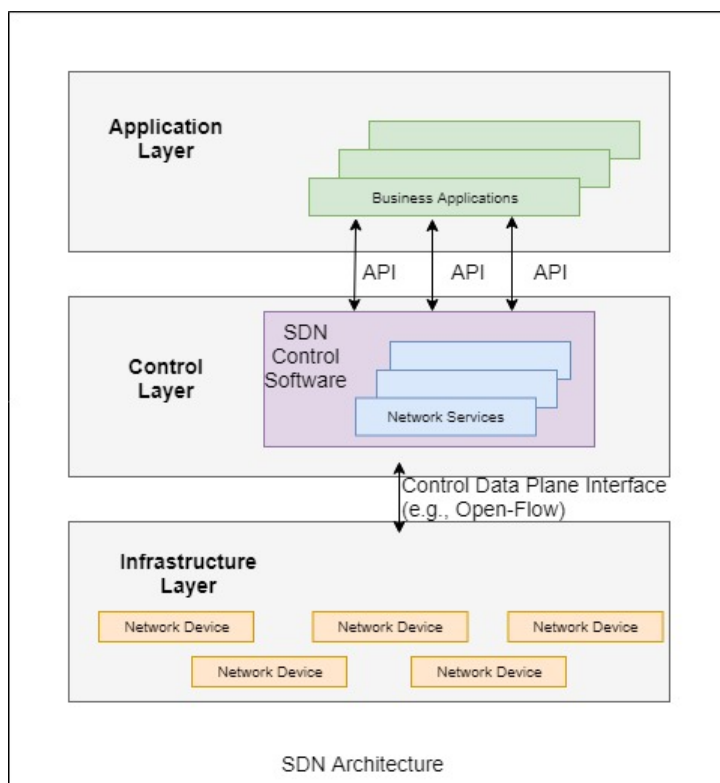


Figure 1: SDN Architecture

As shown in above Figure 1, software defined architecture contains the application layer, control layer and infrastructure layer (Liu et al.; 2019).

Basically, in Software Defined Network (SDN) control-plane and data-plane are separated. Transfers the control-plane to the central location, runs on generalised purpose computer which permits the network-control to be direct programmable and the fundamental infrastructure to be abstracted for network and application services. software defined networking provides support for software program to control the network. SDN mechanism is a method for cloud computing that eases the network administration and allows programmable effective network configuration to achieve better performance with monitoring.

Research Question: Can Software Defined Network controller assist to monitor the entire topology and find the maximum flow in the multipath network to increase the network bandwidth usage by using Dinic's maximum-flow algorithm?

This question arises because network service providers have many number of topology which is connected through the cities with link bandwidth, and also which contains multipath connectivity across the topology so, Often some links are occupied and some are idle which distress the client/customer traffic, which may leads to bad quality of service and network congestion. So, in order to keep the network without overcrowding, maximization of flow is required to improve the bandwidth.

The rest of the paper is divided into following sections. Section 2 with the related work, section 3 with the Methodology, section 4 with the Design specification, section 5 with the implementation, section 6 with the evaluation and section 7 with the conclusion and future work.

2 Related Work

In this segment we first see how the multipath network architecture are used in traditional scheme. Also, analysing the SDN multipath routing with flow maximization.

2.1 Analysis of Multi-path Network

A technology of choice which emerged was known as multipath provisioning and offers many advantages which are as follows:

- Traffic Engineering which is described in the internet and the issue it deals with is the performance optimization and performance evaluation of IP networks which are operational.
- The main aspect of traffic engineering is load balancing and congestion control and it can be achieved by the use of multipath provisioning. To avoid network hot-spots, the links which are optimally loaded over multiple concurrent paths will be distributed by the traffic flows in the network.

- With the aspect of multipath provisioning a reliable communication can be made without any effort needed with the implicit fault tolerance. An alternate path is used by the routing protocols when a path fails inside a single path routing.
- The deployment of multipath routing can be improved by including another aspect such as the network resource utilization. To get higher network throughput there has to be higher resource utilization like bandwidth.
- Security is one of the most important aspect when it comes to a single-path routing and it is vulnerable to the security threats like denial of service attacks caused by overloading a particular node, path or a link, greater security can be provided by multipath routing by the dispersion of data in multiple paths between end-hosts, between a source-destination pair where each of the path can carry a portion of data (Singh et al.; 2015).

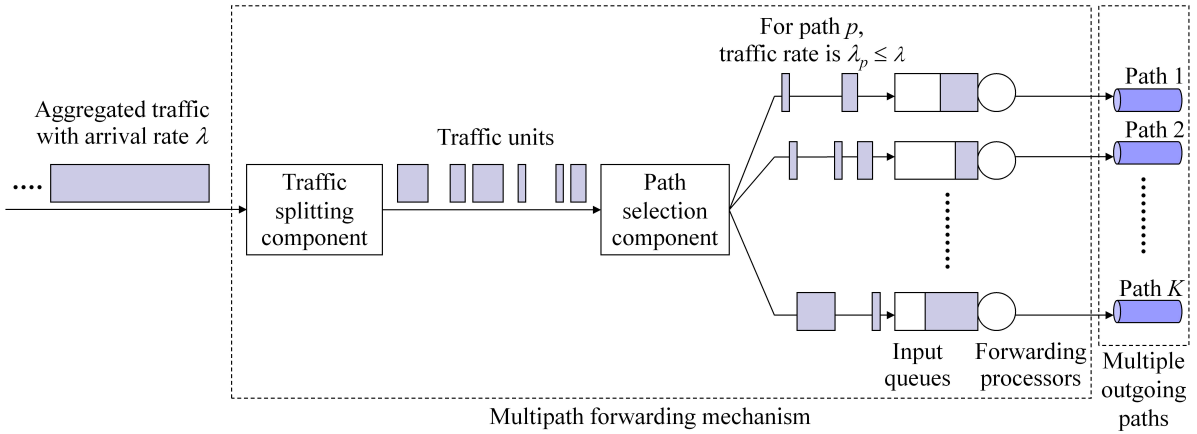


Figure 2: Multipath Forwarding Mechanism

Figure 2 describes the traffic splitting and path selection of the functional components of multipath forwarding. The splitting of the traffic into traffic units is done by the traffic splitting components, where each of the path is taken individually, and the path selection component is determined. In the path selection if the forwarding processor is busy then the output link will be attached by each traffic unit in the input queue. Each of the different models of multipath forwarding will perform the load distributions in numerous manners. Due to the internal functional components there are different shortcomings and advantages that each of the models exhibit, like the path selection and traffic splitting (Prabhavat et al.; 2012).

The mechanism that is required for differentiating elephant flows from mice flows is scheduled by the mechanism of elephant flows. Previously there were few problems over the detection of elephant flow and it has been thoroughly researched and there are many solutions available. To detect the elephant flows, the most effective way is at the end-hosts. Our mechanism relates to the multipath and flows in our network. To monitor the end-hosts it uses a shim layer which is integrated with the mechanism of TCP socket buffers. Identification of the flows is done by the shim layer and it determines if it is an

elephant flow or any other flow in the network when the buffer exceeds and the number of bytes is predefined and the threshold rate is given by the time window. In our project we will simplify the multipath and the flow of which the network can improve its bandwidth (Liu et al.; 2014).

Sehery and Clancy (2017) illustrates how the flow optimization in data centers works with Clos networks. The Clos network can be formulated with a detailed example of how the routing is carried on the Binary Multicommodity Flow Problem. This has been done by the simulation of using a general integer linear programming solver that optimizes the optimal flow routing without collisions in the flow, in polynomial time it can be achieved, but this is not practical Ai et al. (2019). investigated how to deal with the problem of passive attacks in the network coding-based resilient multipath routing. To measure the passive attacks a novel metric is used for resilience and snooping the ratio. For solving this problem, a simulation is designed for annealing-based algorithm to solve it efficiently.

2.2 Analysis of SDN Multipath Routing

Software defined network (SDN) growth is very much aggressive over the years and many SDN controller mechanisms and products are released. Woo et al. (2018) investigates on the SDN and provides the resourceful on important services provided by the SDN controllers, because services help users to implement network virtual functions through programmable technique and validates and provides some vulnerable situations in software defined networking.

Adaptive Multipath Routing (AMR) contains an OpenFlow centralized controller-based application intended to learn topology, with maximum flow capacity calculate multiple paths among the nodes, and dynamically modify the forwarding table of switches to set up loop-free multipath forwarding and routing. Subedi et al. (2015) presented Adaptive Multipath Routing (AMR), which is capable of proactively adapting to network changes based on link capacity and latency, but their approach heavily relies on Link Layer Discovery Protocol (LLDP) as the discovery mechanism. Basically, resolution increases network utilization of the data-centers and combined bandwidth is accomplished per flow by using multiple paths.

For high mobility and throughput, multi-connection virtual access point is proposed to enable multiple transmission paths simultaneously over a set of access points for users. Xu et al. (2017) presented novel Multipath-Transmission supported Software-Defined Wireless Network (MP-SDWN) architecture, with the aim of achieving seamless handover, throughput enhancement, and flow-level wireless transmission control as well as programmable interfaces.

Rezende et al. (2019) proposes an SDN-based framework which gives an interface for the applications and specifies multi-stream rules so that the framework can use the services given by the software defined networking controller to guarantee that the multiple-stream work on the multipath network.

Sheu et al. (2016) proposes a heuristic algorithm for solving the polynomial time in a network. The main focus is on the Multipath TCP (MPTCP). It is an extension of

the TCP where the TCP communication with throughput significantly uses the multiple paths instead of the single path for transmission. A simulation is carried out to show that, the proposed algorithm performs better with an increase in the average throughput and hop count.

Benson et al. (2011) proposed micro-TE, and they have implemented micro-TE within the open-flow framework with minimal changes in destination hosts. And micro-TE is a centralized system that adapts to traffic variations by leveraging the short-term predictability of the DCN (Data-Center-Network) traffic, to achieve fine grained TE. It constantly monitors traffic variations, determines which Top-of-Rack (ToR) pairs have predictable traffic, and assigns the predicted traffic to the optimal path. Similar to Hedera, the remaining unpredictable traffic is then routed using weighted ECMP (Equal-cost Multipath), where the weights reflect the available capacity after the predictable traffic has been assigned.

Guillen et al. (2017) present a pragmatic approach for multipath routing in Distributed Storage Systems (DSS), which is based on Software Defined Networking (SDN) that uses parallel links at the edge-side. Path discovery is calculated by finding the k-maximum disjoint paths in a multi-graph.(Guillen et al.; 2018) also proposes a hybrid approach combining server and link load balancing for multipath routing in DSS. The approach is Software Defined Networking (SDN) based, and uses a process called on-demand inverse multiplexing.

2.3 Analysis of SDN in Maximizing the Flows

The main problem in hybrid SDN (Software Defined Network) network is the traffic engineering problem which concentrates on how to satisfy the global objectives of network performance. By the set of given SDN nodes, the SDN devices main goal is to find the maximum flow by tuning the forward behaviours in the network. Through partial SDN deployment, the maximum flow problem can be formulated within the networks, to develop and solving a fast Fully Polynomial Time Approximation Scheme (FPTAS). The performance of hybrid SDNs are better compared to the traditional OSPF networks and this is proved by the results gained the extensive simulation using the real network topologies. The network maximum flow can be demonstrated by the impact of node numbers in SDN, and conclude that by deploying almost 50 percent of the SDN can lead a near gain in the optimal performance. (Hu et al.; 2015).

In SDN, only a few works have been dealt with the SDN network utilization with limitation of forwarding table size. A problem known as path-degree max flow problem has been used. Each node in a network has limited sizes of forwarding tables that has the maximum number of paths that can pass through a network. It is known as the forwarding table size of the node or the path degree. There is a difference in the formulation of forwarding table and we can guarantee that a certain QoS and the performance of the individual flows are not considered. (Yao et al.; 2016).

3 Methodology

The outline of the multipath software defined networking is designed by experimentation and simulation which exhibits the benefits and challenges of this methodology and support to make suitable selections. This section defines the architecture of the research-project and express the reason behind the decisions that are taken during the experimentation.

3.1 Architecture of Multipath

This section describes the approach and design of this research project. For this approach because of the programmable networking software defined network has been preferred which has the enormous propositions to support. And software defined network can easily be installed on commodity-hardware and further helps for implementation and testing. And this can be easily used in the data-center network.

The architecture of the flow optimization in the multipath network contains a centralized software defined network (SDN) controller with the information of the topology which dynamically sets up loop forwarding-rules on the switches. The controller doesn't have real time information of the network as it is for experimentation and use the primary structure as a directed graph where the capacity of the link among the nodes are pre-defined. From this information optimal multipath is computed and sends the data from one node to all other nodes. With a max flow algorithm based on Dinic's algorithm maximizes the bandwidth usage to solve the maximum flow problem. In the circumstances of network congestion, failure, or block of paths software defined controller will re-route by computing the new configuration and forward it to the switches. From this multi-path are manipulated for increased bandwidth and also over-all data-throughput is increased due to the traffic acceleration through the multi-paths. And this is the situation of data-center topology of multipath where the nodes are controlled and traffic initiates from the hosts connected, for instance please refer the architecture in Figure 3 below. And this approach achieves the main aim of this research project.

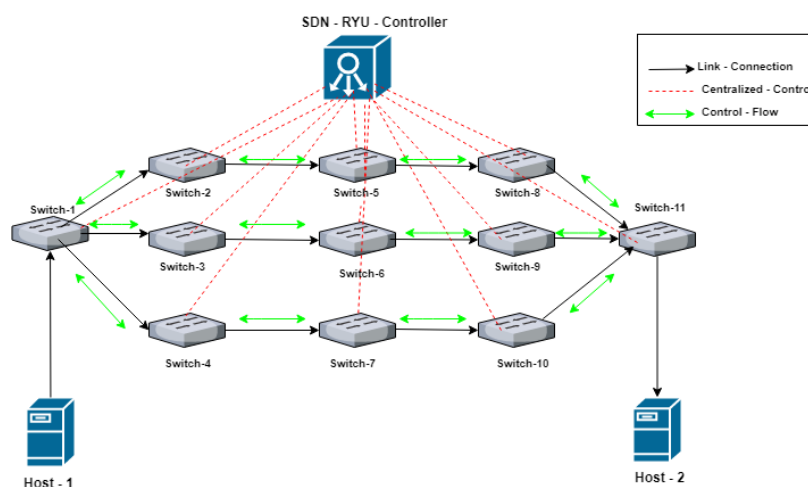


Figure 3: Multipath Architecture

4 Design Specification

In this section we will describe the high-level-diagram of this research implementation. To start the maximum-flow approach we need all the paths of the network as input. When the maximum-flow approach starts BFS (Breadth-First-Search) checks the level of multipath network graph, by doing this we will get to know the value of every node to find the short distance from source node. And the edges from the level graph do multiple iterations of DFS (Depth-First-Search) approach from source to destination until there are no augmented paths possible, then the maximum flow is found. Once we received the maximum flow-value RYU-Controller sets that as constant value for maximum paths which is used for multipath routing. Through this approach we can improve the throughput(bandwidth). The below Figure 4 defines the high-level execution of this project.

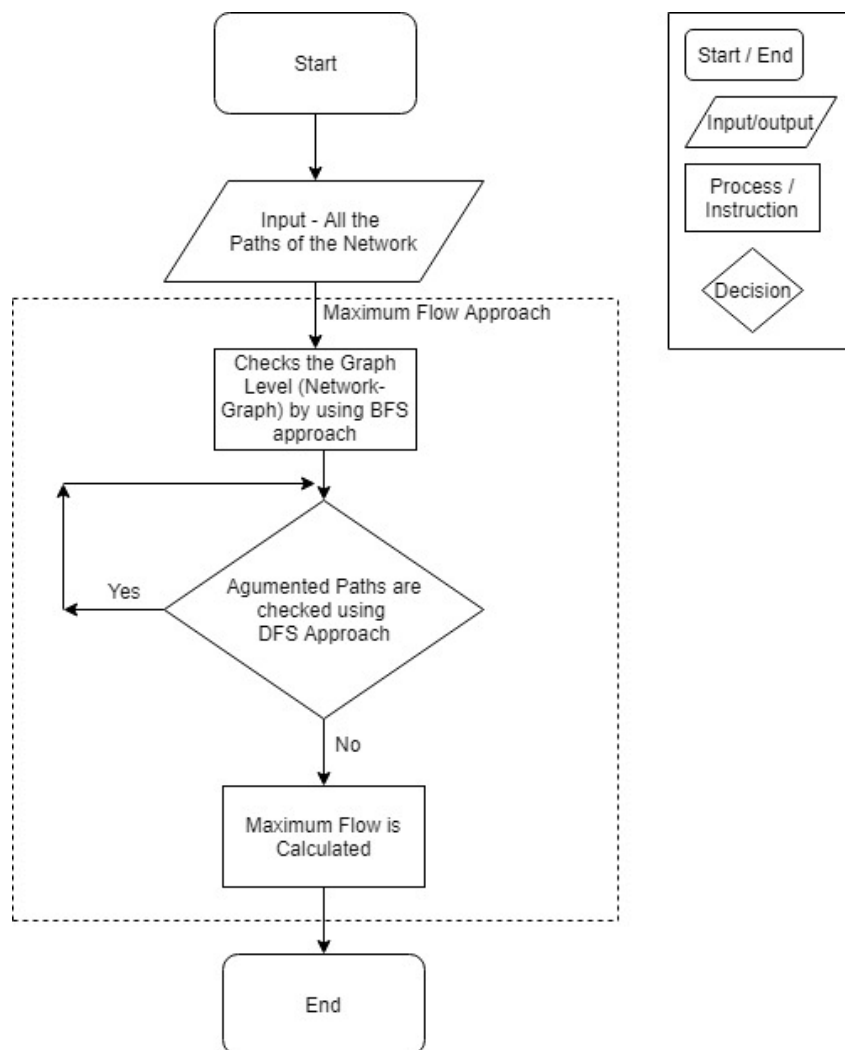


Figure 4: Flow Chart

5 Implementation

This section deliberates the resolution for research question which is defined earlier. The proposed max flow algorithm is developed by modifying the RYU controller. For evaluating this algorithm multi-path topology is created and used for the implementation. Below section 5.1. provides an outline of experimental setup. Followed by section 5.2 which defines network emulation setup and the specifics of the RYU-controller algorithm.

5.1 Outline of Experimental Setup

The subsequent software and hardware components are used to perform this experiment. In Table 1 provides the list of the software with description.

Table 1: Software used for implementation.

Software	Software-Tool	Version
OS	Ubuntu-Desktop	18.04
Test Bed	Mininet	2.3.0d6
Controller	RYU	4.4
Switch	Open-v-switch	2.9.5
Traffic Generator	IPerf	2.0.10
Virtual Box	Oracle VM Virtual Box Manager	6.1

In Table 2 provides the list of the hardware specification.

Table 2: Hardware specifications used for implementation..

Hardware	Version
CPU - Processor	Intel Core - i5 - 2.3Gz
RAM	4 GB
Hard-Disk-Capacity	40 GB

5.2 Network Emulation Setup

Mininet is python based and open-Source network emulator which lets scholars to generate simulated networks with hosts and switches in a machine. It offers an easy going and low-cost test-bed of network for generating the open-flow applications and permits to test the complex topologies without wiring up the physical network. It consists of Command-Line Interface (CLI) for debugging and executing the network-wide trials. And mininet gives the easy way for getting accurate performance results and to test with topology ¹. And mininet can be installed anywhere in a cloud machine, desktop or a server.

5.2.1 Implementation of Multipath-Topology in Mininet

To implement the methodology that we have discussed in the section 3, we have used Mininet-version-2.3.0d6 SDN (software defined network) simulation tool to run the network topology (Multipath-Topology). We have executed Multipath-Topology which has

¹<http://mininet.org/overview/>

2 hosts and 11 Open-vSwitches which represents in the below Figure 5. And Figure 6 denotes the network i.e., hosts, switches are established. We have used the pingall command to check the connectivity among the switches and host. When we do pingall command it should result in zero percent dropped with all the packets received and this defines hosts are active and up. And this happens when we have connected topology to the controller. And we have used one RYU controller with the mininet which we have discussed RYU controller deployment in the below section 5.2.2. To transmit the data-packets, 1Mbit bandwidths are set between the switches and the hosts.

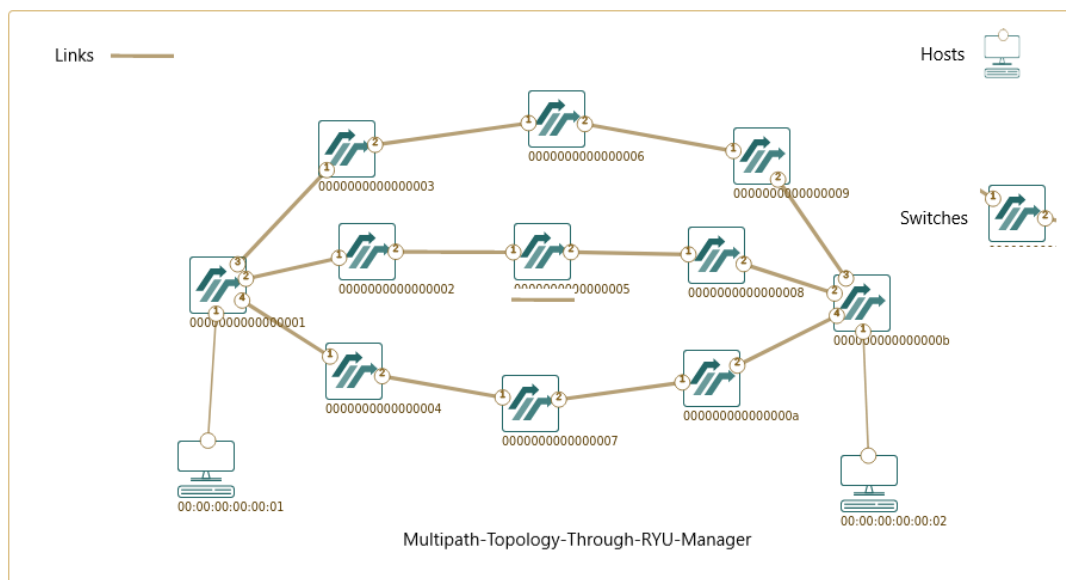


Figure 5: Multipath-Topology

```

x18147062-vivek@x18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller$ ls
Multipath_Topology_14_Links_1C.py Multipath_Topology_22_Links_1C.py
x18147062-vivek@x18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller$ sudo python
Multipath_Topology_14_Links_1C.py
*** Adding switches
*** Adding hosts
*** Adding controllers
*** Adding links
(1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (
1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1
.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit)
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller 1
*** Starting switches
(1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1
.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1
.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) *** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>

```

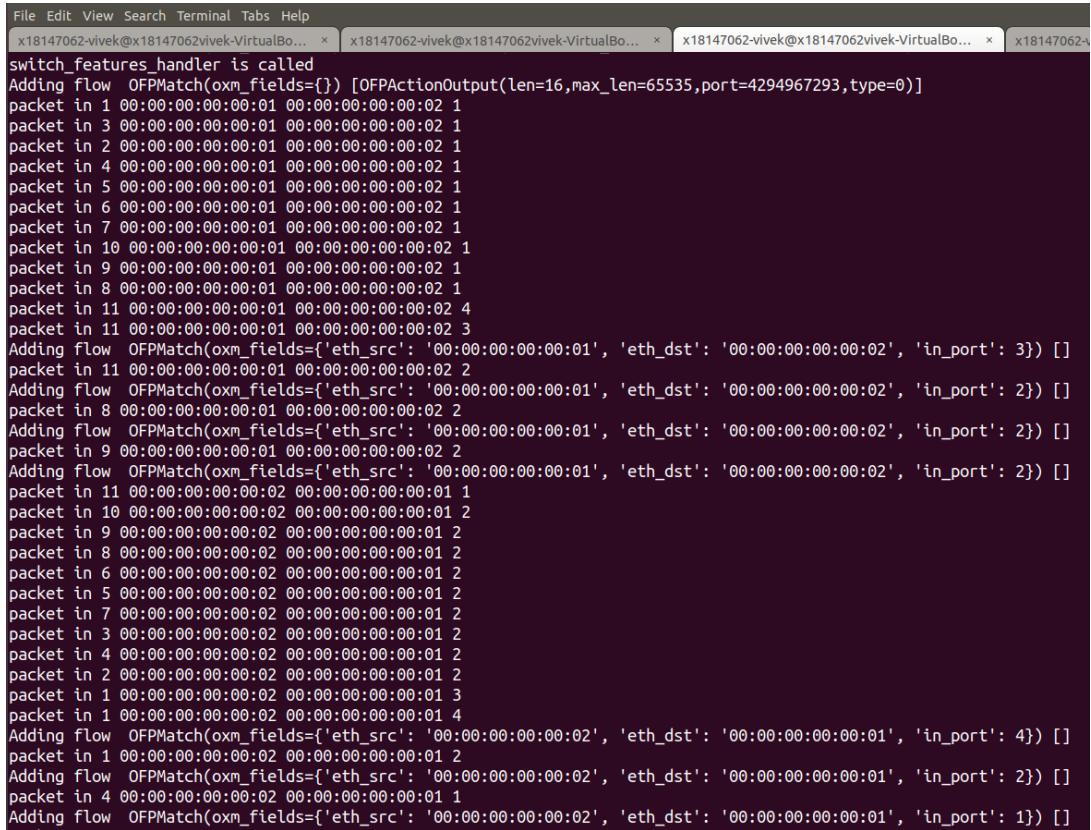
Figure 6: Multipath-Implementation

6 Evaluation

In this section we have undergone with connectivity test between mininet emulator and the ryu-controller and we have captured the test results to evaluate the developed RYU controller by using max flow algorithm for multipath topology. We have captured the transferred packets, path bandwidth which we can say as throughput between the hosts and also compared with another multipath RYU controller which they have used Dijkstra algorithm and developed RYU controller for multipath topology. Below sections will give the more information on the test results and the setup.

6.1 Connectivity Test with mininet and RYU Controller

The multipath topology network is created with 11 switches and 2 hosts as explained in 5.2.1. to execute on mininet. To implement the Dininc's max flow algorithm RYU-controller (python script name: Main-Dinics-Implementaion.py) as proposed, we need all paths between the hosts included in multipath network as input. Switches uses open-flow mechanism to connect the hosts. So, to test the connection between the hosts we have used pingall command to check the connectivity and switches (shows all 11 switch features are called and adding flows) are working as shown in Figure 6 and Figure 7 in section 5.2.1 and in section 5.2.2 respectively. And the continuation of the output from Figure 7 in section 5.2.2 shows all paths are working between the host-H1 and host-H2 with port numbers 1,2,3,4 as shown in below Figure 8.



```
File Edit View Search Terminal Tabs Help
x18147062-vivek@x18147062-vivek-VirtualBo... x18147062-vivek@x18147062-vivek-VirtualBo... x18147062-vivek@x18147062-vivek-VirtualBo... x18147062-
switch_features_handler is called
Adding flow OFPMatch(oxm_fields={}) [OFPActionOutput(len=16,max_len=65535,port=4294967293,type=0)]
packet in 1 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 3 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 2 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 4 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 5 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 6 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 7 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 10 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 9 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 8 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 11 00:00:00:00:00:01 00:00:00:00:00:02 4
packet in 11 00:00:00:00:00:01 00:00:00:00:00:02 3
Adding flow OFPMatch(oxm_fields={'eth_src': '00:00:00:00:00:01', 'eth_dst': '00:00:00:00:00:02', 'in_port': 3}) []
packet in 11 00:00:00:00:00:01 00:00:00:00:00:02 2
Adding flow OFPMatch(oxm_fields={'eth_src': '00:00:00:00:00:01', 'eth_dst': '00:00:00:00:00:02', 'in_port': 2}) []
packet in 8 00:00:00:00:00:01 00:00:00:00:00:02 2
Adding flow OFPMatch(oxm_fields={'eth_src': '00:00:00:00:00:01', 'eth_dst': '00:00:00:00:00:02', 'in_port': 2}) []
packet in 9 00:00:00:00:00:01 00:00:00:00:00:02 2
Adding flow OFPMatch(oxm_fields={'eth_src': '00:00:00:00:00:01', 'eth_dst': '00:00:00:00:00:02', 'in_port': 2}) []
packet in 11 00:00:00:00:00:02 00:00:00:00:00:01 1
packet in 10 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 9 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 8 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 6 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 5 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 7 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 3 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 4 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 2 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 1 00:00:00:00:00:02 00:00:00:00:00:01 3
packet in 1 00:00:00:00:00:02 00:00:00:00:00:01 4
Adding flow OFPMatch(oxm_fields={'eth_src': '00:00:00:00:00:02', 'eth_dst': '00:00:00:00:00:01', 'in_port': 4}) []
packet in 1 00:00:00:00:00:02 00:00:00:00:00:01 2
Adding flow OFPMatch(oxm_fields={'eth_src': '00:00:00:00:00:02', 'eth_dst': '00:00:00:00:00:01', 'in_port': 2}) []
packet in 4 00:00:00:00:00:02 00:00:00:00:00:01 1
Adding flow OFPMatch(oxm_fields={'eth_src': '00:00:00:00:00:02', 'eth_dst': '00:00:00:00:00:01', 'in_port': 1}) []
```

Figure 8: RYU-controller-Connectivity Test

6.2 Results

Iperf tool is used to measure the bandwidth in the network. The software defined network RYU-Controller uses its handlers for sending open-flow messages among the switches. With the Iperf tool UDP (User Datagram Protocol) and TCP (Transfer Control Protocol) throughput can be measured by sending the packets among the host-1 and host-2.

6.2.1 Case Study 1 - TCP and UDP for proposed algorithm

- **TCP Test between Host-1 and Host-2**

TCP (Transmission Control Protocol) packets transmission tested between the two hosts i.e., host-1(h1) with IP-address-10.0.0.1 and host-2(h2) with IP-address 10.0.0.2. And we are capturing the results every one (-i) second by starting the TCP as server (-s) at host-2 with port (-p) 5555 as shown in Figure 9. Meanwhile we have started the TCP as client (-c) at host-1 and duration of the transmission is set to twenty-seconds (-t) as shown in Figure 9 and store the results in another file to plot the graph through gnuplot. Basically, here we are calculating the time and bandwidth for the quantity of data sent. Thus, from the results average bandwidth throughput among the time interval i.e., 0.0 to 22.6 seconds sent from host-1 to host-2 is 232 Kbits/sec with 640 Kbytes packets transferred.

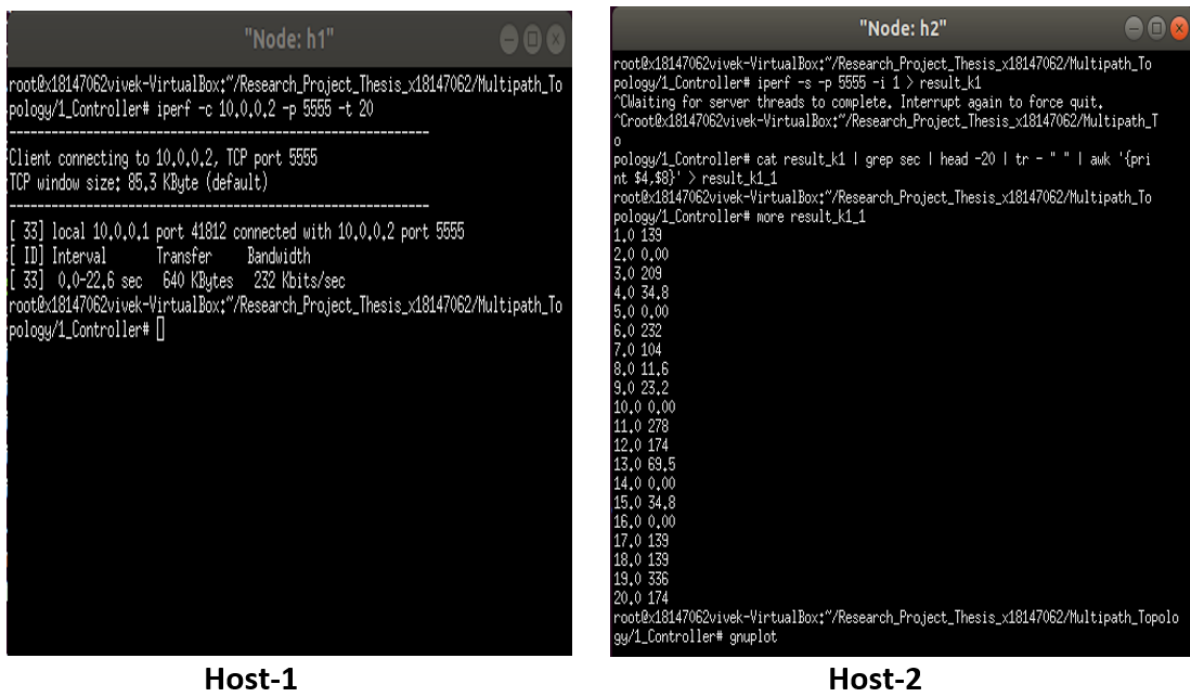


Figure 9: TCP-Testing-A

- **UDP Test between Host-1 and Host-2**

As identical to TCP (Transmission Control Protocol), similarly we have tried to test the UDP (user datagram protocol(-u)) between the Host-1(h1) and Host-2 (h2). We have

started the UDP as client (-c) at host-1 and duration of the transmission is set to twenty-seconds (-t) as shown in Figure 10. In meantime we have started capturing the results every one (-i) second by starting the UDP as server (-s) at host-2 (h2) with port (-p) 5555 as shown in Figure 10 and store the results in another file to plot the graph through gnuplot. Average bandwidth throughput among the time interval 0.0 to 20.1seconds sent from host-1 to host-2 is 1000 Kbits/sec with 2.39 MBytes packets transferred. And also, after the 10 tries with a warning that 1708 datagrams of messages went with data-loss.

The figure shows two terminal windows side-by-side. The left window, titled "Node: h1", shows the execution of the iperf client command: `iperf -c 10.0.0.2 -u -b 10M -t 20 -p 5555`. It displays connection details and a table of results for the 0.0-20.1 second interval, showing a bandwidth of 1000 Kbits/sec and 2.39 MBytes transferred. A warning message indicates that 1708 datagrams were lost. The right window, titled "Node: h2", shows the iperf server command: `iperf -s -u -p 5555 -i 1 > result_j2`. It shows the server capturing data and the results being written to a file named `result_j2_1`. The results in the file show a bandwidth of 1000 Kbits/sec and 2.39 MBytes transferred over the 20-second interval.

```

"Node: h1"
root@18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller# iperf -c 10.0.0.2 -u -b 10M -t 20 -p 5555
-----
Client connecting to 10.0.0.2, UDP port 5555
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 33] local 10.0.0.1 port 42789 connected with 10.0.0.2 port 5555
[ ID] Interval      Transfer      Bandwidth
[ 33] 0.0-20.1 sec  2.39 MBytes  1000 Kbits/sec
[ 33] Sent 1708 datagrams
[ 33] WARNING: did not receive ack of last datagram after 10 tries.
root@18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller#

"Node: h2"
root@18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller# iperf -s -u -p 5555 -i 1 > result_j2
"Waiting for server threads to complete. Interrupt again to force quit."
root@18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller# cat result_j2 | grep sec | head -20 | tr - " " | awk '{print $4,$5}' > result_j2_1
root@18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller# more result_j2_1
1.0 11.8
2.0 70.6
3.0 129
4.0 11.8
5.0 259
6.0 270
7.0 153
8.0 259
9.0 235
10.0 153
11.0 270
12.0 141
13.0 259
14.0 270
15.0 153
16.0 270
17.0 141
18.0 259
19.0 282
20.0 235
root@18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller#

```

Figure 10: UDP-Testing-A

6.2.2 Case Study 2 - TCP and UDP for Existing Algorithm

- TCP Test between Host-1 and Host-2

To compare and test with another algorithm (Ryu-Multipath-Controller–using Dijkstra Algorithm)³ made the same evaluation with same test environment as described above in section 6.2.1 And we have calculated the time and bandwidth for the quantity of data sent. Thus, from the results average bandwidth throughput among the time interval i.e., 0.0 to 20.1 seconds sent from host-1 to host-2 is 64.1 Kbits/sec with 157 Kbytes packets transferred as shown in below Figure 11.

³<https://github.com/wildan2711/multipath/blob/master/ryu-multipath.py>

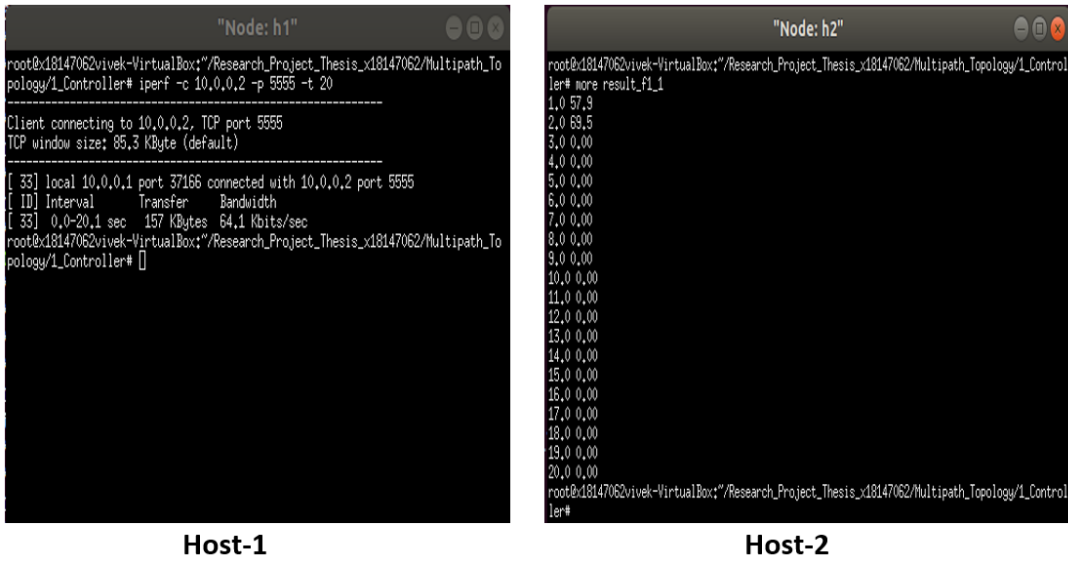


Figure 11: TCP-Testing-B

- **UDP Test between Host-1 and Host-2**

Similarly, as we did test for UDP for proposed algorithm we made the same evaluation with same test environment as described above in section 6.2.1 and we have captured the average bandwidth throughput among the time interval 0.0 to 20.3seconds sent from host-1 to host-2 as shown in Figure 12 is 992 Kbits/sec with 2.39 MBytes packets transferred. And also, after the 10 tries with a warning that 1708 datagrams of messages went with data-loss as shown in below Figure 12.

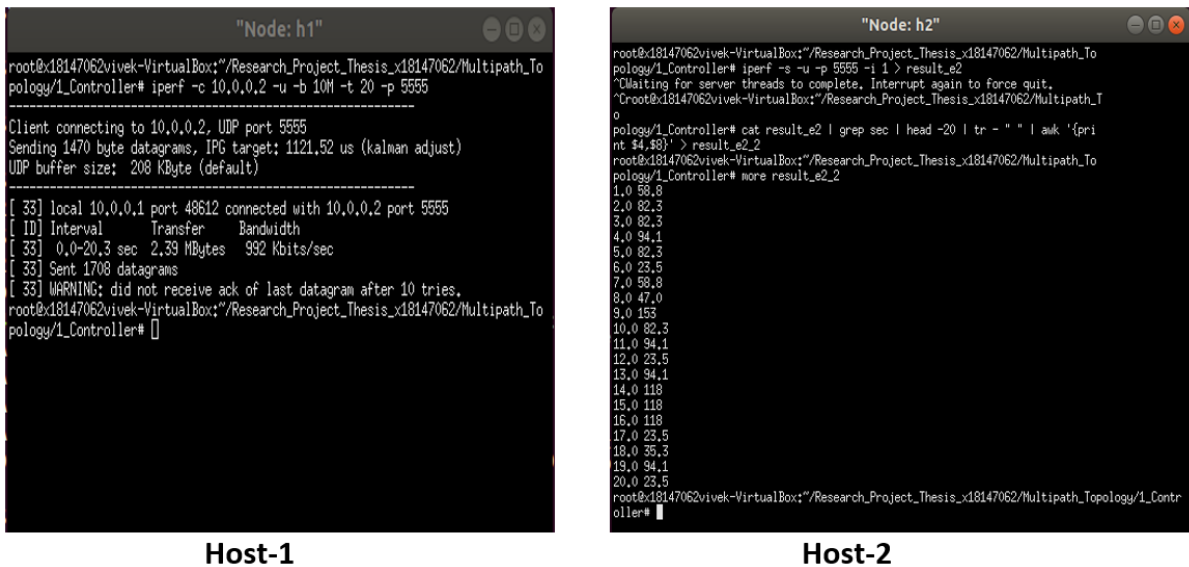


Figure 12: UDP-Testing-B

6.2.3 Case Study 3 - Evaluation of TCP and UDP

- **TCP and UDP Evaluation for Proposed Algorithm**

In the multipath network the performance from host-1 (h1) to host-2 (h2) of the packet transfer is reliant on the data-rate of packets transferred and average bandwidth of the data. The variation exists among the UDP bandwidth and TCP bandwidth and the quantity of data produced in a multipath network is calculated by TCP, whereas UDP will calculate the data rate transmission. Through the help of the gnuplot tool, graph is plotted between the time and throughput (bandwidth) i.e., x and y-axis respectively in host-2 TCP and Host-2 UDP as shown in the below Figure 13. And from the graphs we can say that bandwidth of TCP variates more than the UDP variates. The highest bandwidth for TCP is captured at 13.0 to 14.0 seconds interval of time with 324Kbits/sec. And at the UDP highest bandwidth is captured at 6.0 to 7.0 seconds interval of time with 282Kbits/sec.

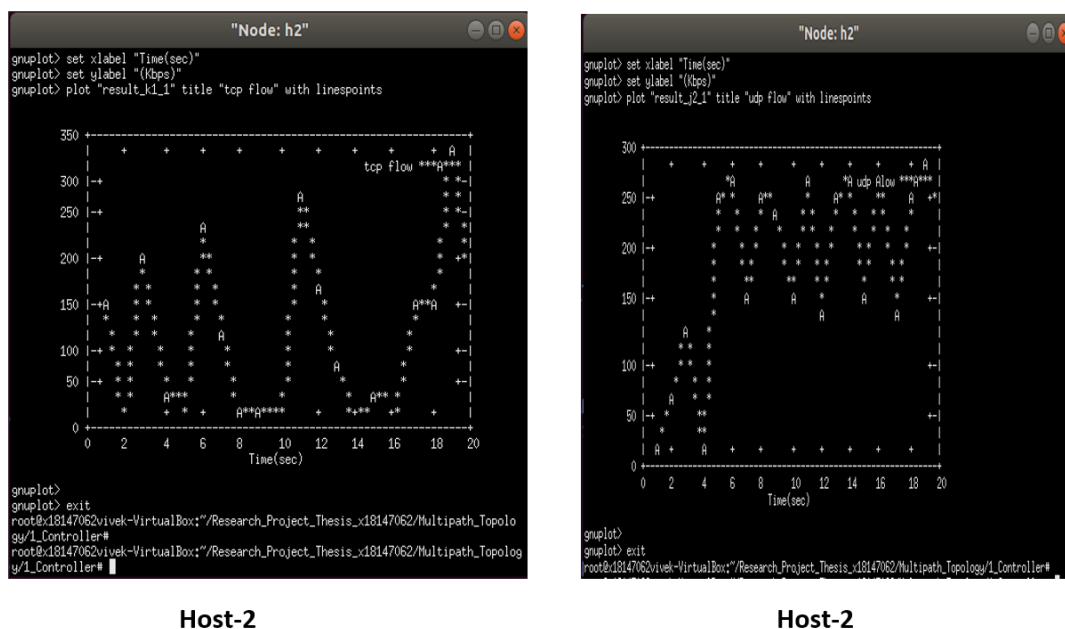


Figure 13: TCP-UDP-Evaluation-For-Proposed-Algorithm

- **TCP and UDP Evaluation for Existing Algorithm**

To compare and test with existing algorithm (Ryu-Multipath-Controller-using Dijkstra Algorithm) made the same evaluation with same test environment as described above. Through the help of the gnuplot tool, graph is plotted between the time and throughput (bandwidth) i.e., x and y-axis respectively in host-2 TCP and Host-2 UDP as shown in the below Figure 14. And from the graphs we can say that bandwidth of UDP variates more than the TCP variates. The highest bandwidth for TCP is captured at 1.0 to 2.0 seconds interval of time with 69.5Kbits/sec. And at the UDP highest bandwidth is captured at 8.0 to 9.0 seconds interval of time with 153Kbits/sec.

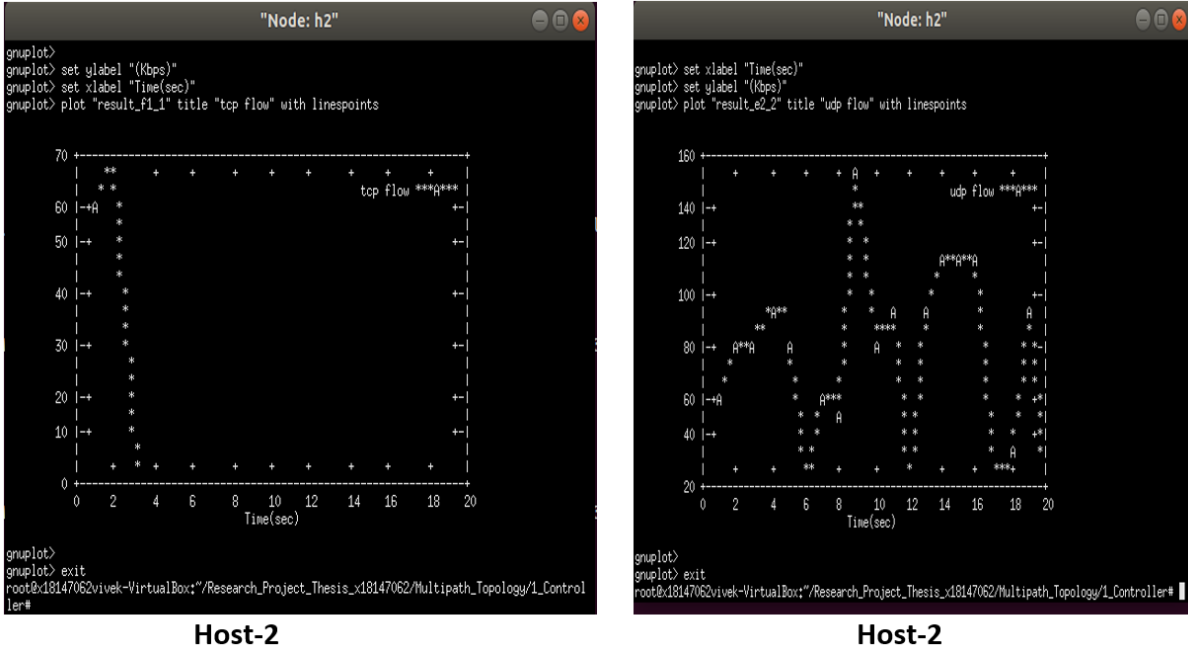


Figure 14: TCP-UDP-Evaluation-For-Existing-Algorithm

6.2.4 Case Study 4 - Evaluating the Congestion in Multipath Topology network

We have created a multipath topology (name of the python script: Multipath-Topology-22-links-1c) with extra links (Link-Connection-2) to validate the overcome of congestion in the multipath network as shown in below Figure 15. And we have executed the proposed Max-Flow controller on this multipath topology (script name: Multipath-Topology-22-links-1c) and validated that source (host-1) can reach destination (host-2) by avoiding the congestion in the network. To check this, we used pingall command as shown in Figure 16.

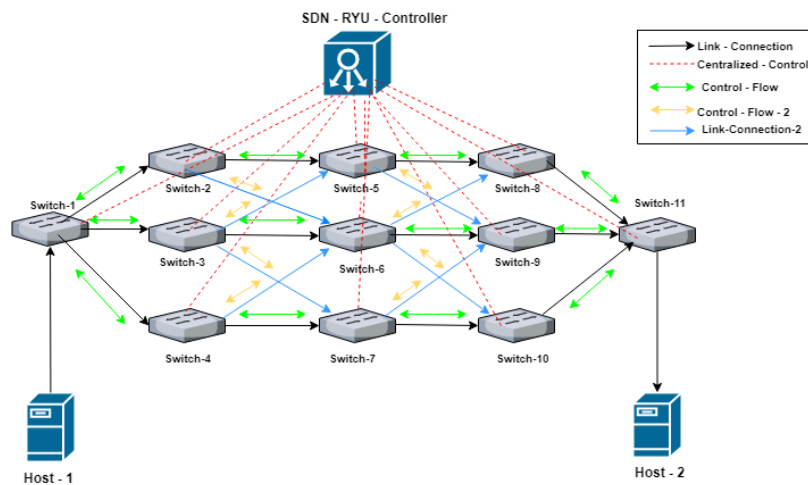


Figure 15: Congestion in Multipath Network

```

File Edit View Search Terminal Tabs Help
x18147062-vivek@x18147062vivek-VirtualBo... x18147062-vivek@x18147062vivek-VirtualBo... x18147062-vivek@x18147062vivek-VirtualBo... x18147062-vivek@x18147062vivek-VirtualBo...
x18147062-vivek@x18147062vivek-VirtualBox:~/Research_Project_Thesis_x18147062/Multipath_Topology/1_Controller$ sudo python Multipath_Topology_22_Links_IC.py
[Sudo] password for x18147062-vivek:
*** Adding switches
*** Adding hosts
*** Adding controllers
*** Adding links
(1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit)
(1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit)
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controller 1
*** Starting switches
(1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit)
(1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>

```

Figure 16: Evaluation of Congestion Multipath Network

6.3 Discussion

In this research project based on the case studies, experiments were performed. From Figure 17 represents the comparison of TCP throughput for multipath topology and it shows for proposed algorithm has higher throughput between the time interval 0.0 to 20.0 seconds sent from host-1 to host-2 is 232 Kbits/sec with 640 Kbytes packets transferred and for the existing algorithm when we compare to proposed algorithm has lesser throughput between the time interval 0.0 to 20.0 seconds sent from host-1 to host-2 is 64.1 Kbits/sec with 157 Kbytes packets. From Figure 18 represents the comparison of UDP throughput for multipath topology and proposed algorithm hasn't much significant difference from the existing algorithm, for proposed algorithm has little higher throughput between the time interval 0.0 to 20.0 seconds sent from host-1 to host-2 is 1000 Kbits/sec and for existing algorithm is 992 Kbits/sec, but it is same with 2.39 MBytes packets transferred. Form the case study 4, we can overcome of congestion in the multipath network. To conclude from case studies, we can say that utilization of bandwidth is more and can overcome the problem of network congestion. So, proposed algorithm is better when we compared to existing algorithm.

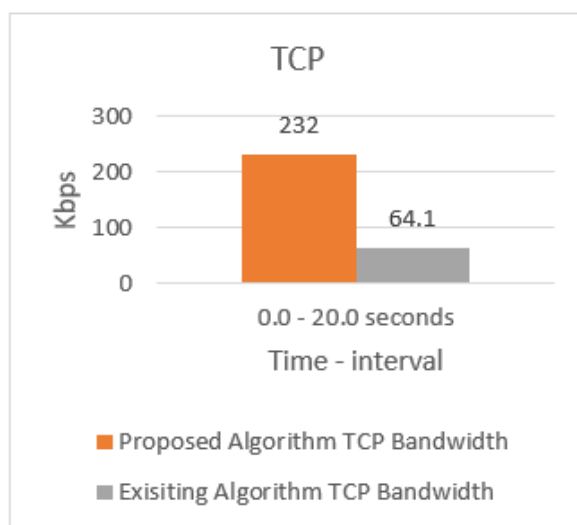


Figure 17: TCP

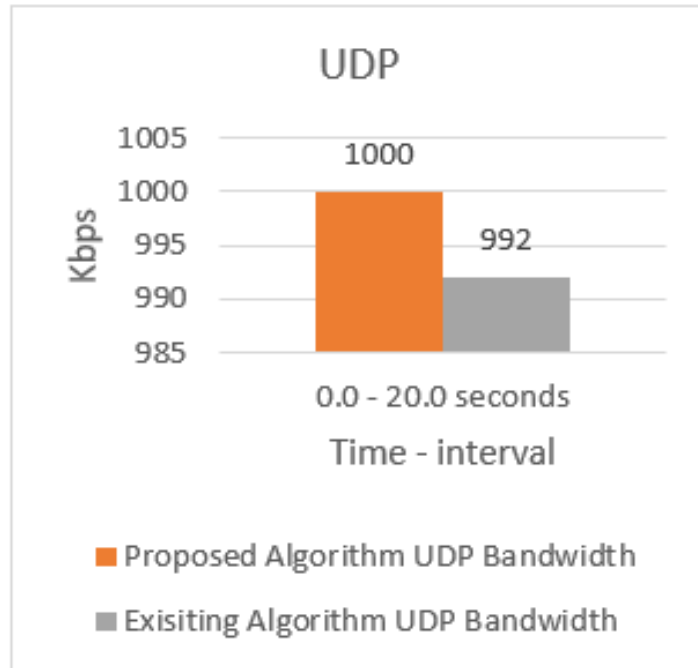


Figure 18: UDP

7 Conclusion and Future Work

The present network situation needs a higher bandwidth usage and solve the network congestion problem, because the traditional one is complex with the distributed architecture. To overcome this, SDN is practiced in the current infrastructure. From the case studies we get to know that TCP throughput shows significant difference between the proposed and existing algorithm and in case of UDP throughput, there wasn't much significant difference but proposed algorithm is performing better. Thus, the proposed RYU-controller algorithm performs better than the existing algorithm. Service Function Chaining can be applied in the future enhancement to meet further requirements like load balancing. And also, we can apply virtual network functions to enhance the resource storage utilization.

Acknowledgement

I would like to express my sincere gratitude to my mentor, Dr. Muhammad Iqbal for his valuable guidance and support for my research project. I would also thank our programme director Dr. Horacio Gonzalez-Velez for continuous support through out the course. And I would also thank my parents and friends for the continuous support during the course.

References

Ai, J., Chen, H., Guo, Z., Cheng, G. and Baker, T. (2019). Improving resiliency of software-defined networks with network coding-based multipath routing, *2019 IEEE*

Symposium on Computers and Communications, ISCC 2019, Barcelona, Spain, June 29 - July 3, 2019, IEEE, pp. 1–6.

- Benson, T., Anand, A., Akella, A. and Zhang, M. (2011). Microte: fine grained traffic engineering for data centers, *in* K. Cho and M. Crovella (eds), *Proceedings of the 2011 Conference on Emerging Networking Experiments and Technologies, Co-NEXT '11, Tokyo, Japan, December 6-9, 2011*, ACM, p. 8.
- Bera, S., Misra, S. and Jamalipour, A. (2019). Flowstat: Adaptive flow-rule placement for per-flow statistics in SDN, *IEEE J. Sel. Areas Commun.* **37**(3): 530–539.
- Guillen, L., Izumi, S., Abe, T., Sukanuma, T. and Muraoka, H. (2017). SDN implementation of multipath discovery to improve network performance in distributed storage systems, *13th International Conference on Network and Service Management, CNSM 2017, Tokyo, Japan, November 26-30, 2017*, IEEE Computer Society, pp. 1–4.
- Guillen, L., Izumi, S., Abe, T., Sukanuma, T. and Muraoka, H. (2018). Sdn-based hybrid server and link load balancing in multipath distributed storage systems, *2018 IEEE/IFIP Network Operations and Management Symposium, NOMS 2018, Taipei, Taiwan, April 23-27, 2018*, IEEE, pp. 1–6.
- Hu, Y., Wang, W., Gong, X., Que, X., Ma, Y. and Cheng, S. (2015). Maximizing network utilization in hybrid software-defined networks, *2015 IEEE Global Communications Conference, GLOBECOM 2015, San Diego, CA, USA, December 6-10, 2015*, IEEE, pp. 1–6.
- Lin, S., Wang, P., Akyildiz, I. F. and Luo, M. (2018). Towards optimal network planning for software-defined networks, *IEEE Trans. Mob. Comput.* **17**(12): 2953–2967.
- Liu, J., Li, J., Shou, G., Hu, Y., Guo, Z. and Dai, W. (2014). SDN based load balancing mechanism for elephant flow in data center networks, *2014 International Symposium on Wireless Personal Multimedia Communications, WPMC 2014, Sydney, Australia, September 7-10, 2014*, IEEE, pp. 486–490.
- Liu, Z., Wang, Q. and Lee, J. (2019). A SDN controller enabled architecture for the IMS, *20th Asia-Pacific Network Operations and Management Symposium, APNOMS 2019, Matsue, Japan, September 18-20, 2019*, IEEE, pp. 1–4.
- Prabhavat, S., Nishiyama, H., Ansari, N. and Kato, N. (2012). On load distribution over multipath networks, *IEEE Commun. Surv. Tutorials* **14**(3): 662–680.
- Rezende, P. H. A., Kianpisheh, S., Glitho, R. H. and Madeira, E. R. M. (2019). An sdn-based framework for routing multi-streams transport traffic over multipath networks, *2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019*, IEEE, pp. 1–6.
- Sehery, W. and Clancy, C. (2017). Flow optimization in data centers with clos networks in support of cloud applications, *IEEE Trans. Network and Service Management* **14**(4): 847–859.

- Sheu, J., Liu, L., Jagadeesha, R. B. and Chang, Y. (2016). An efficient multipath routing algorithm for multipath TCP in software-defined networks, *European Conference on Networks and Communications, EuCNC 2016, Athens, Greece, June 27-30, 2016*, IEEE, pp. 371–376.
- Singh, S. K., Das, T. and Jukan, A. (2015). A survey on internet multipath routing and provisioning, *IEEE Commun. Surv. Tutorials* **17**(4): 2157–2175.
- Subedi, T. N., Nguyen, K. K. and Cheriet, M. (2015). Openflow-based in-network layer-2 adaptive multipath aggregation in data centers, *Comput. Commun.* **61**: 58–69.
- Woo, S., Lee, S., Kim, J. and Shin, S. (2018). RE-CHECKER: towards secure restful service in software-defined networking, *IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2018, Verona, Italy, November 27-29, 2018*, IEEE, pp. 1–5.
- Xu, C., Jin, W., Zhao, G., Tianfield, H., Yu, S. and Qu, Y. (2017). A novel multipath-transmission supported software defined wireless network architecture, *IEEE Access* **5**: 2111–2125.
- Yao, X., Wang, H., Gao, C. and Yi, S. (2016). Maximizing network utilization for SDN based on particle swarm optimization, *28th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2016, San Jose, CA, USA, November 6-8, 2016*, IEEE Computer Society, pp. 921–925.
- Yoshioka, K., Hirata, K. and Yamamoto, M. (2017). Routing method with flow entry aggregation for software-defined networking, *2017 International Conference on Information Networking, ICOIN 2017, Da Nang, Vietnam, January 11-13, 2017*, IEEE, pp. 157–162.