STATISTICAL INFERENCE BASED LOAD BALANCED ROUTING IN SOFTWARE DEFINED NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

SEMİH KAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2020

Approval of the thesis:

STATISTICAL INFERENCE BASED LOAD BALANCED ROUTING IN SOFTWARE DEFINED NETWORKS

Submitted by Semih Kaya in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics**

Prof. Dr. Sevgi Özkan Yıldırım
Head of Department, **Information Systems**

Assoc. Prof. Dr. Altan Koçyiğit
Supervisor, **Information Systems, METU**

Examining Committee Members:

Assoc. Prof. Dr. Erhan Eren
Information Systems, METU

Assoc. Prof. Dr. Altan Koçyiğit
Information Systems, METU

Assoc. Prof. Dr. Aysu Betin Can
Information Systems, METU

Assoc. Prof. Dr. Enver Ever
Computer Engineering, METU-NCC

Assist. Prof. Dr. Serhat Peker
Management Information Systems, İzmir Bakırçay University

Date: _*24/09/2020*

Name, Last name :	**Semih Kaya**
Signature :	

ABSTRACT

STATISTICAL INFERENCE BASED LOAD BALANCED ROUTING IN SOFTWARE DEFINED NETWORKS

Kaya, Semih
MSc., Department of Information Systems
Supervisor: Assoc. Prof. Dr. Altan Koçyiğit

September 2020, 55 pages

Networks have been the main method of transferring data for more than forty years. The traffic volumes and sizes of networks have increased considerably in the last two decades. The traditional methods used in the networks to transfer data become inefficient due to this growth. Therefore, network planning and smart delivery methods have gained importance. Accordingly, traffic engineering methods are deployed to meet the faster and more efficient delivery requirements. These methods have been proven beneficial and they are still being used on every level of networking. Recently, software defined networking redefined the architecture of networks and network devices. This new architecture paved the way for more flexible network and traffic management techniques.

In this thesis, we propose a new routing method, which minimizes the maximum link utilization in the software-defined networks. The proposed method defines a new cost metric based on statistical inference to distribute load evenly in the network. The method is demonstrated, and its performance is evaluated on virtual software defined network topologies under various artificial network loads. The experiments show that the proposed algorithm achieves the even distribution of traffic and minimizes the maximum link utilization in software defined networks.

Keywords: Software Defined Networks, Routing, Traffic Engineering, Minimization of Maximum Link Utilization, Statistical Inference

ÖZ

YAZILIM TABANLI AĞLARDA İSTATİSTİKSEL ÇIKARIM TEMELLİ
YÜK DAĞILIMLI YÖNLENDİRME

Kaya, Semih
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Doç. Dr. Altan Koçyiğit

Eylül 2020, 55 sayfa

Bilgisayar ağları veri, kırk yıldan fazla bir süredir, veri aktarımının ana yöntemi olmuştur. Son yirmi yıl içerisinde trafik hacmi ağların büyüklükleri kayda değer şekilde artmıştır. Bu büyümeden kaynaklı olarak ağlarda veri iletiminde kullanılan geleneksel yöntemler yetersiz kalmıştır. Böylece ağ planlama ve akıllı yönlendirme yöntemleri önem kazanmıştır. Bu doğrultuda daha hızlı ve etkin veri iletimi gereksinimlerini karşılamak için trafik mühendisliği yöntemleri konuşlandırılmaktadır. Bu yöntemlerin faydaları ispat edilmiştir ve bilgisayar ağlarının her seviyesinde hala kullanılmaktadırlar. Yakın geçmişte, yazılım tanımlı ağlar ağların ve ağ donanımlarının mimarisini yeniden tanımlamıştır. Bu yeni mimari daha esnek bir ağ ve trafik yönetimi için yeni yollar açmıştır.

Bu tezde, yazılım tanımlı ağlarda en yüksek kullanımı asgari düzeye indiren yeni bir yönlendirme yöntem öneriyoruz. Önerilen yöntem, ağ içinde trafiği eşit dağıtmak için istatistiksel çıkarım üzerine kurulu bir hat maliyeti ölçütü tanımlamaktadır. Farklı yapay ağ yükleri altındaki sanal bir yazılım tanımlı ağ topolojisi üzerinde yöntem gösterilmiş ve başarımı değerlendirilmiştir. Gerçekleştirilen deneyler, önerilen yöntemin yazılım tanımlı ağlar üzerinde düzgün trafik dağılımı sağladığını ve en yüksek bağlantı kullanımını asgari seviyeye indirdiğini göstermektedir.

Anahtar Sözcükler: Yazılım Tanımlı Ağ, Yönlendirme, Trafik Mühendisliği, En Yüksek Bağlantı Kullanımının Asgari Düzeye İndirgenmesi, İstatistiksel Çıkarım

DEDICATION

To My Wife and Our Beautiful Children

ACKNOWLEDGMENTS

First, I would like to express my gratitude to my supervisor Assoc. Prof. Dr. Altan Koçyiğit, for his guidance through my Master’s program. His positive attitude, tolerance, and expertise has been the most important enabler of this work.

Besides my supervisor, I would like to thank my company ASELSAN A.Ş and my managers Hüsnü Coşkun, Mehmet Uçak and Tanın Afacan for their continuous support.

Finally, I would like to thank to my wife, the love of my life, for the tolerance and support during my absence from home.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

5G	The Fifth Generation Mobile Network
AP	Access Point
API	Application Programming Interface
ARM	Adaptive Route Modification
ARP	Address Resolution Protocol
CSPF	Constraint Based Shortest Path First
DIP	Dynamic Information Polling
DRILL	Distributed Randomized In-network Localized Load-balancing
ECMP	Equal Cost Multi Path
ESF	Equally Split Flow
IETF	Internet Engineering Task Force
IMAB	Inverse Maximum Available Bandwidth
LAT	Load Allocation Table
LTE	Long Term Evolution
MCP	Minimal Impact Congestion Control algorithm
MH	Minimum Hop
MLU	Maximum Link Utilization
MPLS	Multi-Protocol Label Switching
RSVP	Resource Reservation Protocol
S2SP	switch to switch path table
SCH	Switch Health Check
SDN	Software Defined Networking
SILBR	Statistical Inference based Load Balanced Routing
SILBR	Statistical Inference Based Routing
SR	Segment Routing
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
W	Window

CHAPTER 1

INTRODUCTION

Traditional networks have primarily utilized per hop routing [1] in order to transfer data through the best paths. As the traffic types and volume increase tremendously, per hop routing turns out to be inefficient since, for instance, it over utilizes the best path, while non-best paths are left under-utilized. This inefficiency has led to the development of traffic engineering. The methods used in traffic engineering evolved from offline methods to more dynamic approaches such as resource reservation protocol, segment routing and constraint based shortest path routing. The common point of these methods is optimizing the performance of a network by dynamically monitoring, predicting and regulating traffic. They essentially properly distribute traffic over the network and avoid congestion on the network's links by efficient utilization of the available network resources in the network.

The offline methods in traffic engineering require knowing traffic demand and topology information in advance to find the optimal distribution of traffic over a network. Such methods usually presume that the network and the traffic demand is static or change very slowly in time. However, contemporary networks are very large and highly dynamic, thereby rendering such methods unpractical. On the other hand, dynamic approaches, such as the Resource Reservation Protocol (RSVP) and Constraint Based Shortest Path First (CSPF) [1], aim to automate the resource planning of network devices. The operational process, however, depends on each router to provide the information about its available resources to other routers in the network when a new flow is to be routed. Thus, such methods are inherently distributed and require frequent exchange of resource information between the routers. Also, in order to properly distribute the traffic, continuous monitoring and short/long term estimation of the traffic demands are required. These monitoring and demand estimations necessitate the use of advanced routers with high-end processing capabilities. Similarly, segment routing (SR) [2] aims to automate the resource reservation problems in the network. Segment routing combines traffic engineering with source-based routing. The SR can leverage distributed or centralized control structures. SR traffic engineering can only be applied to IPv6 or MPLS networks.

The recently introduced Software Defined Networking (SDN) technology has the potential to transform traditional data networks to open and flexible architectures. Basically, SDN decouples control plane which makes routing decisions and data plane which performs packet forwarding. Hence, it makes network control programmable by centralizing network state and intelligence. With SDN, application and control layers are packed closely [3] and communication with the data layer becomes simpler compared to traditional networks. While the traffic engineering methods suggested for traditional networks are also applicable to SDN, dynamic behavior of the management plane enables different approaches [4] for traffic engineering. The focus of these approaches can be summarized as traffic steering, traffic load balancing, traffic analysis, fault tolerance and concurrent management.

SDN can provide dynamic management of flow tables [5]. Therefore, routing can be made dynamic without the need for information exchange between switching devices in the data plane. This reduces the complexity and adds agility to the management process of SDN based networks and allows an SDN controller to steer the traffic more dynamically than traditional networks.

The flow rules can mark traffic of interest in a range from physical port level to transport layer protocols [3] and allows for fine grained traffic control. Moreover, SDN can provide up-to-date traffic information [6] about the network in the control plane. This enables applications to easily access and consume statistical information about the network and modify the state of network according to the changes in traffic demand.

The load balancing mechanisms are deployed widely in networks for services provided over networks in order to prevent resource overwhelming. SDN based load balancing solutions are deployed over a wide range of networks [7] such as 5G, LTE, Data Center, radio access. Since SDN based solutions provide a global view of the network, these solutions can improve overall system performance. Availability of real time statistical information and agile management allow the development of novel approaches in load balancing. Both deterministic[4], [8], [9] and non-deterministic [10] methods are being researched for SDN traffic engineering.

In traditional networks, a routing mechanism is limited by its software implementation and predefined metrics[11] [12]. In order to carry out statistical calculations and incorporate inferences into routing decisions, either a complex signaling protocol along with the routing should be used as in the example of RSVP, or an external controller should be deployed to constantly monitor and configure the underlying network. In the latter, SDN facilitate collecting and involving statistical indicators in the routing decisions and dynamically managing the routing in order to obtain a balanced traffic distribution.

In this thesis, we focus on providing a heuristic load balanced routing scheme. We propose a method, namely Statistical Inference Based Load Balanced Routing (SILBR) that utilizes statistical inference in routing decisions. Similar to [9] and [13], we strive for the distribution of flows in the network in a way that creates a balanced traffic distribution over the network. We introduce a link cost metric that is calculated by the current link usage information collected by means of centralized global view of SDN. SDN also supports the agile management of traffic by allowing fast transformation of flow tables, which can be used like a routing table of traditional routing. The work in [14] shows an example of route management by manipulating the flow table rules.

The primary objective of the routing algorithm proposed is to minimize the maximum link utilization. Hence, the network can be used to carry more traffic and lower delay experienced by the packets by avoiding congestion in the links. The beneficial side effect of such a traffic management method is the optimal utilization of the underlying network.

The primary contribution of this thesis is the SILBR algorithm proposed. SILBR introduces a dynamic link cost metric that can be used in routing to achieve an even traffic distribution over a network. The proposed routing approach makes use of statistical analysis of recent link utilization measurements to minimize the maximum link utilization by dynamic link cost adjustments.

The rest of the thesis is organized as follows. In chapter 2, the key concepts used to realize SILBR are introduced. In chapter 3, we present a review of the literature regarding traffic management and engineering in software-defined networks. In chapter 4, we present our routing algorithm and details of proposed traffic management method. In chapter 5, we present the details of our implementation, testing environment and test results. In chapter 6, we conclude our study and suggest directions for the relevant future work for improvements.

CHAPTER 2

BACKGROUND

This chapter explains the necessary concepts about traffic engineering and software defined networks that we used in this study.

2.1. Traffic Engineering

The traffic-engineering addresses the problem of sharing limited sources of the network according to demands of the traffic. Since the traffic demands and the available resources are constantly changing there is no obvious solution to this problem. Internet Engineering Task Force (IETF) defines the traffic-engineering [15] as "encompassing the application of technology and scientific principles to the measurement, modeling, characterization, and control of Internet traffic, and the application of such knowledge and techniques to achieve specific performance objectives". As seen from the definition, traffic engineering requires measurement and control of traffic in order to optimize the resource utilization.

Performance objectives of traffic-engineering can be traffic oriented or resource oriented. Traffic oriented traffic engineering aims to increase the service quality by minimization of packet loss, minimization of delay, maximization of throughput, and enforcement of service level agreements. Resource oriented traffic engineering aims to adjust the resource utilization so that the degradation in service quality can be avoided. The resource limitations can be the hardware related or service provider related such as bandwidth.

Best-effort Internet traffic is an example of traffic-oriented traffic-engineering and congestion avoidance is an example of resource-oriented traffic-engineering. While the former is more concerned with classification of traffic in order to apply the correct policies, the latter requires measurement of resources and distribution of load among available resources.

The control scheme can be reactive or proactive in traffic engineering. Traffic engineering actions are listed in [15] as:

- Modification of traffic management parameters

- Modification of routing parameters
- Modification of attributes and constraints associated with resources

As stated in [1], [15] and [16] control mechanisms of current routing protocols are not adequate for Traffic engineering; moreover, the shortest path algorithms significantly increase the congestion problems. The overlay network models extend the design space for networks by enabling virtual topologies atop physical network topology and provides important services to support traffic engineering such as constraint-based routing, traffic shaping, traffic policing, path compression and virtual paths.

Each router in the network floods information about its performance characteristics, connected links and administrative policy using the extensions to the intra-autonomous system routing protocols such as OSPF [17] and IS-IS [18].

Constraint Based Shortest Path First (CSPF) aims to combine additional metric parameters with the shortest path routing. Constraints for the routing can be bandwidth requirements, hop limitations, administrative groups (link colors), priority (setup and hold), explicit route link attributes and available free bandwidths of the links. The CSPF computation is based on graph pruning, however it requires hop by hop calculation [16].

MPLS uses separation of control and forwarding planes [19] which allows it to base routing decisions other than shortest path. MPLS traffic-engineering mechanism is based on distributing the resource information, constraint-based routing and RSVP based signaling for pseudo wire tunneling the traffic. MPLS pseudo wire tunnels can be created dynamically and label switching paths can be calculated in a distributed fashion or can be determined by a centralized function in an offline manner. MPLS-TE uses constraint-based routing for tunnel creation and label switching path formation and the routing decision is delivered to the network as an explicit route object.

Explicit route objects are delivered to RSVP [20] [15] in order to handle the signaling in a hop by hop manner. RSVP is designed to provide resource reservation for IP protocols using the integrated services architecture [21]. RSVP reserves resources for flows. A flow can be a destination IP address, protocol identifier and destination port or a label switching path. The RSVP protocol explicitly signals each node for resource reservation requirement from source to the destination. Assuming the RSVP messages are valid, each router along the path is responsible for generating the RSVP message for the next hop along the path and passing the message downstream. If the necessary resources can be reserved for the flow, actual transmission of data starts.

2.2. Software Defined Networking

Traditional networks operate in a distributed structure and vertically integrated manner [3]. This structure increases complexities and obstructs the management. The software defined networking is the structure that aims to overcome these hindrances of traditional networking by separation of control and data planes. The separated structure is realized as control plane and forwarding plane where a single control plane controls several devices in the forwarding plane[22]. While separated structure solves the problem of management, it does not change the operational principles of networking. The solution for orchestration of operation is developed as open interfaces and programmable network. In summary, SDN is the collection of open, programmable interfaces deployed in physically separated control and data planes [23].

SDN architecture comprises [3] Forwarding Devices, Data Plane, South Bound Interface, Control Plane, North Bound Interface, Management Plane. According to this separation, forwarding devices are the hardware or software devices that perform a set of basic operations, data plane is the representation of interconnections for forwarding devices, southbound interfaces are the open API's that enable communication with forwarding devices, control plane is where all the control logic for the network rests, northbound interface is the abstraction of southbound interface and the management plane is where all the network functions operate. The enabler of this structure is the soutbound API's used in the network. ForCES [24], OpenFlow [25], and POF [26] are examples of these interfaces. OpenFlow is a very frequently used interface and we will examine it in the next section

2.2.1. OpenFlow

OpenFlow is one of the southbound specifications defined for SDN. OpenFlow specification set is maintained by Open Networking Foundation (ONF) [27]. The abstract packet processing machine of the OpenFlow protocol stack is called switch. An OpenFlow enabled switch is divided into three parts flow table, secure channel and OpenFlow protocol [25]. The flow table holds the rules and actions related to packet processing, secure channel enables communication between the controller and the switch and the OpenFlow protocol enables controller to define and modify flow entries in the switch flow tables.

OpenFlow Protocol Stack [28] is composed of message layer, state machine, system interface, configuration and data model. Message layer defines structure and semantics for all messages. State machine defines the low-level protocol behaviors like negotiation, capability discover, flow control and delivery. System interface defines communication method to be used in information exchange with the outside world such as TCP or TLS. Configuration is the initial state or default values of variables in the protocol stack. Data model is the abstract information that describes the process capabilities, configuration state and current statistics of an OpenFlow switch.

OpenFlow data model, illustrated in Figure 1 describes the relationship of entities in the protocol stack of OpenFlow. The entity relations also describe the decision process of an OpenFlow switch.

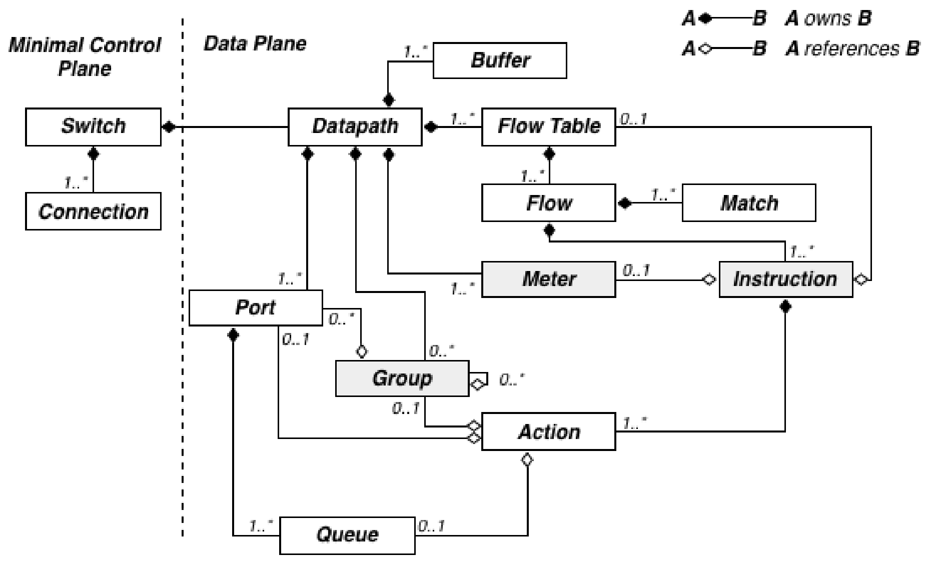


Figure 1 OpenFlow Pipeline [29]

Each flow that enters an OpenFlow switch is matched against the entries in flow tables. A general structure of flow entry is illustrated in the Figure 2 as presented in [30].

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Figure 2 OpenFlow Match Fields [30]

If a match is found in flow entries, related instructions are carried out by the switch. The actions can be Apply-Actions, Clear-Actions,Write-Actions,Write-Metadata,Stat-Trigger or Goto-Table.

The OpenFlow control channel connects each switch to a controller. This interface allows controller to configure switches and switches can send asynchronous event messages to a controller. The event messages can be related to packet arrivals or departures, port status changes or flow table changes. In this thesis, we are interested in new flow arrivals to the network. A new flow can be detected by table miss events caused by the first packet of the flow entering a switch in the network. When a table miss event happens, switches can be configured to deliver the control of the packet to the controller through the control channel. The PACKET_IN message is sent from the switch to the controller along with partial or full packet data. The PACKET_IN message contains important information like DataPathID of the switch sending the packet, packet header, ID of the table that was looked up, reasons packet is sent to controller. The packet in messages can be used to detect the flows entering the network and by following PACKET_IN messages controller can run applications that can dynamically change the behavior of network.

2.3. Statistical Inference

The normal distribution is a well-known and natural distribution that occurs frequently. According to study in [31], the network traffic distribution models and analysis based on normal distributions are valid. The normal distribution can be standardized using the z distribution. The z distribution and the standard z-value gives us the measure of the difference between the raw sample mean and the population mean. The formula for calculating a z value is given by

$$z = \frac{X-\mu}{\sigma} \tag{1}$$

where:

X is the raw measurement.

μ is the population mean.

σ is the population standard deviation.

The z-score enables to use a standard view of the normal distribution. Figure 3 taken from [32] shows empirical 68-95-99.7 rule of normal distribution and its relation to the Z and T Scores.

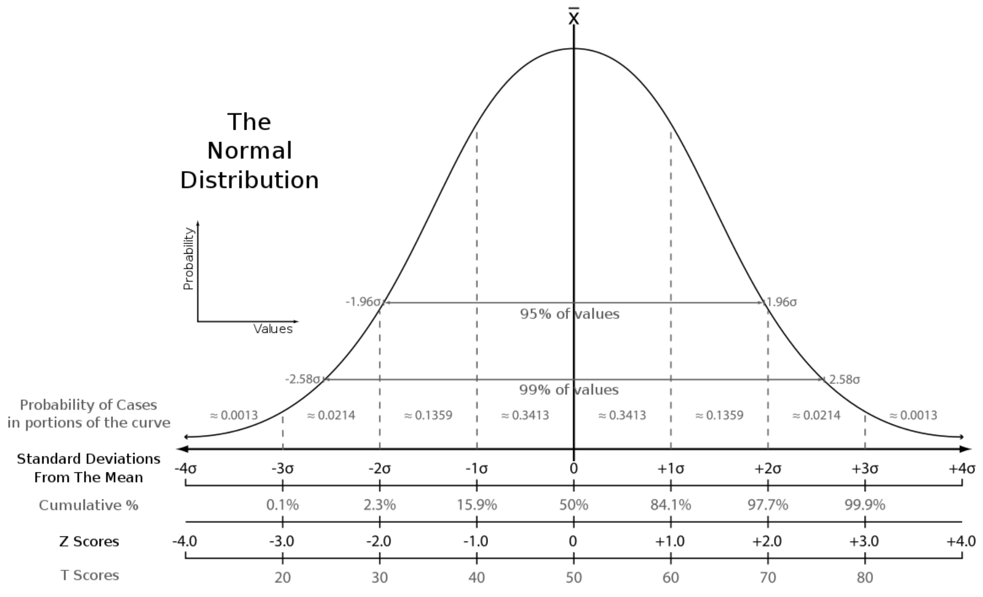


Figure 3 Normal Distribution and Scales [32]

In statistical analysis the z distribution can be used for predictive inference of the population mean when the population variance of measured variable is known.

By the central limit theorem, we know that sample means are approximately normally distributed. Hence, the formula for z distribution can be used to relate sample statistics to population statistics. The equation (2) and (3) shows this derivation.

$$z = \frac{\bar{x}-\mu}{\sigma/\sqrt{n}} \tag{2}$$

where $\bar{x}$is the sample mean and n is the sample size.

Solving the eq (2) for μ:

$$\mu = \bar{x} - \frac{z*\sigma}{\sqrt{n}} \tag{3}$$

The sample mean can be lower or higher than the population mean. The corrected version of equation (3) is given in equation (4).

$$\mu = \bar{x} \pm \frac{z*\sigma}{\sqrt{n}} \tag{4}$$

Thus, with a given confidence interval of α the estimate of population mean can be calculated using the Z table as given in equation (5):

$$\bar{x} - z * \frac{\sigma}{\sqrt{n}} < \mu < \bar{x} + z * \frac{\sigma}{\sqrt{n}} \tag{5}$$

When the population variance is unknown the Student's t distribution can be used to estimate the population mean [31]. Similar to z distribution, the t distribution requires the population to be normally distributed.

The formula for t value is given in equation number (6):

$$t = \frac{\bar{x}-\mu}{\frac{s}{\sqrt{n}}} \tag{6}$$

where s is the sample standard deviation and n is the sample size and n-1 is the degree of freedom.

The t table is created from many different t distributions, and each line shows a different behavior. The degree of freedom, defined as "the number of independent observations for a source of variation minus the number of independent parameters estimated in computing the variation" [31], is used to differentiate between the lines of the t table. Similar to z distribution the t distribution can be used to estimate the mean value of a population. The formula for this is given in equation (7).

$$\bar{x} - t_{n-1} * \frac{s}{\sqrt{n}} < \mu < \bar{x} + t_{n-1} * \frac{s}{\sqrt{n}}, df = n - 1 \tag{7}$$

CHAPTER 3

RELATED WORK

There are many methods to achieve a uniform traffic distribution on networks. In this chapter, we review examples for such methods. Some of the work we present here aims to distribute the network traffic with micro level decisions that are taken in the operating systems of switching hardware and the others apply a macro level management by combining the high level network view of SDN with flow rule manipulation.

Ghorbani et al. [33] present an example of micro level load distributions. Their work is centered on Clos network topologies. A Clos network is a commonly used data center layout which is composed of spine and leaf nodes. Although, such layouts contain large amount of diversity, due to the shortcomings of shortest path and Equal Cost Multi Path (ECMP) routing these networks suffer heavily from short burst congestions. Ghorbani et al. [33] proposed DRILL (Distributed Randomized In-network Localized Load-balancing) algorithm to micromanage flow traffic in packet level and achieve and estimate Equally Split Flow (ESF) load distribution using a switch local algorithm. DRILL assumes that for a given destination, a set of least cost routes are installed into each switch's forwarding table by the routing protocol. Then the algorithm starts leveraging the outgoing buffers' loading levels and selects the shortest queue for each packet and updates memory slots with the identities of selected buffers. As the performance metric, they use the standard deviation of uplink queues in the spine switches. Their work can outperform the state-of-the-art load balancing mechanisms. Their measurements show that as the amount of memory and choice is increased, the performance of the algorithm increases; however, excessive increase in memory and choices decreases the effectiveness of the algorithm and the performance of the switches. The contribution of this work is that, by micromanaging and moving packets to the shortest queues, load balancing can be achieved in data center networks. The load balancing achieved this way can also be a solution to the large number packet losses that are caused by microbursts. However, due to the fact that a small amount of choices should be given on each switch, the amount of data that can be processed in the micromanagement scale is limited due to the algorithm's time constraints. Also, the algorithm is constrained in the number of paths that can be chosen. The main problem of packet level load balancing, as the writers have discussed in their work, is the reordering of the packets. The paper illustrates the need for reordering and claims that the TCP is not affected by this reordering, no mention of UDP traffic degradation or loss of quality in real time traffics is given.

Karuna [34] is another example of micromanagement for load balancing in data center networks. The paper focuses on flow completion times and the aim is to minimize the deadline meet rate, minimize the average flow completion time and also prevent the reverse effects that are caused by such algorithms on the flows that doesn't have a deadline requirement such as earliest deadline first. The basic approach is to reserve the minimum amount of bandwidth to complete a flow barely before its deadline and leave the maximum possible bandwidth for the remaining flows. The flows are classified into three categories in accordance with their completion time requirements: flows with deadlines, flows without deadlines but known sizes, and flows without deadlines or known sizes. In order to handle the deadline requirements, they propose a Minimal Impact Congestion Control algorithm (MCP) that utilizes the explicit congestion control available in the TCP protocol stack. Their approach is superior to similar preceding work in that the MCP is a non-greedy algorithm in terms of bandwidth. MCP places the flows with deadlines into highest priority queues and their rates are throttled in order to just meet the deadline. In order to measure the impact of such prioritization on other flows, they measure the long-term time averaged per packet delay and try to minimize this value. In order to sustain the queue stability, the Lyapunov optimization framework with drift-plus-penalty method is applied to the network. The link model used in the paper is an M/M/1 queue, by using the size of the last window and predefined queue and congestion threshold an early flow termination mechanism is deployed in order to prevent unnecessary bandwidth preservation for flows that cannot complete in time. The tests are carried out on a small size data center network model. The testbed network is composed of a single switch and 16 servers. The results show that Karuna reserves just enough bandwidth to complete flows in time and achieves almost zero deadline miss rate. The study presented in [34] does not create any starvation problem that is commonly experienced by shortest job first or strict priority queueing algorithms. Although their algorithm is successful in terms of increasing the number of flows meeting their deadline criteria, proposed mechanism works as a kernel module on the host machines and acts as a shim layer between network interface and TCP/IP protocols. Packet level queueing and manipulation also requires packet reordering.

LABERIO [9] aims to achieve dynamic load balancing in data center networks with a path switching algorithm. The basic idea is to deploy a halfway switching strategy which switches the path of a flow in the midway in order to better utilize the agility of OpenFlow. The goal of load balancing is to improve the network throughput. The quality of service parameter is defined and measured as the minimum bandwidth requirement, packet reordering due to path switch is left to end hosts and end hosts are required to set a priority weight to their flows. LABERIO divides the problem into two stages such as end host scheduling and load balanced routing. The end host scheduling part is simply weighting the flows by the hosts and application of Largest Weight First

Served algorithm in the switches. For the load balanced routing part, two important tables, namely switch to switch path table (S2SPT) and load allocation table (LAT), are created and maintained in the controller. S2STP provides all the path information and in the LAT, the remaining bandwidth of every link along a path is maintained. The load balance detector parameter is used as the indicator for starting path switching. The load balance detector parameter is defined as the absolute gap between the average load of the network and real time load on each link. The metric used is very similar to variance except that instead of using the links' mean bandwidth occupancy value, overall average bandwidth occupancy is used. When a flow request is received by the controller, the algorithm starts calculating an initial path. When the load balance parameter on a link exceeds the predefined threshold value, the algorithm starts path switching by finding a substitute path for the flows on that link and selecting the least loaded one. The authors report the problem of this approach as the frequent switching of some flows, which causes the heavy increase in total number of hops. They revise their algorithm to into account the number of switching actions taken on a flow and the number of hops increase on the path for that flow. They further refine the algorithm by measuring the bandwidth requirement in the flow level. LABERIO is highly criticized for its frequent path switching problem in the preceding works on the subject. Although the authors do not specify the reason for this behavior, we suspect the reason is the usage of variance as the decision parameter. Moreover, their work only indicates the variance as the decision parameter and states the usage of absolute distance; however, the absolute distance cannot be the only indicator since it does not contain the directional information about the trend of bandwidth on a link. In addition, the algorithm requires end hosts to mark their traffic for prioritization.

Like LABERIO, DLPO [13] tries to load balance data center traffic during flow transmission using path switching. It improves LABERIO by providing a priority-based flow table update strategy. The aim is to improve network throughput and effectively resolve the congestion problem. The use of priority-based flow table update is to redirect the flows of the congested paths to lightly loaded paths without causing packet loss. DLPO algorithm is composed of two stages: path initialization and dynamic path optimization. In the path initialization stage, the algorithm tries to find a temporary path according to the available bandwidth of each path's bottleneck link among all the paths between source and destination. In the dynamic path optimization stage, DLPO uses OpenFlow to retrieve statistics from switches to detect the load balance status. If the link status is imbalanced, the path optimization algorithm is triggered. In order not to face the frequent path switching problem reported in the LABERIO, authors chose to use the simple moving average (SMA) of the variance of link loads as the indicator of load imbalance. The SMA formulation used in their algorithm uses the average load of all the links in the network. DLPO employs two different load balancing methods as multi-link and single-link DLPO. The multi-link tries to load balance the top 10% busiest links and the single link version reroute flows in the highest utilization link. The priority-based update strategy increments the newly added flow rules' priority before removing the old rules. DLPO introduces significant improvements over LABERIO and has special methods for avoiding the frequent path changes and packet loss due to changes made on the switch flow tables. The partial path modification and activation after the congestion occurs, loses the edge provided by the global view of SDN.

L2RM [14] is an example of macro management for data center networks. L2RM uses an SDN controller and statistic polling from switches to get a global view of network and distributes flows by inserting appropriate flow rules to the switches. The work presented in L2RM has three main parts: adaptive route modification (ARM) mechanism, switch health check (SHC) mechanism, and dynamic information polling (DIP) mechanism. The aim of L2RM is to increase network performance by load balancing traffic in fat tree data center topologies. The arm mechanism follows the load variation in order to insert a route modification. In order to achieve load balancing among the routes, L2RM uses bucket weights to share the traffic among action buckets in different group tables. L2RM maintains path, load and entry tables for each switch in the network and uses SHC mechanism to update these tables. The load table keeps traffic information of each port on each switch. The load table is also used as the general view of the network. The ARM mechanism is invoked according to changes in the load table. The SHC mechanism is used for controlling the path table. As the metric for invoking the ARM, L2RM uses a scaled-up version of variance from the average port load in the network. If the variation is greater than a predetermined threshold value, the ARM mechanism is invoked. In order to avoid unnecessary invocations that can be caused by instantaneous bursts, the ARM mechanism is invoked if the port traffic exceeds the threshold for two successive measurement intervals. The ARM mechanism selects the link with heaviest load then selects the highest bandwidth demanding flow from the link and moves the flow to alternate link. If the alternate link is also heavily loaded, another alternate link is selected. Then the bucket weights are adjusted to share the traffic among primary and alternate ports. The DIP mechanism provides the data needed to take these actions. L2RM deploys methods for reduction of process load on the controller, such as timeout values for flow entries, random exponential back off algorithm for statistics polling. The algorithm however considers only primary and backup routes for load balancing. The load balancing mechanism is based on variance directly, which can give false alarms by rapidly changing in accordance with the traffic load. Also, algorithm waits for two consecutive measurement intervals for invoking the ARM to distribute the load on a link, which would degrade its performance under highly varying loads by not performing the load distribution.

A different approach to load balancing problem is proposed in [8] and named Centflow. The authors propose the use of centrality functions which has been research topic of social networking, in order to determine highly utilized central nodes and edges in a network. CentFlow employs betweenness centrality and temporal node degree to measure how central a node is or how often an edge is selected for flows. The aim is to influence the routing algorithms to select the paths that uses fewer central nodes by dynamically evaluating the packet forwarding capacity of each node and deferring the nodes that can cause saturation in the network. This way, the3 authors aim to achieve a more load balanced traffic distribution in the network. The definitions of geodesic and geodesic distance are given as the shortest path and its length between two nodes in the network. Betweenness centrality of a node is defined as the ratio of number of geodesics to the total number of geodesics from a source to destination. Similarly, edge betweenness is also defined as the ratio number of geodesics to the total number of geodesics from a source to destination. After a network topology is obtained CentFlow starts by assigning random weights to the nodes. The node and centrality measures are computed and dynamically. When a node hits the threshold utilization value the node is disconnected from the graph and centrality values are recomputed. The authors combine their algorithm with the Dijkstra's shortest path algorithm and demonstrate the results as 62% increase in node utilization and 49% increase in link utilization. The contributions of the CentFlow are very significant since it describes a more inclusive approach to load balancing, puts no differentiation between TCP or UDP flows and doesn't require any end host or switch level modification on flows.

The research presented in [35] aims to perform load balancing by calculating each switch capacity across a path to which packets are routed in advance. The proposed algorithm presents the network as a simple, directed, connected graph. The capacity is defined as bytes per second data thorough a given switch. When a flow is received by the controller simple paths are created to deliver the packets from source to destination. Algorithm determines multiple paths from source to destination. Minimum threshold values are given for each path and the average of these values are calculated to determine the average path threshold. The load balancing action redirects packets to alternate path when the received packet exceeds this threshold. The test results for the study is given in terms of Request Success Rate, Request Failure Rate, Response Time, and Link utilization, no mention of request. These metrics are meaningful for measuring the effectiveness of the load balancing on service or a server, however no consideration is given to the traffic loads in the network or its distribution.

In [36], Shu et al. several studies in SDN are inspected in terms of traffic engineering. The study by Shu et al. propose a reference framework that divides the traffic engineering into two main fields as traffic measurement and traffic management. Their work also divides the network structures as SDN and IP where IP refers to the traditional networks. Traffic measurement technologies are classified as network parameter measurement, general measurement framework, traffic analysis and prediction. The traffic management technologies are classified as traffic load balancing, quality of service guaranteeing, energy saving and hybrid SDN/IP traffic management. The authors emphasize that the traffic engineering is a necessity in traditional networks due to routing protocols' inefficiency in managing traffic in accordance with the changing traffic parameters and SDN is a promising technology in traffic engineering due to the control, programmability and openness characteristics. It is claimed that these characteristics can solve current traffic engineering problems in traffic measurement, scheduling, management, flexible flow management. As examples of network parameters measurement, the authors inspected works on traffic statistics collection, traffic matrix estimation and dynamic traffic change analysis. For generic measurement, the paper focuses on flow sampling, flow polling, network device resource management and accuracy of measurement. For the traffic analysis, the authors focus on the consistency checks in terms of configuration and topology, abnormal traffic detection and routing loop detection. In the traffic load balancing, the authors inspect studies on package level load balancing, equivalence multipath routing (ECMP), elephant flow detection and routing optimization according to elephant/mice flow classification. In quality of service guarantee scheduling, the authors focus on studies related to flexible traffic scheduling strategies to satisfy QoS requirements such as queue management, flow scheduling and IP packet header information handling to improve QoS performance of networks. The authors claim that networks consume 50% of energy spent for services delivered over the networks, therefore the energy consumption should be considered for traffic management. They support their claim by presenting research that tries to utilize sleep states on ports, minimizing number of routes used in order to shut down switches that are not relevant for optimized traffic routing and optimizing service schedule in accordance with the QoS requirements of flows in order to minimize the number of routing paths used. The survey handles the hybrid IP/SDN structures from the aspects of mutual benefit and interoperability. In terms of mutual benefit, deployment strategies create a performance improvement for both IP and SDN networks and migration strategies are inspected in terms of interoperability. The survey is concluded by stating the importance of measurement for providing traffic engineering in SDN and importance of traffic engineering for the widespread adoption of SDN architecture.

The work in [37] is a survey on load balancing techniques in software defined networks. It also provides a summary of research challenges and directions for load balancing. The authors also investigate mathematical models and emulators commonly used in testing of algorithms. Authors state the difficulties arising from the traditional IP networks routing and route distribution mechanisms prevent the implementation and deployment of intelligent routing solutions while the global view and flexible programming provided by SDN enables such solutions. The load balancing techniques are classified as controller load balancing, server load balancing, wireless networks link load balancing, communication path load balancing and artificial intelligence-based load balancing. Controller load balancing focuses on solving the controller load disparity in the presence of multiple controllers. The main idea of related papers is to cluster the switches to the controllers in a way that allows the controller to share the process load. In server load balancing, research efforts show variations since the performance metrics show a high variety when the applications are involved. Some examples of the metrics are request numbers, throughput and response time and the methods used involve application of swarm intelligence for cloud computing, adjusting flow paths according to traffic demand or server allocation according to client demand. The authors state that SDN controllers and load balancing solutions can be deployed to wireless networks in order to dynamically load balance access point (AP) traffic. The research work investigated involves measuring traffic load to determine the optimal associations, variance-based traffic routing and measuring the end user experience for effective load balancing. In communication path load balancing, the authors inspect routing methods used in [9] and [13] and state that these algorithms are not generalized and need to be changed when the topology changes. The remaining studies are criticized in terms of scaling problems, low response time, packet loss or high energy consumption. In the artificial intelligence-based load balancing, it is stated that load balancing is achieved by different artificial intelligence techniques such as artificial neural network, reinforced learning and deep neural networks. The authors state that, since artificial intelligence methods can model complex algorithms with precision and rationality, the decision-making process of the controller is positively affected by the deployment of such algorithms.

The SILBR routing algorithm proposed in this thesis, like the macro level traffic management studies explained, relies heavily on the utilization measurements carried out by the switches and collected by the SDN controller in a network. In a network, traffic demand generally exhibits a highly dynamic behavior and instantaneous link usage measurements may not reflect the actual long-term usage behavior. Hence, SILBR employs moving averages and statistical inference to evaluate the long-term traffic demand on core network links. Hence, a more credible network state information is obtained, and this contributes to the stability of the routing decisions. Moreover, our algorithm relies on inferences for decision making rather than depending on the measurements alone. This approach improves the accuracy of the routing decisions made. Unlike the studies examined so far, in SILBR, rerouting is only used as a correction method for even traffic distribution when necessary and adequate. Therefore, the need for rerouting is minimized thereby reducing the processing load on the controller and the control messaging required for management of the network.

CHAPTER 4

STATISTICAL INFERENCE BASED LOAD BALANCED ROUTING

The primary objective of Statistical Inference Based Routing (SIBLR) algorithm proposed in this thesis is to evenly distribute traffic in the network by minimizing the maximum link utilization. The utilization of a link reflects the average load on the link and it is expressed as the percentage of the utilized link capacity [38]. Hence, Maximum Link Utilization (MLU) in a network can be defined as the utilization of the most utilized link in the network at a given time instant or during a given time interval. One of the most common approaches employed in traffic engineering is the minimization of MLU and it serves several purposes such as improved scalability of the network, lower end-to-end packet delay, and higher flow throughput.

A routing algorithm that leads to a smaller MLU allows increasing the existing traffic volume by a larger factor. Hence, such routing algorithms contribute to the scalability of the networks. Accordingly, by minimizing MLU, the mean time between network link capacity upgrades can be minimized.

The average queuing delay experienced by packets in a link increases with the utilization of the link and there is a non-linear relationship between them. Therefore, if queuing delays in a network are larger compared to other delay components, a lower MLU value usually results in a lower end-to-end packet delay.

In a network, flows can arrive at and leave the network at random times and multiple flows usually share one or more network links. The fairness objective of the communication protocols, such as TCP, leads to equal bandwidth share for the flows passing through a bottleneck link [39]. Hence, minimizing the number of flows passing through the bottleneck link enables providing more bandwidth to individual flows passing through that link. On the other hand, over-utilization of some of the links in the network results in lower throughput for the flows passing through that link. Therefore, MLU also plays a very important role in the maximum throughput that can be achieved by individual flows.

In order to minimize MLU, the routing algorithm must take into account the current state of the network such as the topology of the network, link capacities and link loads. If a flow is routed without considering the current link utilizations, some of the links may be over-utilized while many links in the network are lightly loaded. Hence, a routing algorithm that aims to minimize MLU may proactively route flows over under-utilized links to avoid congestion in some of the links. Hence, minimizing MLU generally leads to moving load that would normally be passing through the most utilized link to other links. Alternatively, the routing algorithm may reactively achieve minimization of MLU by re-routing flows passing through the mostly utilized link to alternative paths passing through lightly loaded links. In both cases, the routing algorithm needs to know the current state of the network. The SILBR embodies both approaches and depends on the global view of the SDN controller in order to obtain current network status in terms of characteristics such as topology and the link bandwidth utilization levels.

In order to minimize MLU (and accordingly achieve evenly distributed traffic in the network) SILBR defines a link cost metric that reflects the degree of even distribution of load across the links. The link loads are periodically collected by SILBR. These successive link load measurements serve as sample measurements that allow inferring the average load on that link. SILBR then calculates the probability that the link's utilization is larger than the average link utilization in the network. The link cost that will be used in routing decisions is determined according to this probability. SILBR also has rerouting feature. SILBR keeps track of the paths that are being used by the flows. If an imbalance in the flow distribution is observed SILBR reroutes some of the flows to other alternate paths to restore even distribution of the load across the links.

In SILBR, we assume that each link in the network core has a predefined and equal maximum capacity. However, the access networks, which are used to connect end nodes to the network, may have different capacities. As the link capacities are the same in the core of the network, the link utilizations are proportional to traffic load on the links (i.e., utilization = link load / capacity). Hence, in the following, link load and link utilization are used interchangeably.

We periodically measure the load on each link in the network core by using the simple moving average method. Please note that the moving average for a link merely corresponds to a sample mean which may be smaller or larger than the actual mean load on that link. Nevertheless, by considering the measurements used in moving average operation and the relevant statistics such as the number of measurements, the sample means and standard deviations, we can build confidence intervals and make inferences about the actual mean of the link load.

We also periodically compute the "grand mean link load" for the entire network by computing the average of the latest link load averages for all links in the network core. At any time, the mean load on a link could be higher or lower than the grand mean link load. The selection of paths should be made in a way to minimize MLU of the links that constitute the paths. That is, we try to avoid passing through links that have mean loads above the grand mean link load. Therefore, we can minimize MLU in the network to achieve our objective. To this end, we calculate the probability that an individual link's actual mean load is higher than the grand mean link load in the network. Hence, the routing algorithm chooses the path that has the minimum probability that at least one of the links in the path has a mean load that is larger than the grand mean link load.

The details of the path selection are as follows:

Time is divided into slots of equal length and average amount of traffic passing through each link is computed for each slot.

W: Window size to calculate the simple moving average of link load.

$\beta_i(n)$: The amount of traffic flowing through ith link at time slot n.

$\mu_i(n)$: At time slot n, the simple moving average of the latest W traffic measurements for the ith link. It is calculated as:

$$\mu_i(n) = \frac{1}{W} * \sum_{k=0}^{W-1} \beta_i(n-k) \tag{8}$$

$s_i(n)$: The unbiased sample variance of the load in ith link at time slot n. It is calculated as:

$$s_i(n) = \sqrt{\sum_{k=1}^{W-1} (\beta_i(n-k) - \mu_i)^2/(W-1)} \tag{9}$$

$\mathrm{M}(n)$: The grand mean link load at time slot n for the network. It is equal to the average of the sample means of the links in the network core. Let N be the number of links in the network core. The grand mean link load is calculated as:

$$\mathrm{M}(n) = \frac{1}{N}\sum_{1}^{N} \mu_i(n) \tag{10}$$

The above definitions and formulas give the information about the current state of the network in terms of the grand mean link load and individual link loads. The grand mean link load gives an idea about the average amount of data carried by the links in the entire network. An indicator of even distribution of load in the network may be how close the individual link means are to the grand mean link load. Hence, we can conclude that if the link loads are very close to the grand mean link load the load is evenly distributed in the network.

As explained above, we can think of the moving average for an individual link as the sample mean for that link. The sample mean gets closer to the actual mean of that link as the size of the moving average window (W) gets larger. From the statistical point of view, the sample means may be assumed to follow the Student's t-distribution [31] with a degree of freedom one less than the sample size (i.e., W-1). Hence, we can make inferences about the actual link mean by using the Student's t-distribution. For instance, we can find the probability that the actual link load is greater than the grand mean link load by using the t-value. The t-value for each link at the nth time slot, denoted by $t_i(n)$, is calculated as:

$$t_i(n) = (\mu_i(n) - \mathrm{M}(n))/(s_i(n)/\sqrt{W}) \tag{11}$$

The cumulative distribution function of t distribution evaluated at the t_i(n) for each n gives us the probability $P_i(n)$ of finding the actual link load, greater than $\mathrm{M}(n)$. That is, $P_i(n) = \Pr(\mu(n) > M(n))$. Hence, the probability that the actual link load mean is less than M(n) becomes $1-P_i(n)$.

As the traffic on a link I increases, it will be more likely to have a sample mean (i.e., the moving average) greater than the grand mean link load and $P_i(n)$ will be close to zero. Similarly, as the load on a link gets lighter, it will be more likely to have a sample mean less than the grand mean link load and $P_i(n)$ will be close to 1. This implies that the probability $P_i(n)$ and the cost for link may be related to each other to achieve even distribution of load in the network.

If we assume that the loads on the links are independent of each other, for a path composed of k links, the probability C(n) that every link that the path passes through has a sample mean less than the grand mean, M(n) can be found as:

$$\mathrm{C(n)} = \prod_{i=1}^{k}(1 - P_i(n)) \tag{12}$$

In order to convert multiplication to sum, we can take the natural logarithm of both sides as:

$$\ln C(n) = \ln\left(\prod_{i=1}^{k}(1 - P_i(n))\right) = \sum_{i=1}^{k}\ln(1 - P_i(n)) \tag{13}$$

Therefore, instead of maximizing C(n), we can maximize $ln\ C(n)$, or equivalently minimize $-ln\ C(n)$, in order to select the paths with least amount of traffic. Accordingly, finding the best path reduces to finding the shortest path in the network when we define the cost of link I at time slot n as:

$$c_i(n) = -\ln(1 - P_i(n)) \tag{14}$$

At time n, the cost of a path consisting of k links becomes:

$$Path\ Cost(n) = -\sum_{i=1}^{k}\ln(1 - P_i(n)) \tag{15}$$

Therefore, in order to minimize MLU, we can choose the path that minimizes equation (15). The equation (15) represents the more natural way of path selection in the shortest path finding algorithms.

In order to further refine the minimization of MLU, SILBR uses rerouting process which is also based on the SILBR metric. Each flow in the network is registered with a unique hash code. The hash code is also used to track the path information that the flow is assigned to. The SILBR first controls the number of flows assigned to each path. If the algorithm finds an imbalance in the flow distribution a rerouting process starts. The process checks for the alternative path costs of each flow. If an alternative with lesser cost is found, the flow is moved to the alternative path.

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation

This Section describes the design and implementation of the described algorithm and its performance evaluation.

Floodlight SDN controller [40] is chosen as the development environment of the algorithm. Floodlight controller is designed to operate as a highly concurrent system and utilizes the multithreaded design. The controller supports for users to create their own modules and supports REST applications by making several REST interfaces available. For ease of access to the controller resources and information, the algorithm is implemented as a module into the Floodlight controller.

The process flow of the algorithm is given in the Figure 4.

5.1.1 Packet In Listener:

The module is implemented as a PACKET_IN listener. Whenever a packet in message is received from the switches, the module starts related processes. Process details of SILBR implementation are given in the following sections.

5.1.2 Topology Discovery:

For every PacketIn message received by the controller, SILBR checks if the internal switch set contains the DatapathID of the switch that sends the message. If the switch DatapathID is not recorded before, module starts an update process using the Switch Service of the Floodlight Controller.

5.1.3 Host Discovery:

SILBR implements passive host discovery by listening to the Address Resolution Protocol (ARP) messages. Host discovery uses the Device Service of the controller. The findDevice method returns the connection point information about the device.

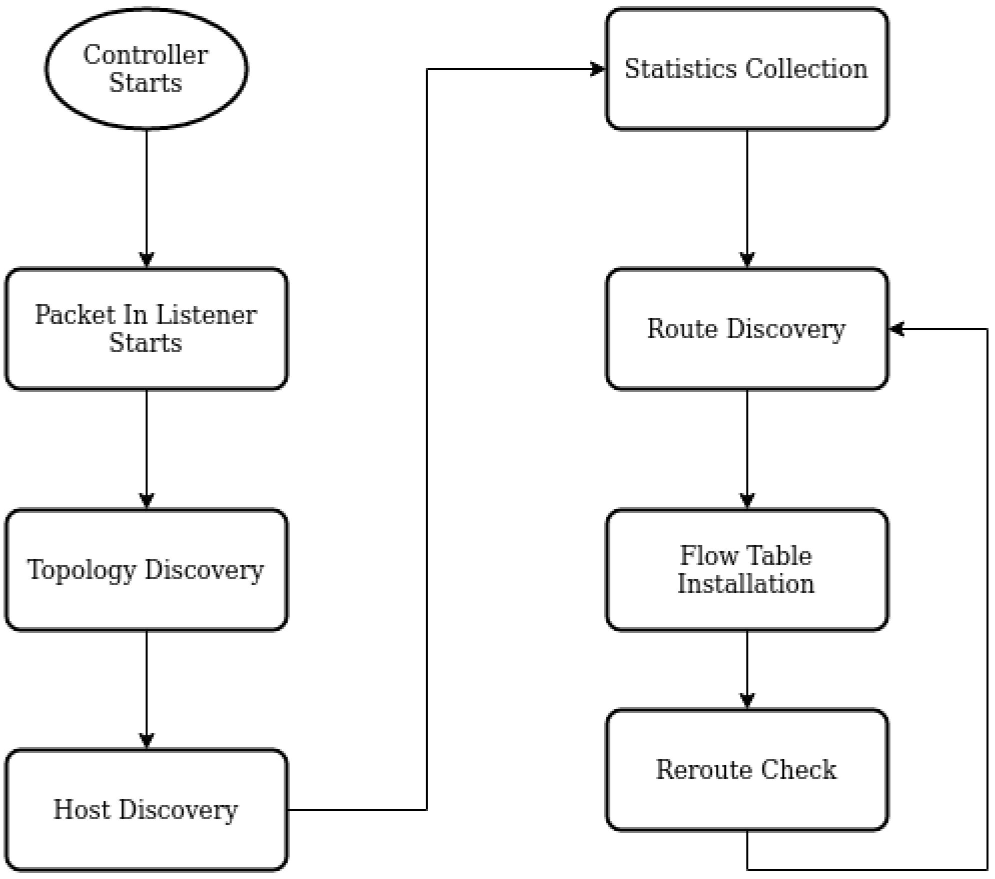


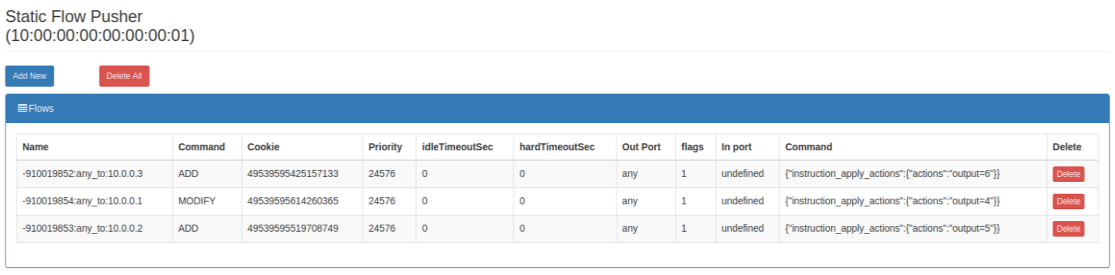
Figure 4 Process Pipeline of SILBR Implementation

The host information is stored in the form of an IP table that contains the DatapathID of the switch and port number of the host connection. The flow routing process is not started until both the source and destination IP addresses of the flow is included in this table. The structure of the table is exemplified in Figure 5.

IPv4 Address	Switch Datapath ID	OF Port
10.0.0.1	Id=10:00:00:00:00:00:00:01	Port = 1

Figure 5 SILBR Host Table

As the hosts are discovered, SILBR adds flow rules to the switches. These rules are exceptions to the normal operation of the SILBR algorithm. Under normal operation conditions the flow rules are used for routing and are installed for a limited time. However, host rules are permanent, and these rules direct the packets, whose destination IP addresses matches the end host, to the connection port of the end host. An example of the process is displayed in Figure 6.


Static Flow Pusher
(10:00:00:00:00:00:00:01)

Name	Command	Cookie	Priority	idleTimeoutSec	hardTimeoutSec	Out Port	flags	In port	Command	Delete
-910019852:any_to:10.0.0.3	ADD	49539595425157133	24576	0	0	any	1	undefined	{"instruction_apply_actions":{"actions":"output=6"}}	Delete
-910019854:any_to:10.0.0.1	MODIFY	49539595614260365	24576	0	0	any	1	undefined	{"instruction_apply_actions":{"actions":"output=4"}}	Delete
-910019853:any_to:10.0.0.2	ADD	49539595519708749	24576	0	0	any	1	undefined	{"instruction_apply_actions":{"actions":"output=5"}}	Delete

Figure 6 Floodlight Host Rules

5.1.4 *Statistics Collection:*

SILBR collects switch port bandwidth usage statistics from every switch in the topology in every 2 seconds. The data contains bandwidth measurements from the ports in RX and TX directions separately. After the collection, port data is associated with a performance data. Performance data contains separate records of RX and TX bandwidth values. Using these records statistics such as mean, variance and t-score values are calculated and added to the record. According to these calculated values each port is assigned a cost value. For this process SILBR Link Discovery, Statistics and Topology services of the controller. The process runs as a single separate task.

5.1.5 *Route Discovery:*

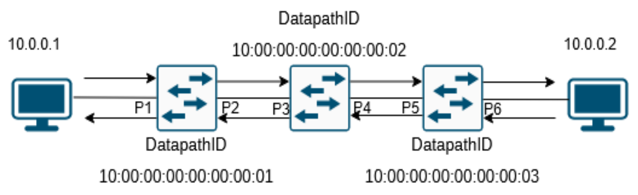
SILBR collects routing information with a separate thread in every 2 seconds. In order to prevent the algorithm from choosing irrationally long paths, we limited the number of paths that are used in the load balanced routing process. SILBR choses the smallest cost path among the k alternate shortest paths. The k value is chosen as 60 during the testing but it may be determined according to the size of a network. The algorithm calculates up to 60 paths from each source switch to destination switch using the IRoutingService of the controller. The collected path information is stored locally as exemplified in Figure 7.

<Source Switch Datapath ID, Dest. Switch Datapath ID>	List of Unidirectional Paths
<Id=10:00:00:00:00:00:00:01,Id=10:00:00:00:00:00:00:02>	{Path1, Path2 … Path60}

Figure 7 SILBR Path Information

Each path starts with the outbound connection from the starting switch then continues as receiving and leaving ports on the next switch until receiving port of the destination switch is reached.

The collected routing information is combined with the cost value calculated from the port performance data and stored locally to create a routing table. For the purpose of traffic load distribution, we defined the path cost value as the cumulative value of outbound link costs that the path is traversing. An illustration of path cost calculation is given in Figure 8.



Path 1 from 10.0.0.1 to 10.0.0.2

<10:00:00:00:00:00:00:01:P2><00:00:00:00:00:00:00:02:P4><00:00:00:00:00:00:00:03:P6>

Cost of Path1 = Cost(P2) + Cost(P4) + Cost(P6)

Path 2 from 10.0.0.2 to 10.0.0.1

<10:00:00:00:00:00:00:03:P5><00:00:00:00:00:00:00:02:P3><00:00:00:00:00:00:00:03:P1>

Cost of Path2 = Cost(P5) + Cost(P3) + Cost(P1)

Figure 8 Path Cost Calculation

5.1.6 Flow Table Installation Process:

SILBR listens to incoming PACKET_IN messages from switches to decide on flow routing. When both host discovery and route table processes are completed, the module starts routing flows by installing static flow entries into the switches.

According to [41], PACKET_IN messages contain the header information of the packet that causes the message. The SILBR module checks the header information in order to determine the corresponding flow’s source and destination IP addresses. SILBR obtains the source and destination switch’s DatapathID information by using the host discovery information. Possible paths for the source and destination switch DatapathIDs are sorted according to the cost information; after sorting, the minimum cost path and the data payload information is passed to a new update thread for installing corresponding flow table rules to the switches.

The update task uses source, destination IP and port information in the data payload and creates a unique hash value for each flow. The produced hash, flow details and path information are stored locally and updated as flows added and removed in order to use in rerouting process and keep track of the flows in the network. In order to create the required match information, the task uses OFFactories for OpenFlow version 1.4.

The factories are implemented in the controller. The update task uses the Static Entry Pusher Service of the floodlight controller to send the flow table rules to the switches.

If the flow is a new one, the flow rules are added with medium level priority and 2 seconds idle timeout value. Therefore, when a flow is finished related rules are removed from the switches automatically. If the flow is a rerouted one, the flow rules are added with highest priority and 2 seconds idle timeout value so that the previous rules, related to the same flow, time out within 2 seconds.

5.1.7 Rerouting:

The rerouting process also works as a separate task. The task is activated in every 2 seconds to check for reroute requirements. The task uses Static Entry Pusher Service to collect the information about the continuing flows in the network. The flow table installation process keeps track of the paths that the flows are routed through. By considering flow-path information and already constructed routing table, the rerouting task counts the number of flows assigned to the paths and determines the minimum and maximum number of flows assigned to the paths. If the difference is more than 2, the task starts to process the flows. If a path with cost value lower than the current one is found, the task moves the flow to this path and starts a Flow Table Installation task. This way the SILBR tries to equally distribute the flows over the topology.

5.2 Testing

In this section, we present the performance of SILBR and compare it with the performances of other protocols. In order to illustrate the essence of SILBR, we first used a very simple topology and compared the SILBR to static routing and the minimum hop (MH) routing (i.e., the shortest path routing based on hop count). We present the results of this section by box plots of link bandwidth usage levels obtained from the whole network. Then, we compare the SILBR to MH and the Inverse Maximum Available Bandwidth (IMAB) routing (i.e., the shortest path routing with link costs equal to inverse of the available bandwidth) algorithms and observe the traffic distributions and path lengths. We present the results of our experiments as minimum, maximum and average bandwidth plots individually for each algorithm and comparatively for the three algorithms.

For testing we used two different topologies. The tests performed on the simple topology is aimed to show the advantage obtained by using a dynamic metric for traffic distribution. The tests performed on the mesh topology are aimed to compare the performance of SILBR to similar dynamic metrics used for traffic distribution.

The Mininet [42] and MiniEdit [43] tools are used for topology generation and network emulation, respectively. The Mininet tool provides both an API and command line interfaces to create, access and interact with virtual networks and these operations can be automated easily with scripting.

Iperf [44] and D-ITG [45] traffic generators are used for creating desired traffic patterns among the hosts. Both traffic generators are widely used for research purposes and can produce network traffic in the packet level by replicating the necessary stochastic processes. Both traffic generators also allow manipulation of the network traffic by changing the packet level parameters such as packet inter-departure times, average packet sizes.

5.2.1 *Simple Topology:*

The Figure 9 depicts the topology used in this part of testing.

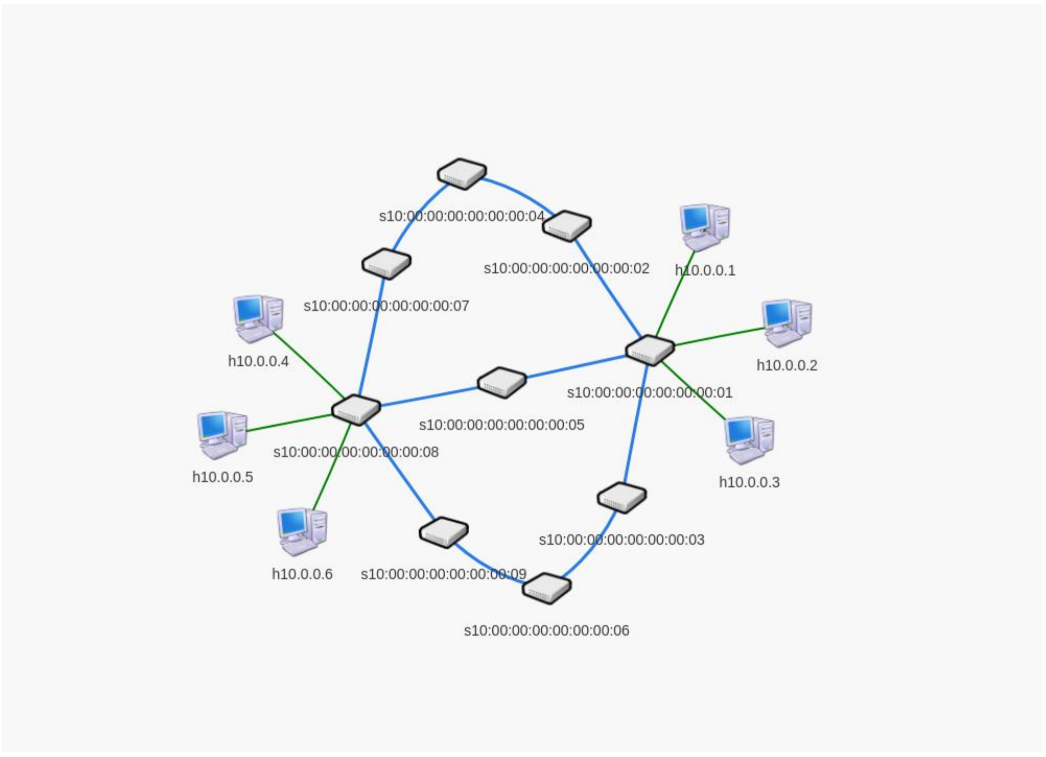


Figure 9 The Simple Topology Used

The network setup is composed of longer alternate paths and a shortest path passing through switch 5. The host uplink connection is limited to 1 Mbps and core links are operating at 2Mpbs. Each host on one side of the network, transfers data by establishing a TCP connection to a destination host on the opposite side of the network. That is, no traffic inter-switch traffic between the hosts is created. The TCP connections start every 6 seconds. The source sends 1400 byte packets a rate of 200 pps (packets per second) and inter-arrival times are exponentially distributed. Each TCP connection carries 12000 Kbytes of data. The resulting throughput of a single TCP connection is shown in Figure 10.

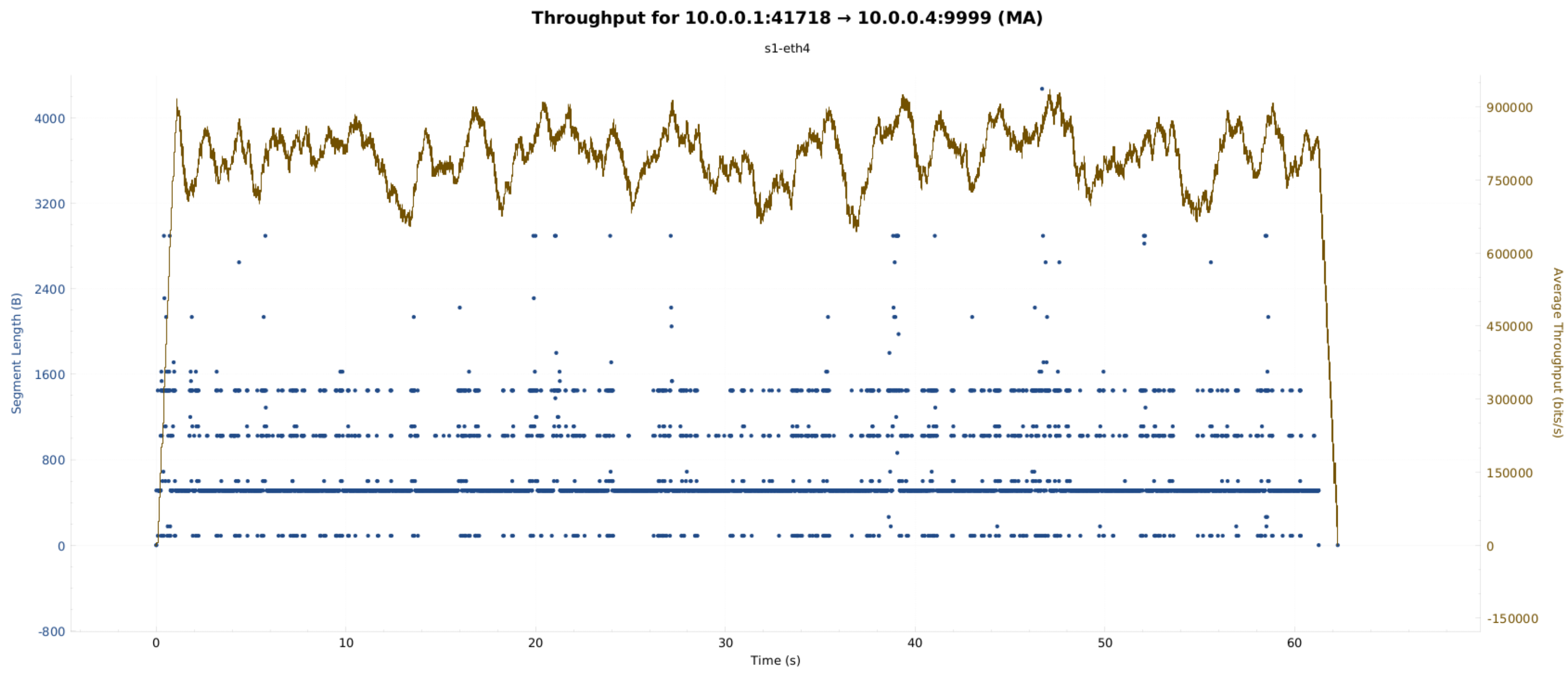


Figure 10 Throughput plot of a single TCP flow – 1400 byte packet size, 200 pps, exponential inter arrival rate .

We first tested the traffic with static routing. The routing is set up in a way that none of the core links will be delivered traffic surpassing 2 Mbps. The resulting traffic distribution is depicted in Figure 11. We would like to emphasize here that; due to the simplicity of the topology and the identical shape of the traffic an ideal traffic distribution is obtained. The average traffic over the network links is 1 Mbps and the maximum utilization is 2 Mbps. Some of the links do not carry any traffic. The resulting box plot shows that the maximum, minimum and average utilization values accordingly.

In Figure 12, the traffic distribution with Minimum Hop (MH) routing (i.e., shortest path routing with unit link cost) is depicted. Since no alternate route is selected for the flows in this algorithm, no traffic distribution is achieved. The resulting box plot shows the measured bandwidths as outliers. The links on the selected path is used up to their capacity.

The Figure 13 depicts the traffic distribution when SILBR algorithm is applied. The resulting box plot shows that the minimum bandwidth usage is increased while the maximum link utilizations are about 25% lower than the link capacity.

When the SILBR routing and the static routing is compared, it is observed that the completion time for the transfer of data is slightly higher for the SILBR, however SILBR increases the completion time only by 2% while it achieves about 40% decrease in the MLU. Also, it is evident from the outliers in Figure 13 SILBR utilizes all the links in the network.

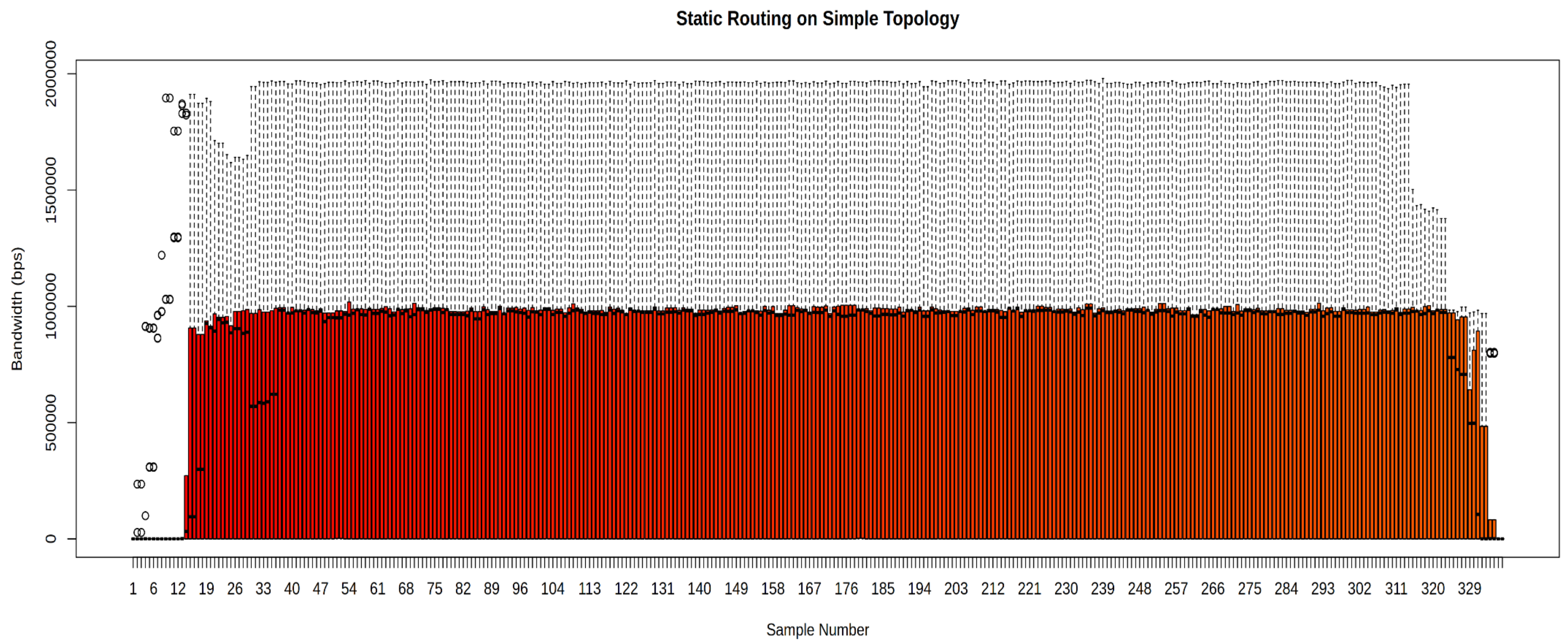


Figure 11 Traffic Distribution with Static Routing on Simple Topology

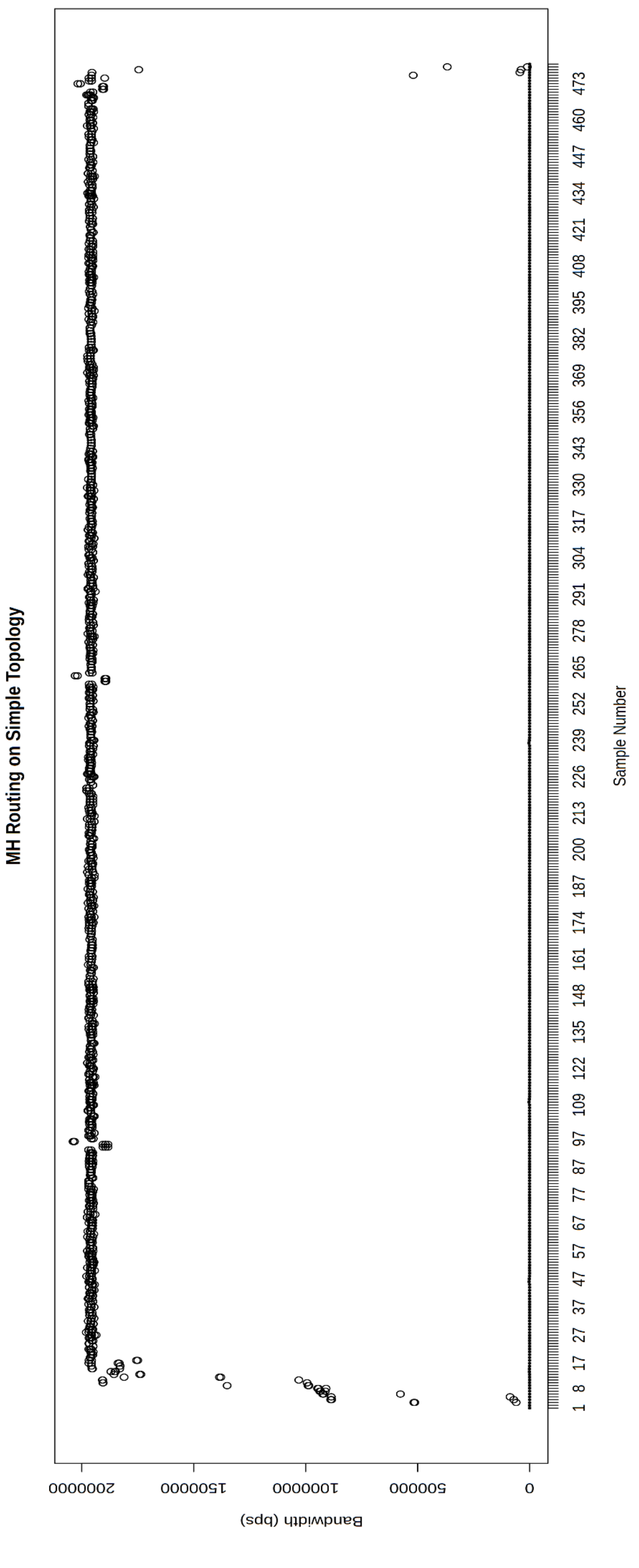


Figure 12 Traffic Distribution on Simple Topology with MH Routing

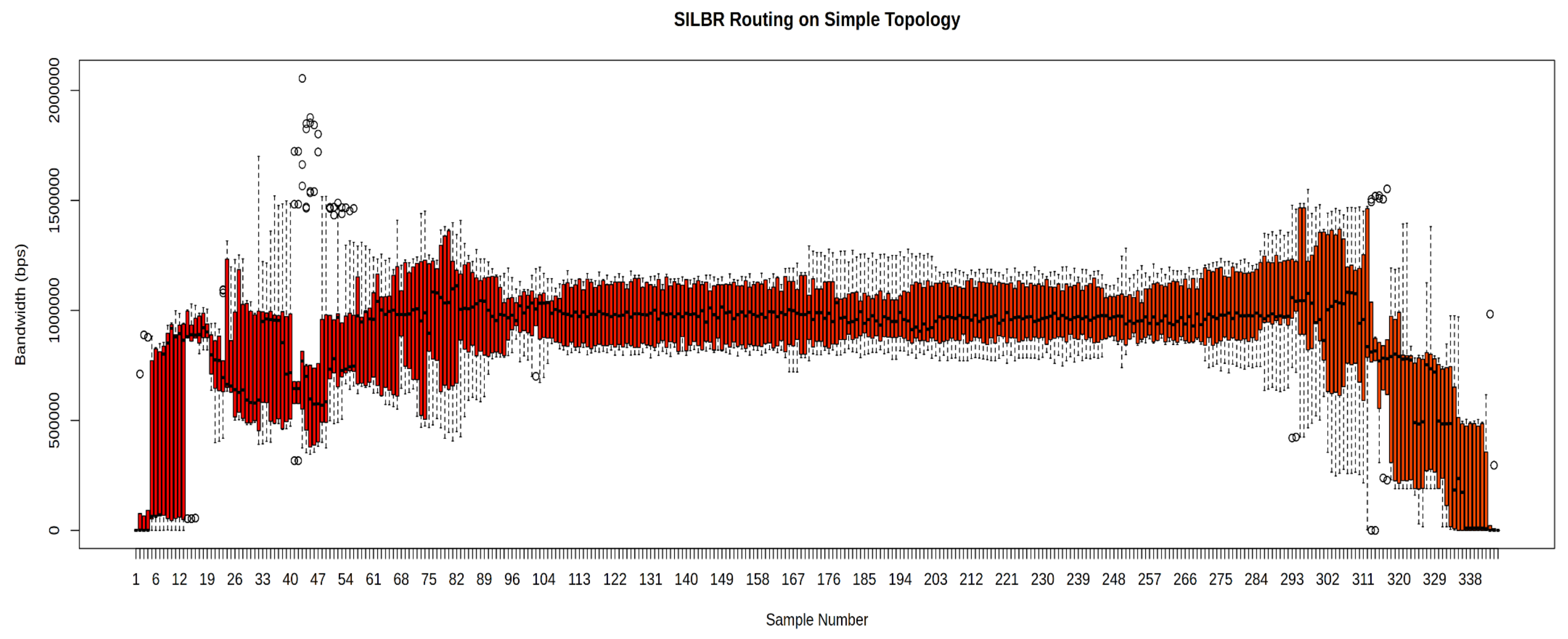


Figure 13Traffic Distribution on Simple Topology with SILBR

5.2.2 *Mesh Topology:*

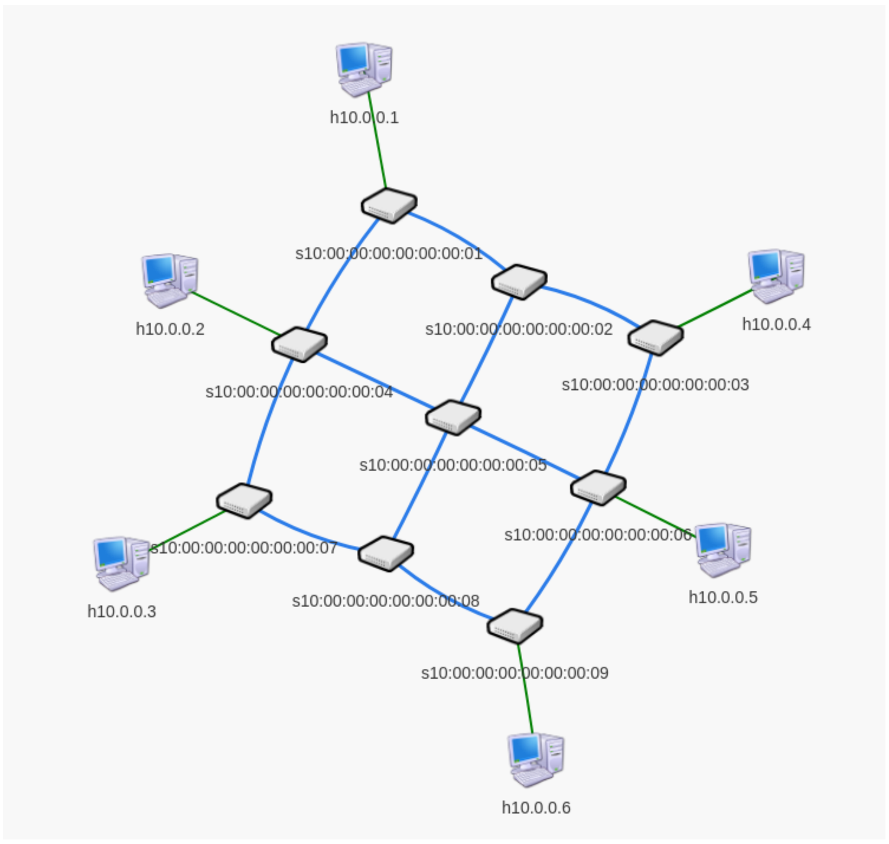


Figure 14 the Mesh Topology Used

Figure 14 shows the second topology used for testing. In this topology, we compared the performance of SILBR with the performances of MH and IMAB. In IMAB, inverse of the remaining link capacity is assigned as the cost for a link and this adaptive approach is widely used for traffic distribution and minimization of MLU.

We carried out two different experiments on this topology. In the first version, we used 1 Mbps uplink connections to the switches (i.e., the access network links), and 5 Mbps links between the switches (i.e., the core network links). We measured the core network traffic and compared the traffic distribution by observing minimum, maximum and average link utilizations. Moreover, we compared the path lengths. In the second version we connected the hosts and switches with 5 Mbps links and observed the minimization of MLU behavior of SILBR, IMAB and MH. The reason we increased the host connections to 5 Mbps is to ensure that the TCP fair share behavior is not limiting the connection speeds.

For this test, we created 10 TCP flows per one second for 20 seconds. Source and destination pairs are chosen randomly for each flow. We conducted the experiment with different payload sizes as 1 Mbyte/flow, 3 Mbyte/flow, and 5 Mbyte/flow. We present 1 Mbyte/flow results here and the results for 3 Mbyte/flow, and 5 Mbyte/flow are presented in the Appendix A. Figure 15, Figure 16, and Figure 17 show our results with 1 Mbps host (i.e., access network) connections.

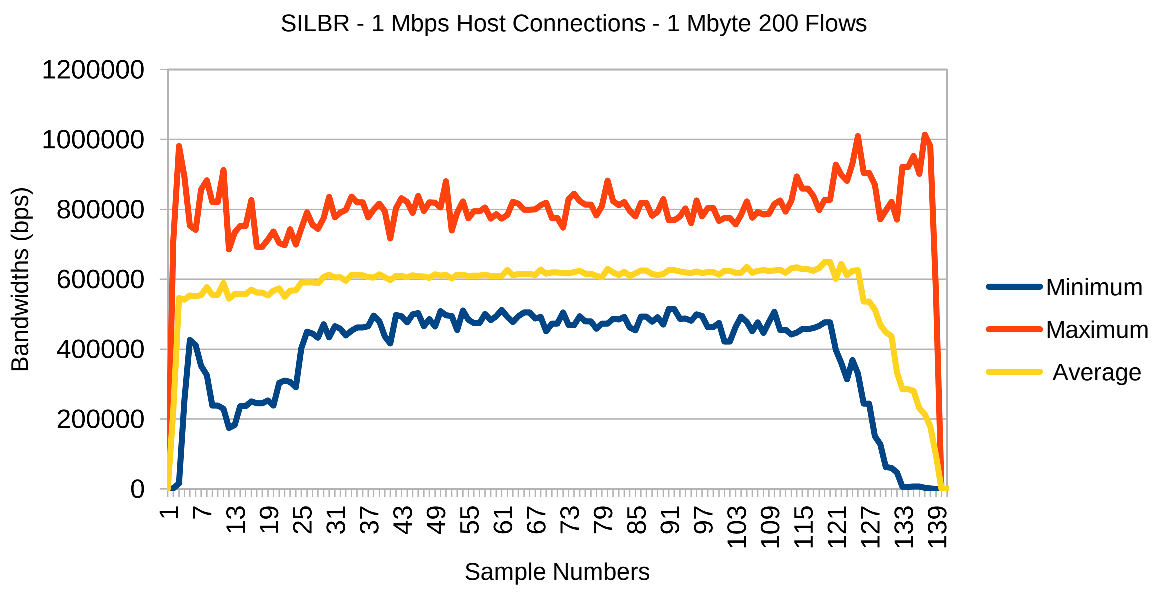


Figure 15 SILBR with 1 Mbps host connections, 1 Mbyte 200 flows

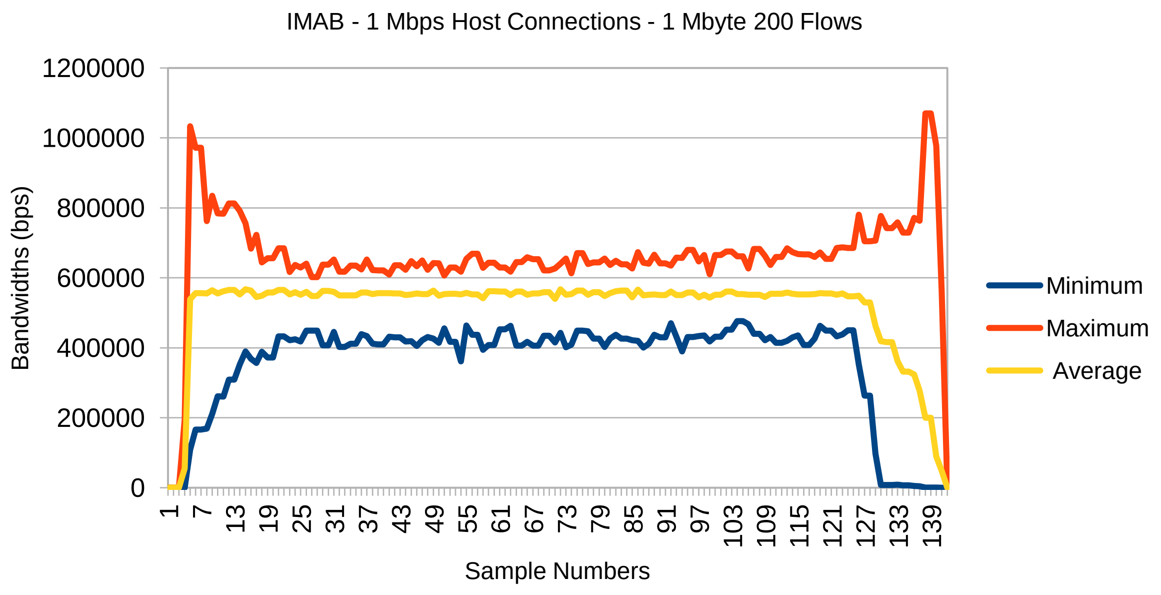


Figure 16 IMAB with 1 Mbps host connections, 1 Mbyte 200 flows

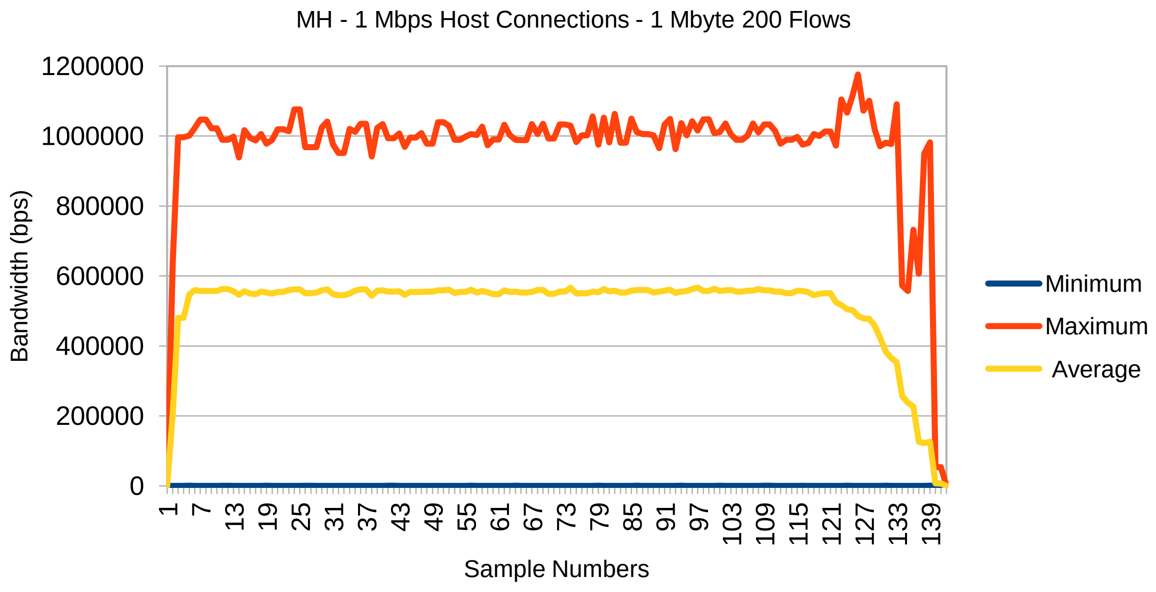


Figure 17 MH with 1 Mbps Host Connections – 1 Mbyte 200 flows

When the results are compared against the MH routing, total transmission times are not affected for both IMAB and SILBR. The maximum bandwidth usages for the IMAB shows better performance. The minimum bandwidth usages are better in the case of SILBR. The supplementary comparison charts for maximum, minimum and average bandwidth usages are given in Figure 18, Figure 19 and Figure 20 respectively.

The sudden increases at the start and end of the traffic shows that the IMAB routes the flows similar to a round robin fashion. As the number of TCP flows increases in the network, flows start to share the same links. Due to the limitations in the host uplinks, as the number of flows increases the TCP fair share principle causes the per flow bandwidth usages to decrease. This shows that the IMAB starts routing flows to lower cost links first and as the number of flows increase in time the newer flows are added to the paths with the oldest flows in the network. This behavior is also evident from the sudden increase at the end of the graph where the latest arriving flows find higher usable bandwidths due to completed earlier arriving flows. This behavior causes changes in the delays and increases jitter in the network.

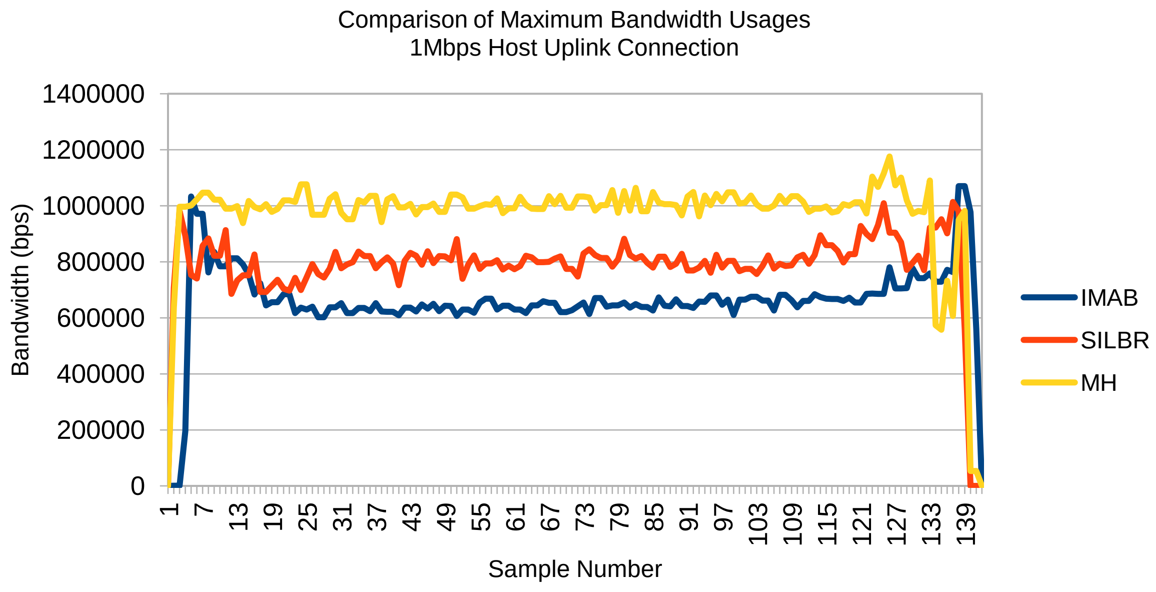


Figure 18 Maximum Bandwidth Usages for SILBR, IMAB and MH metrics, Mesh Topology 1 Mbps Host Uplink Connections

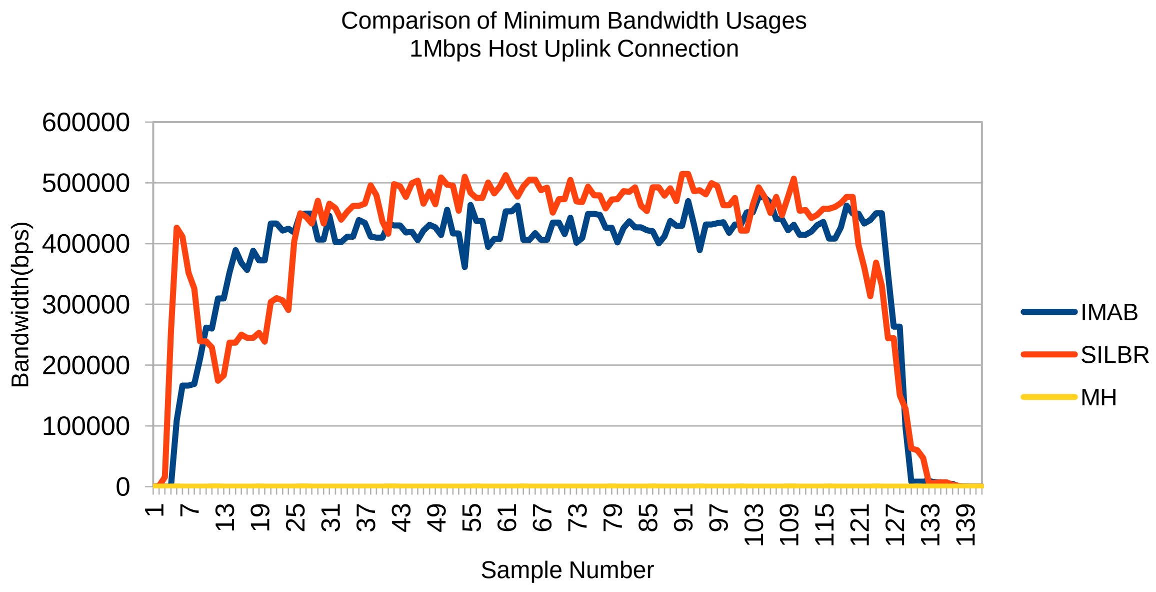


Figure 19 Minimum Bandwidth Usages for SILBR, IMAB and MH metrics, Mesh Topology 1 Mbps Host Uplink Connections

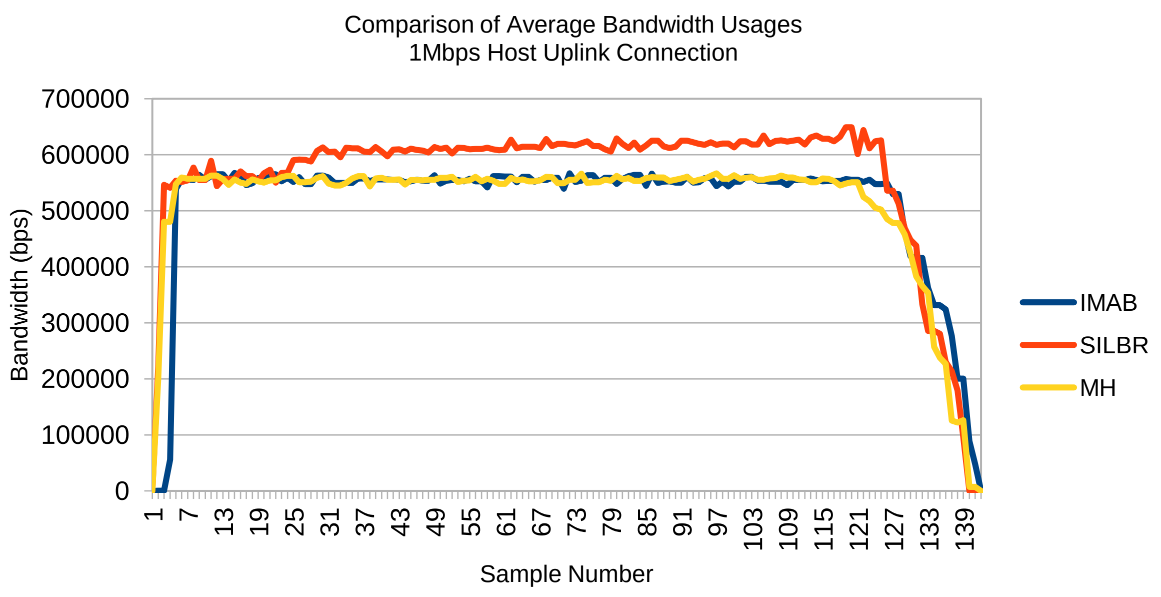


Figure 20 Average Bandwidth Usages for SILBR, IMAB and MH metrics, Mesh Topology 1 Mbps Host Uplink Connections

The comparison of maximum bandwidth usages shows that the SILBR achieves 20% decrease in the MLU and IMAB achieves 38% decrease in the MLU on average. In terms of minimum and average bandwidth usages the SILBR achieves a 20% higher usage. The average bandwidth usage of the SILBR shows an increasing trend.

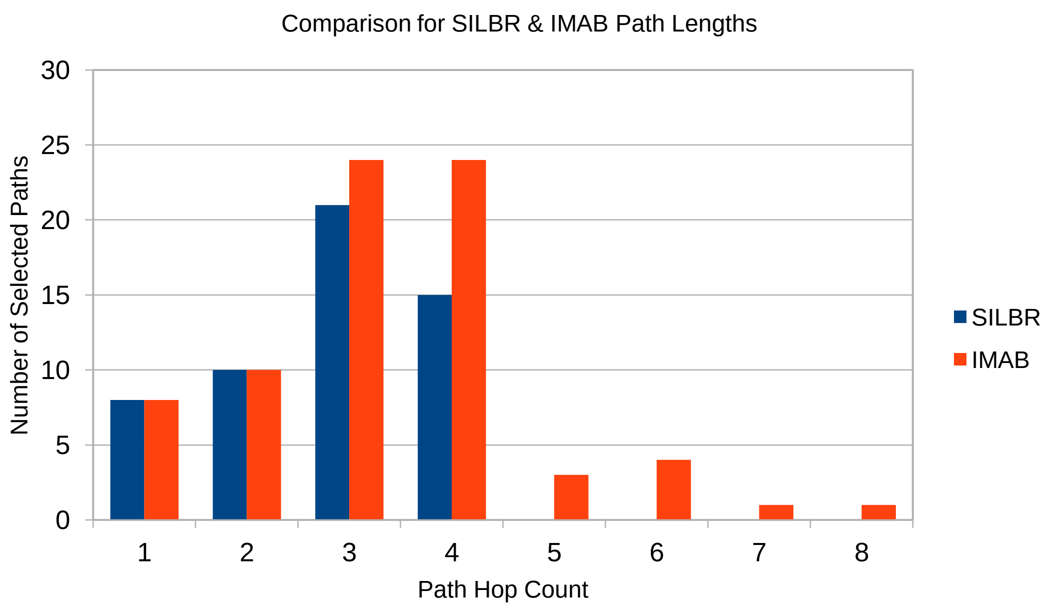


Figure 21 SILBR & IMAB Path Length Comparison 1 Mbps Host Uplink Connections – 1 Mbyte 200 Flows

In addition to the bandwidth measurements, we compared path lengths for SILBR and IMAB in Figure 21. The results of this comparison show that SILBR uses shorter paths compared to that of IMAB. This behavior can be explained by the round robin like traffic distribution performed by the IMAB. The path lengths show that the SILBR creates less traffic load on the network devices since less amount of network devices needs to process the traffic. Another advantage obtained by using the shorter paths is that it causes less processing, transmission, propagation and queuing delays for individual packets. The decrease in both propagation and buffer delays causes a better decrease in the round-trip time. Also, since the variation in the path lengths is smaller in the SILBR, a lower jitter value is expected.

For the second test case, we again created 10 TCP flows per one second for 20 seconds. Source and destination pairs are chosen randomly for each flow. We conducted the experiment with 5 Mbyte/flow payload sizes and the host uplink connections are set to operate at 5 Mbps. The Figure 22, Figure 23 and Figure 24 show our experiment results. The reason for the increase in the host uplink connection is to minimize the effect of TCP fairness and test that the same beneficial results are obtained. The comparisons for 1 Mbps host uplink connections and 3 Mbyte/flow and 5 Mbyte/flow payload cases are presented in the Appendix A

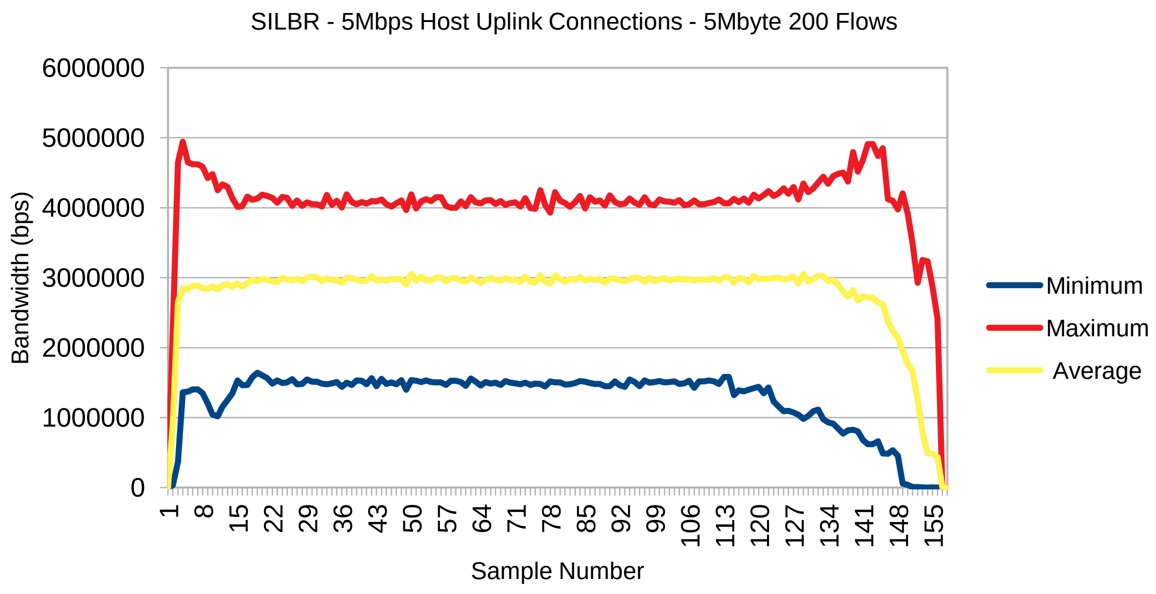


Figure 22 SILBR with 5 Mbps Host Connections – 5 Mbyte 200 flows

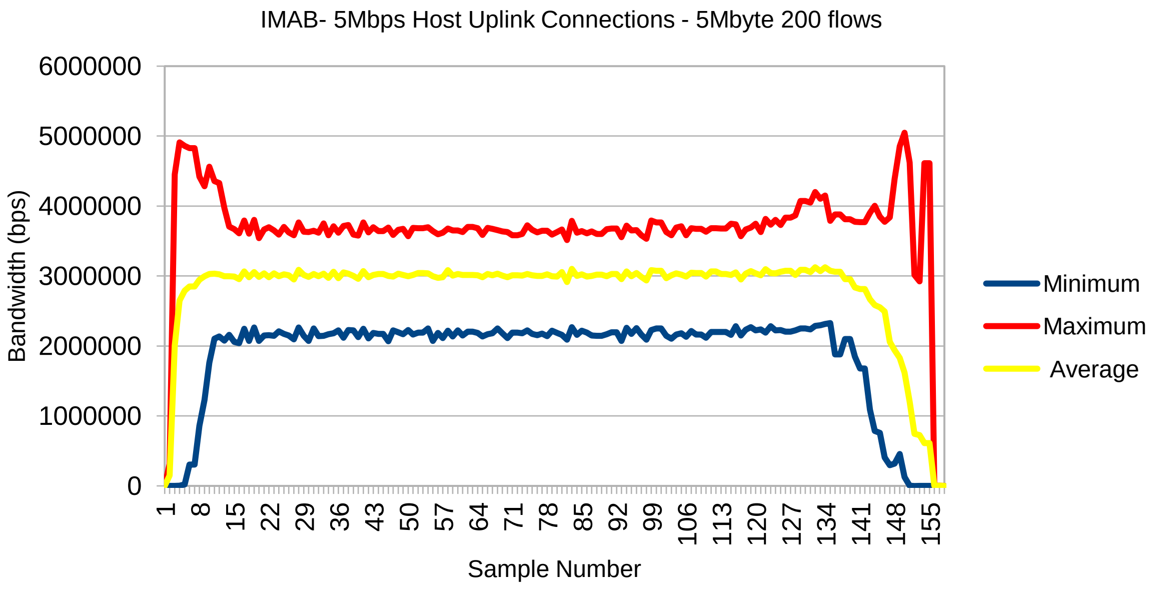


Figure 23 IMAB with 5 Mbps Host Connections – 5 Mbyte 200 flows

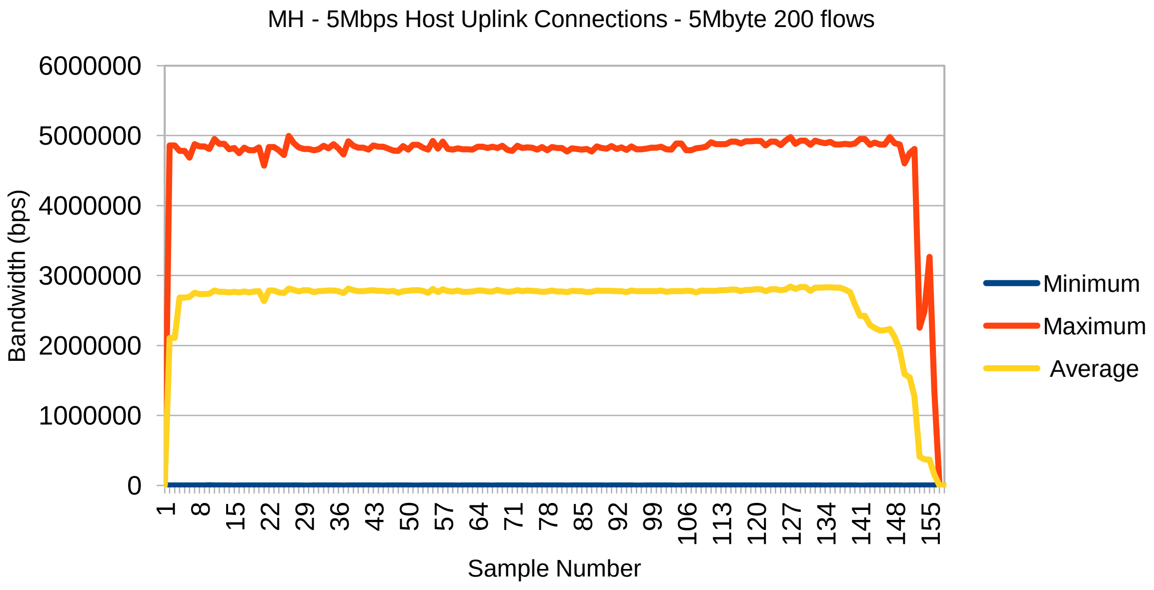


Figure 24 MH with 5 Mbps Host Connections – 5 Mbyte 200 flows

The results presented in Figure 22, Figure 23 and Figure 24 are consistent with the previous test case where the host connections are set to operate at 1 Mbps. The traffic values at the start and end of the data transmission shows a smoother increase and decrease for the SILBR. The supplementary comparison charts for maximum, minimum and average bandwidth usages are given in Figure 25, Figure 26 and Figure 27 respectively. The Figure 28 show the path length comparison for IMAB and SILBR. The results show that the SILBR choses shorter paths compared to IMAB.

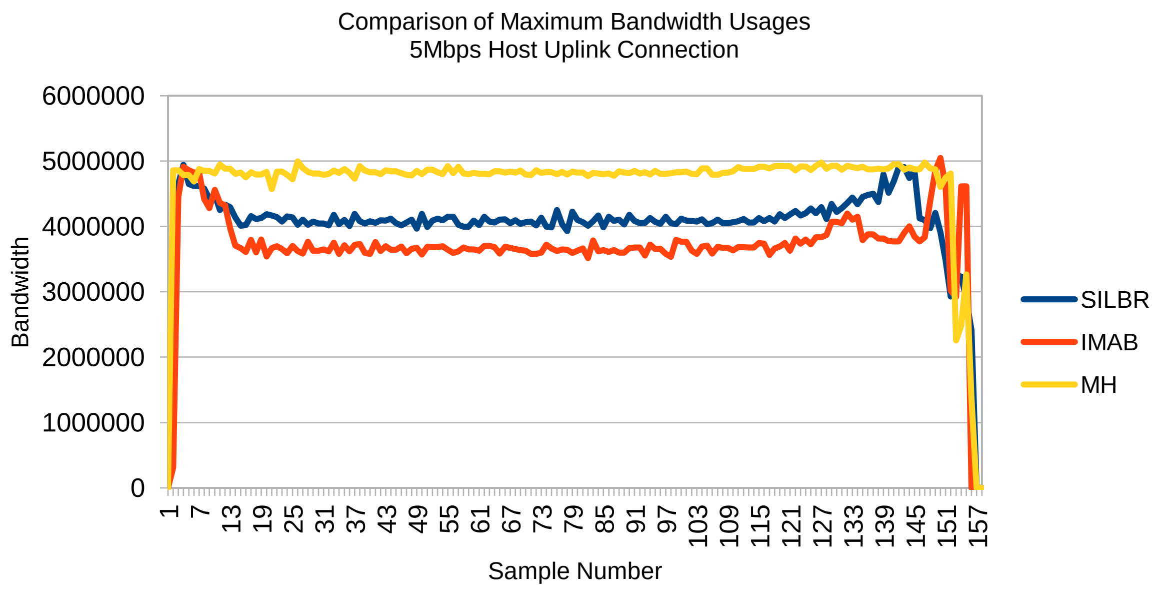


Figure 25 Maximum Bandwidth Usages for SILBR, IMAB and MH metrics, Mesh Topology 5 Mbps Host Uplink Connections

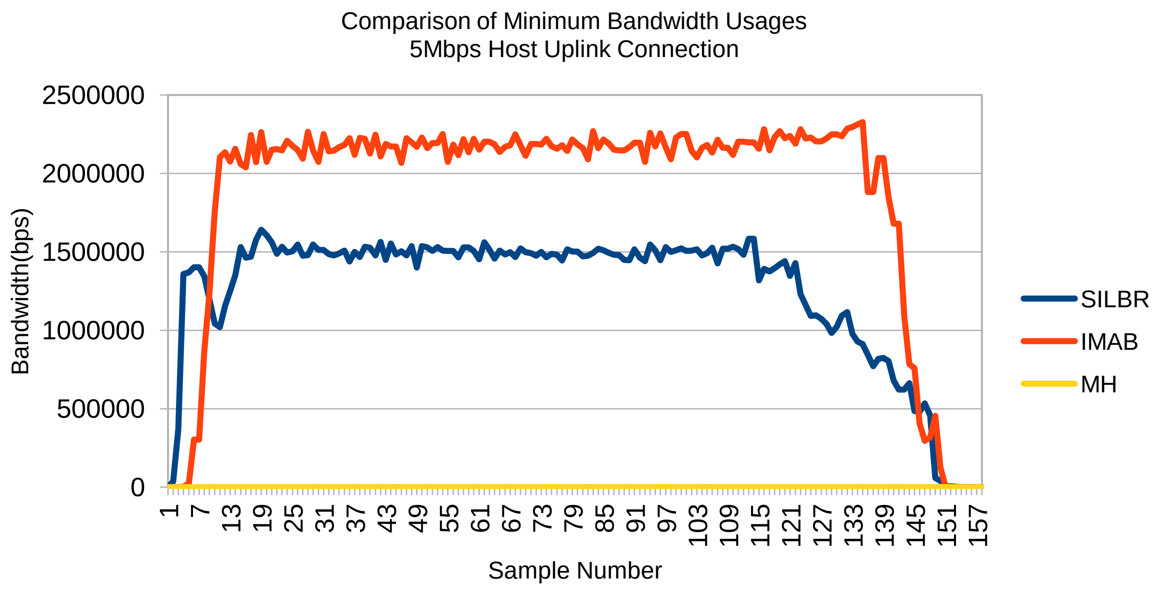


Figure 26 Minimum Bandwidth Usages for SILBR, IMAB and MH metrics, Mesh Topology 1 Mbps Host Uplink Connections

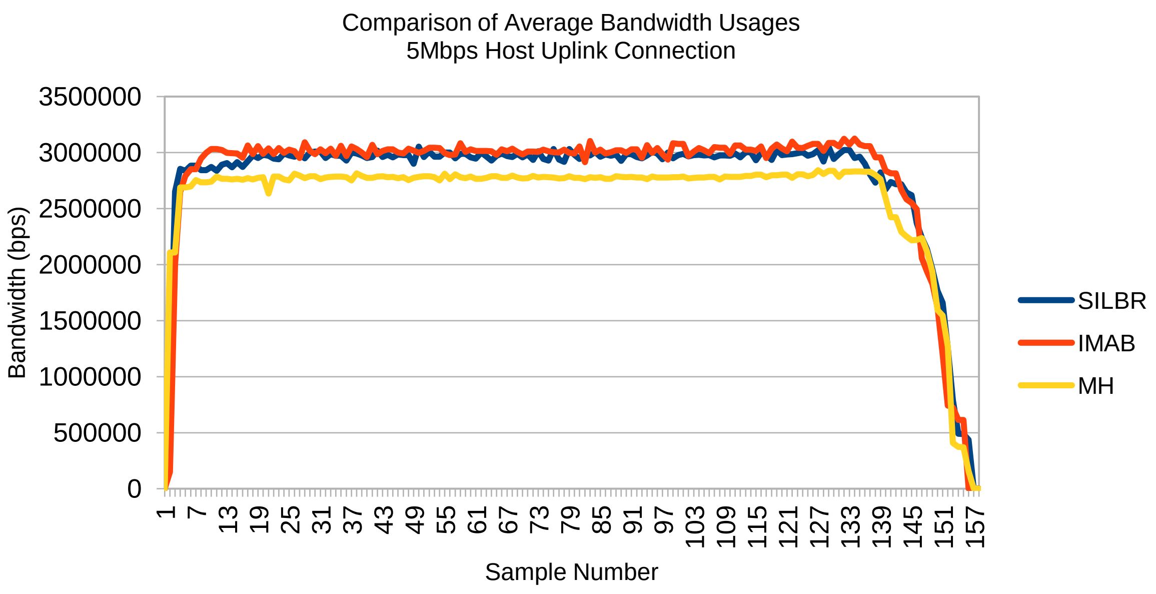


Figure 27 Average Bandwidth Usages for SILBR, IMAB and MH metrics, Mesh Topology 1 Mbps Host Uplink Connections

The comparison of maximum bandwidth usages shows that the SILBR achieves 20% decrease in the MLU and IMAB achieves 25% decrease in the MLU on average. In terms of minimum bandwidth usage, IMAB increases the bandwidth usage by 50% more than SILBR. Both algorithms achieve almost the same results in terms of average bandwidth usages.

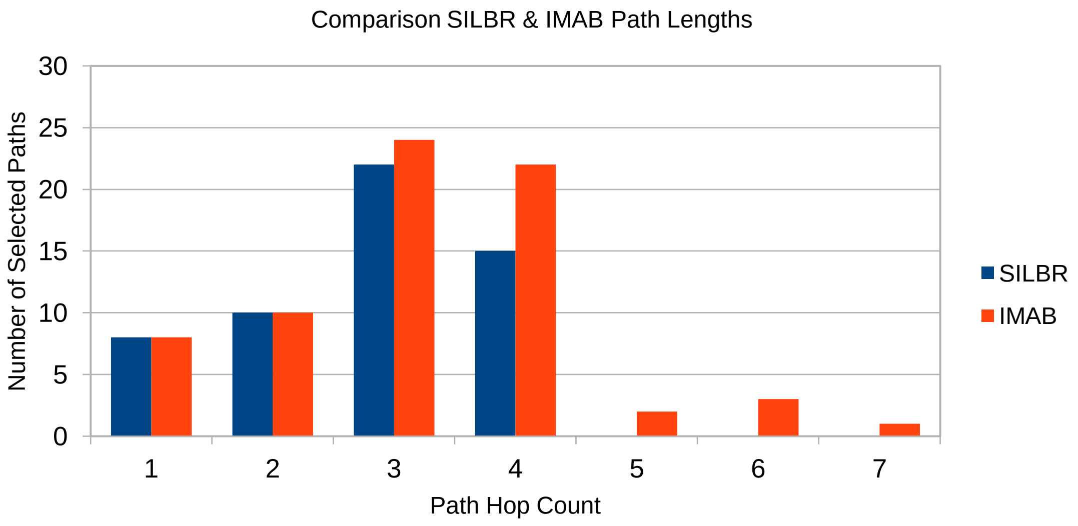


Figure 28 SILBR & IMAB Path Length Comparison 5 Mbps Host Uplink Connections – 5 Mbyte 200 Flows

As a side benefit of the SILBR metric, when results in Figure 21 and Figure 28 is considered, it can be seen that the SILBR requires less number of hops compared to IMAB which is another optimization for the SDN controller resource management. Another advantage of using shorter paths is that the SDN controller will need to send a smaller number of control messages to the network devices in order to control the traffic distribution.

For this test case, we also measured the path latency between the hosts by sending ping packages between the hosts. Figure 29 shows that, on average, the path latency when using SILBR metric is 18.5% less than IMAB metric. For this test, we again used the mesh topology and created flows at a rate of 10 flows/second for 20 seconds. Each flow carried 5 Mbyte data between randomly selected host pairs. After the flows started, ping messages are sent from host 1, host 2 and host 3 to all other hosts in the network. The path selection for the ICMP traffic is also made using SILBR or IMAB metrics for the respective test. The ping messages are stopped as soon as all the flows in the network are completed. The average latency is calculated as arithmetic average of latency values for each ping message.

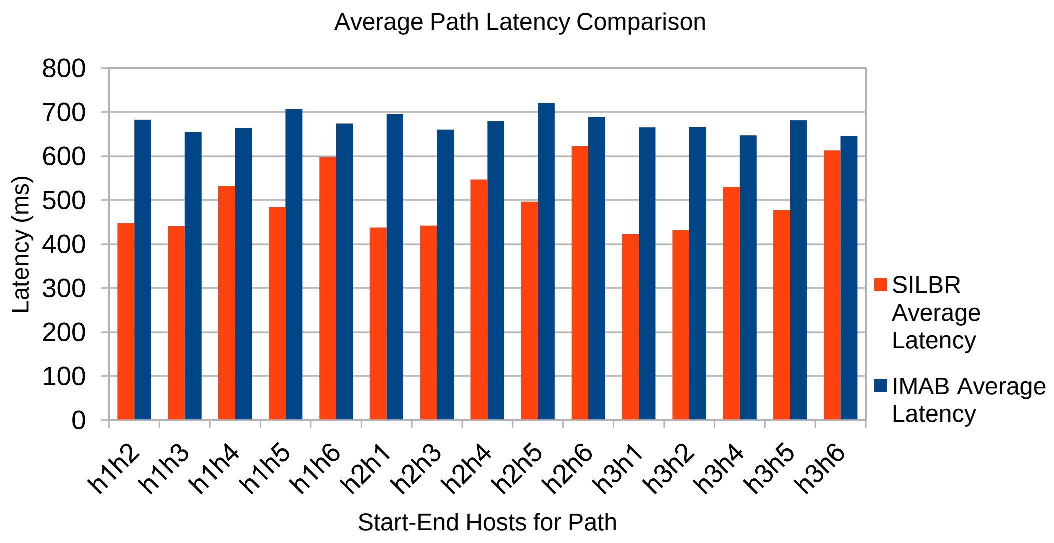


Figure 29 SILBR and IMAB Average Path Latency Comparison – 5 Mbyte 200 flows

CHAPTER 6

CONCLUSION

In this thesis, we focus on minimization of MLU for traffic engineering method. The minimization of MLU has many benefits. Minimizing link utilization leads to lesser queuing delay, improves the scalability of network and increases per flow throughput. Hence, by minimizing MLU, we both improve the scalability of the network and the performance of traffic flows.

Our proposed SILBR metric uses a statistical approach to the minimization problem. In general, the statistical approach, with the application of proper methods and analysis, leads to lesser data requirements for decision-making and helps understanding the nature of change in the data. The SILBR, uses the data collected from the links, combines this data with the information small amount of data from the network. The SILBR uses this combined information and Student's T distribution to infer the status of the link and presents this information as link cost. Thus, SILBR performs a proactive traffic management.

We tested our proposed SILBR metric and algorithm on SDN networks. We used the used the FloodLight SDN controller and its interfaces to measure the required performance characteristics of the network. In order to create the testing topologies, we used Mininet and MiniEdit. We only used software environments for testing. Firstly, we tested SILBR on a trivial topology to show that it works as intended. We then tested SILBR on a more complex topology to understand its benefits and compare to similar traffic engineering approaches.

Our experiment results show that SILBR distributes the traffic as intended and achieves near %20 decrease on average in the MLU. We tested SILBR routing against the MH routing method. Our results show that SILBR improves flow completion times by increasing the utilization of the links in the network. When SILBR routing is compared against IMAB, SILBR performance approaches IMAB in terms of minimization of maximum link utilization.

As a side benefit of SILBR, SILBR creates a smaller number of flows in the physical network and prefers shorter paths compared to IMAB. Our experiments have shown about 18.5 % decrease in the latency when SILBR and IMAB are compared under the same traffic conditions.

We evaluated the performance on a virtual network setting that emulates the tested topologies. As we conducted only software-based experiments, the experimentation has been carried out under suboptimal conditions. For example, only TCP traffic has been tested. We leave the further testing with more generalized topologies and real traffic models for future studies.

Our experiments have shown that SILBR algorithm is a promising approach to Traffic Engineering in SDN. Currently, SILBR utilizes a simple rerouting algorithm. It was tested only on uniform-bandwidth networks. For the future of the work in this thesis, we plan to generalize the routing metric proposed so that it can be used on non-uniform bandwidth networks and improve rerouting by involving approaches that utilizes machine learning based traffic determination methods.

REFERENCES

[1] R. Gallaher, “MPLS Traffic Engineering,” *Rick Gall. MPLS Train. Guid.*, pp. 107–126, 2003, doi: 10.1016/b978-193226600-9/50008-8.

[2] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, “The segment routing architecture,” *2015 IEEE Glob. Commun. Conf. GLOBECOM 2015*, 2015, doi: 10.1109/GLOCOM.2014.7417124.

[3] F. Ieee *et al.*, “Software-Defined Networking : A Comprehensive Survey,” vol. 103, no. 1, 2015.

[4] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “Research challenges for traffic engineering in software defined networks,” no. June, pp. 52–58, 2016.

[5] S. Delaët, S. Dolev, D. Khankin, and S. Tzur-David, “Make&activate-before-break for seamless SDN route updates,” *Comput. Networks*, vol. 147, pp. 81–97, 2018, doi: 10.1016/j.comnet.2018.10.005.

[6] W. Queiroz, M. A. M. Capretz, and M. Dantas, “An approach for SDN traffic monitoring based on big data techniques,” *J. Netw. Comput. Appl.*, vol. 131, pp. 28–39, 2019, doi: 10.1016/j.jnca.2019.01.016.

[7] A. A. Neghabi, N. J. Navimipour, M. Hosseinzadeh, and A. Rezaee, “Load Balancing Mechanisms in the Software Defined Networks: A Systematic and Comprehensive Review of the Literature,” *IEEE Access*, vol. 6, pp. 14159–14178, 2018, doi: 10.1109/ACCESS.2018.2805842.

[8] R. Challa, S. Jeon, D. S. Kim, and H. Choo, “CentFlow: Centrality-Based Flow Balancing and Traffic Distribution for Higher Network Utilization,” *IEEE Access*, vol. 5, pp. 17045–17058, 2017, doi: 10.1109/ACCESS.2017.2743697.

[9] H. Long, Y. Shen, M. Guo, and F. Tang, “LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks,” *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, pp. 290–297, 2013, doi: 10.1109/AINA.2013.7.

[10] H. Ren, X. Li, J. Geng, and J. Yan, “A SDN-Based Dynamic Traffic Scheduling Algorithm,” *2016 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov.*, no. 2, pp. 514–518, 2017, doi: 10.1109/cyberc.2016.103.

[11] “ISO - ISO/IEC 10589:2002 - Information technology — Telecommunications and information exchange between systems — Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473).” https://www.iso.org/standard/30932.html

(accessed Aug. 30, 2020).

[12] "RFC 2328 - OSPF Version 2." https://tools.ietf.org/html/rfc2328 (accessed Aug. 30, 2020).

[13] Y.-L. Lan, K. Wang, and Y.-H. Hsu, "Dynamic load-balanced path optimization in SDN-based data center networks," in *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Jul. 2016, pp. 1–6, doi: 10.1109/CSNDSP.2016.7573945.

[14] Y. C. Wang and S. Y. You, "An Efficient Route Management Framework for Load Balance and Overhead Reduction in SDN-Based Data Center Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1422–1434, 2018, doi: 10.1109/TNSM.2018.2872054.

[15] A. Lipson, S. . Lispson, and H. Lipson, "RSVP-TE: Extensions to RSVP for LSP Tunnels Status - RFC 3209," *https://tools.ietf.org/pdf/rfc3209.pdf*, 2001, doi: 10.1017/CBO9781107415324.004.

[16] H. Kamen, "An IETF Traffic Engineering Overview," *TLS - Times Lit. Suppl.*, no. 5795, p. 5, 2014, doi: 10.1075/jlp.17069.wei.

[17] D. Y. N. D. Katz,K. Kompella(Juniper Networks), "Traffic Engineering (TE) Extensions to OSPF Version 2 Status," *https://tools.ietf.org/html/rfc3630*, 2003, doi: 10.1017/CBO9781107415324.004.

[18] T. L. H. Smit(Procket Networks), "Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE) - RFC 3784," *https://tools.ietf.org/html/rfc3784*, doi: 10.1017/CBO9781107415324.004.

[19] B. C. Isaak and R. Planning, "A Comparison of Approaches for Traffic Engineering in IP and MPLS Networks," pp. 142–155, 2011, doi: 10.1002/fut.

[20] "The Use of RSVP with IETF Integrated Services Status - RFC 2210," 1997, [Online]. Available: https://tools.ietf.org/html/rfc2210.

[21] R. Braden, D. Clark, and S. Shenker, "RFC-1633: Integrated Services in the Internet Architecture: an Overview Status of this Memo," *Internet Res.*, pp. 1–28, 1994.

[22] "Open Networking Foundation is an operator led consortium leveraging SDN, NFV and Cloud technologies to transform operator networks and business models." https://www.opennetworking.org/ (accessed Aug. 02, 2020).

[23] "RFC 7426 - Software-Defined Networking (SDN): Layers and Architecture Terminology." https://tools.ietf.org/html/rfc7426 (accessed Aug. 02, 2020).

[24] "RFC 5810 - Forwarding and Control Element Separation (ForCES) Protocol Specification." https://tools.ietf.org/html/rfc5810 (accessed Aug. 02, 2020).

[25] N. Mckeown, T. Anderson, L. Peterson, J. Rexford, S. Shenker, and S. Louis, "Sigcomm08_Openflow.Pdf," vol. 38, no. 2, pp. 69–74, 2008, doi: 10.1145/1355734.1355746.

[26] H. Song, "Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane," in *HotSDN 2013 - Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 2013, pp. 127–132, doi: 10.1145/2491185.2491190.

[27] "Open Networking Foundation is an operator led consortium leveraging SDN, NFV and Cloud technologies to transform operator networks and business models." https://www.opennetworking.org/ (accessed Jul. 23, 2020).

[28] "SDN | Flowgrammable." http://flowgrammable.org/sdn/ (accessed Jul. 23, 2020).

[29] "SDN / OpenFlow / Data Model | Flowgrammable." http://flowgrammable.org/sdn/openflow/data-model/ (accessed Sep. 12, 2020).

[30] Open Networking Foundation, "OpenFlow Switch Specification (Version 1.5.1)," *Current*, vol. 0, pp. 1–36, 2015, [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

[31] "CHAPTER 8: Statistical Inference: Estimation for Single Populations - Business Statistics: For Contemporary Decision Making, 8th Edition." https://learning.oreilly.com/library/view/business-statistics-for/9781118494769/18_chapter-08.html#ch08 (accessed Aug. 29, 2020).

[32] "File:Normal distribution and scales.gif - Wikimedia Commons." https://commons.wikimedia.org/wiki/File:Normal_distribution_and_scales.gif (accessed Sep. 12, 2020).

[33] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Drill: Micro load balancing for low-latency data center networks," *SIGCOMM 2017 - Proc. 2017 Conf. ACM Spec. Interes. Gr. Data Commun.*, pp. 225–238, 2017, doi: 10.1145/3098822.3098839.

[34] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling mix-flows in commodity datacenters with Karuna," *SIGCOMM 2016 - Proc. 2016 ACM Conf. Spec. Interes. Gr. Data Commun.*, pp. 174–187, 2016, doi: 10.1145/2934872.2934888.

[35] V. D. Chakravarthy and B. Amutha, "A novel software-defined networking

approach for load balancing in data center networks," *Int. J. Commun. Syst.*, no. September, pp. 1–16, 2019, doi: 10.1002/dac.4213.

[36] Z. Shu *et al.*, "Traffic Engineering in Software-Defined Networking: Measurement and Management," *IEEE Access*, vol. 4, pp. 3246–3256, 2016, doi: 10.1109/ACCESS.2016.2582748.

[37] T. Semong *et al.*, "Intelligent load balancing techniques in software defined networks: A survey," *Electron.*, vol. 9, no. 7, pp. 1–24, 2020, doi: 10.3390/electronics9071091.

[38] P. By and M. Z. A Ali, "Part 4: Information Theory (Lectures-2017)."

[39] A. Kumar, D. Manjunath, and J. Kuri, "Adaptive Bandwidth Sharing for Elastic Traffic," in *Communication Networking*, Elsevier, 2004, pp. 323–433.

[40] "Floodlight Controller - Project Floodlight." https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview (accessed Sep. 16, 2020).

[41] V. Protocol, "OpenFlow Switch Specification," vol. 1, pp. 1–283, 2015.

[42] Mininet, "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet," *Mininet.Org*, p. www.mininet.org, 2014, Accessed: Sep. 30, 2020. [Online]. Available: http://mininet.org/.

[43] "mininet/miniedit.py at master · mininet/mininet · GitHub." https://github.com/mininet/mininet/blob/master/examples/miniedit.py (accessed Sep. 30, 2020).

[44] "GitHub - esnet/iperf: iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool." https://github.com/esnet/iperf (accessed Sep. 16, 2020).

[45] D. Manual, A. Botta, W. De Donato, A. Dainotti, S. Avallone, and A. Pescap, "D-ITG 2.8.1 Manual," pp. 1–35, 2013.

APPENDICES

APPENDIX A

PERFORMANCE EVALUATION RESULTS

This section includes test results mentioned in section 5.2.

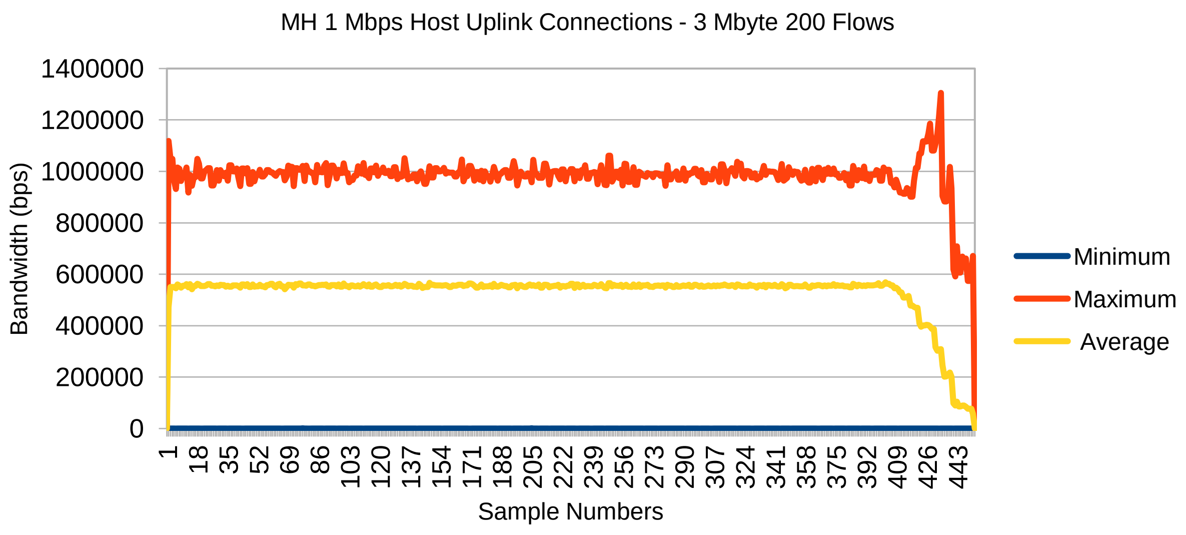


Figure 30 MH with 1 Mbps Host Uplink Connections - 3 Mbyte 200 Flows

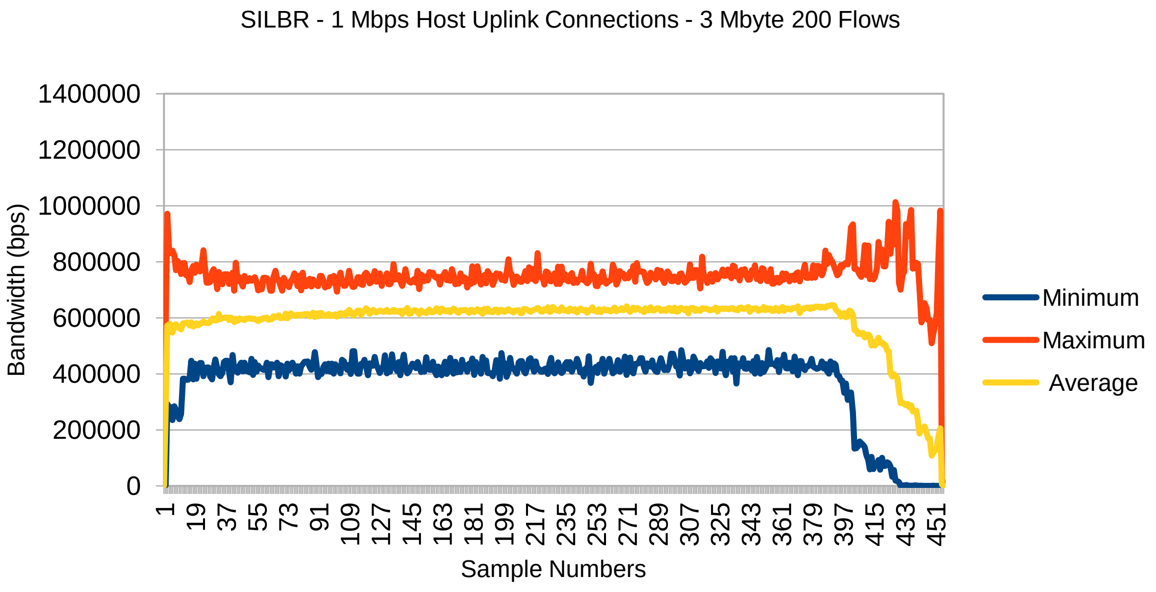


Figure 31 SILBR with 1 Mbps Host Uplink Connections - 3 Mbyte 200 Flows

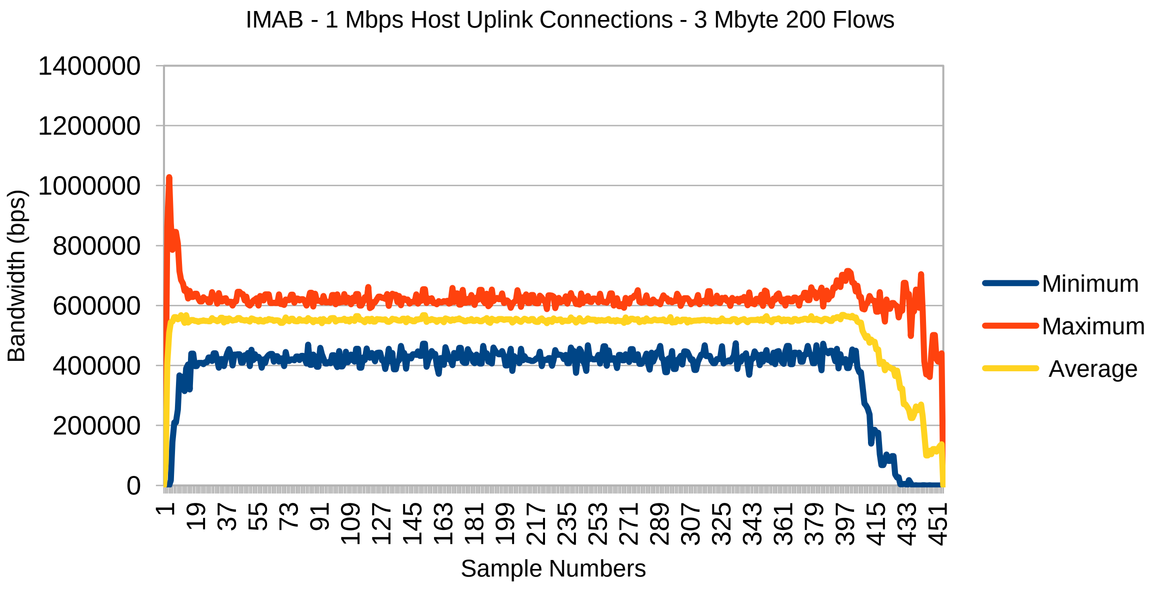


Figure 32 IMAB with 1 Mbps Host Uplink Connections - 3 Mbyte 200 flows

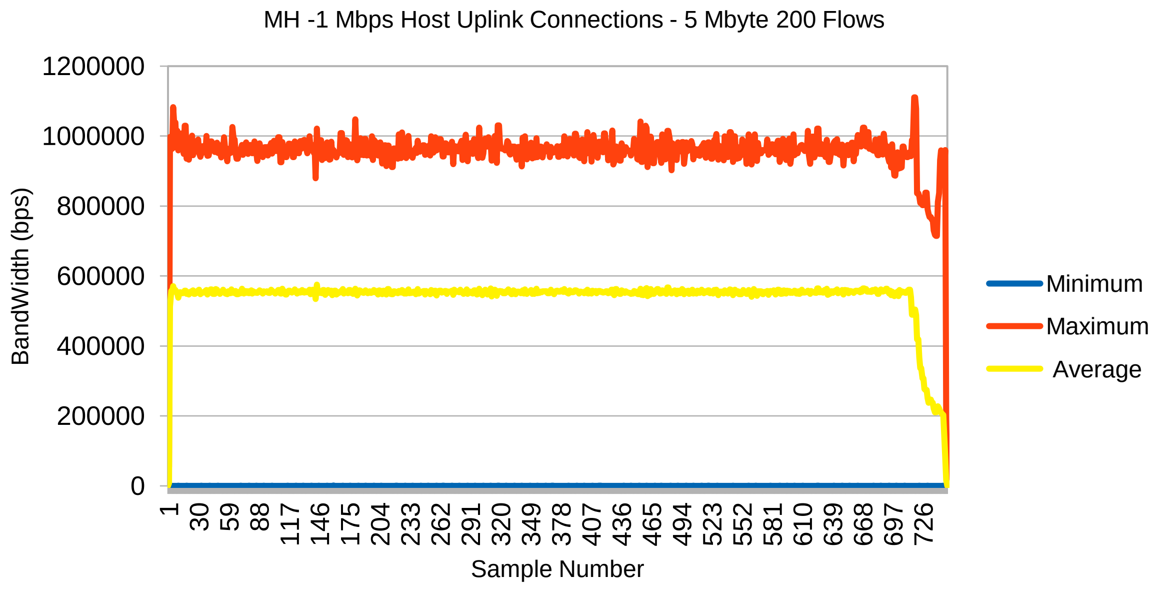


Figure 33 MH - 1 Mbps Host Uplink Connections - 5 Mbyte 200 Flows

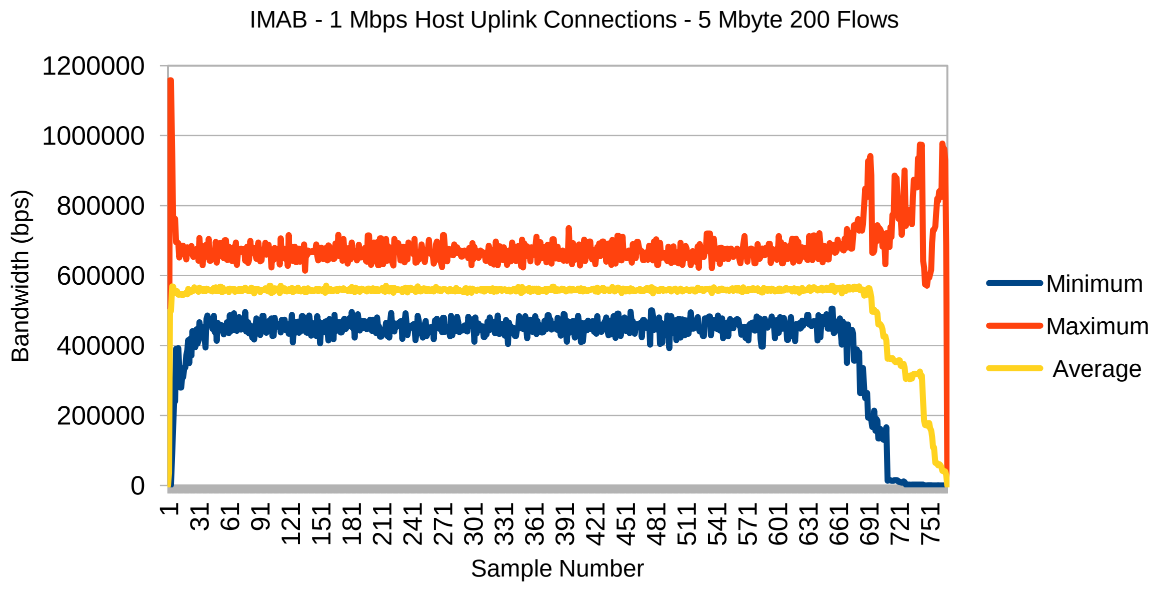


Figure 34 IMAB with 1 Mbps Host Uplink Connections - 5 Mbyte 200 Flows

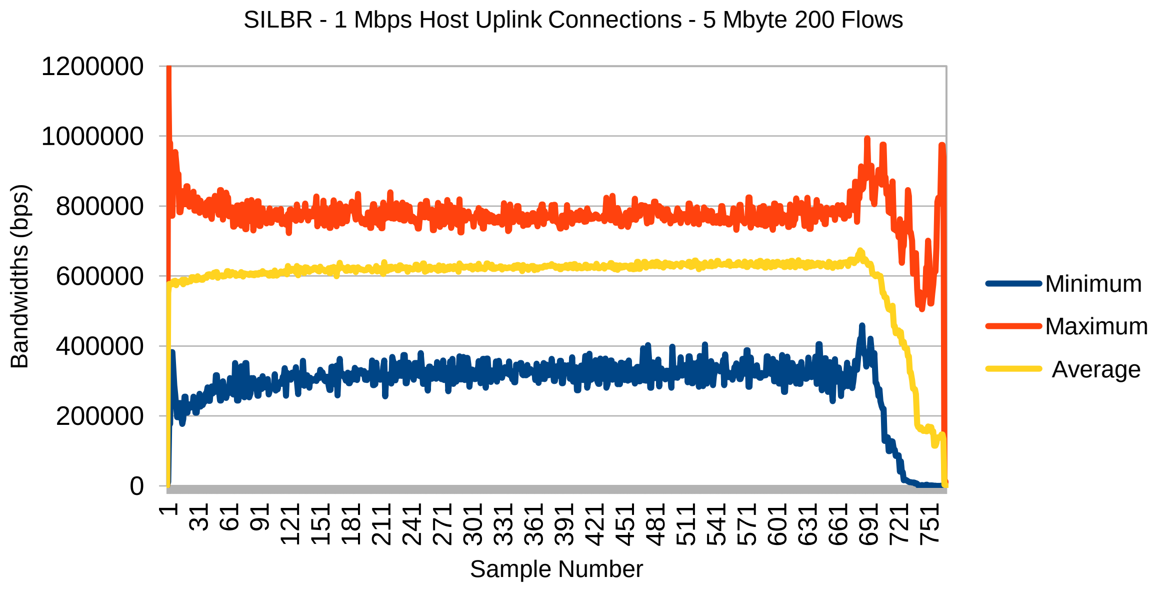


Figure 35 SILBR with 1 Mbps Host Uplink Connections – 5 Mbyte 200 Flows

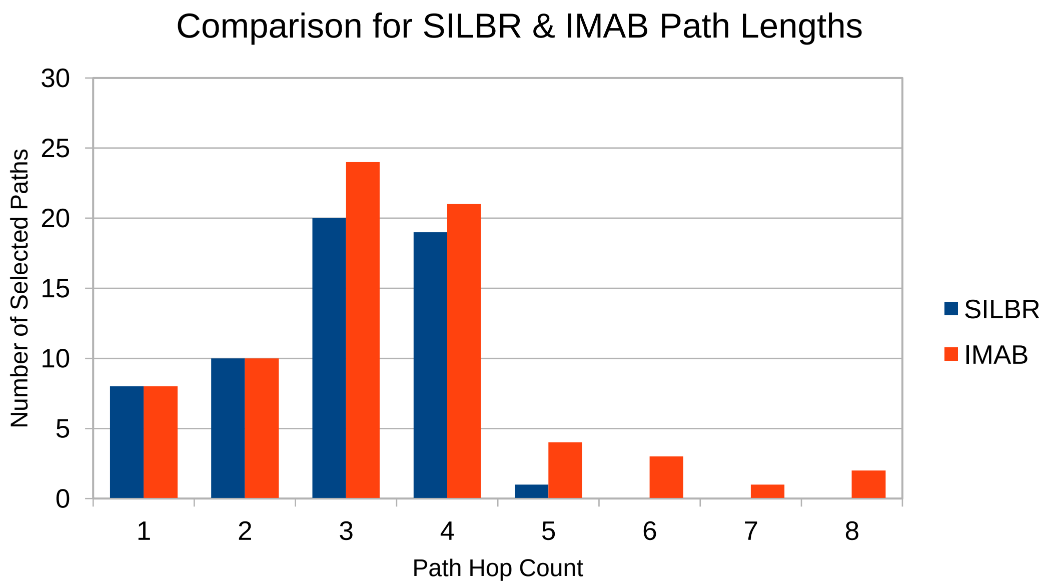


Figure 36 SILBR & IMAB Path Length Comparison 1 Mbps Host Uplink Connections – 3 Mbyte 200 Flows

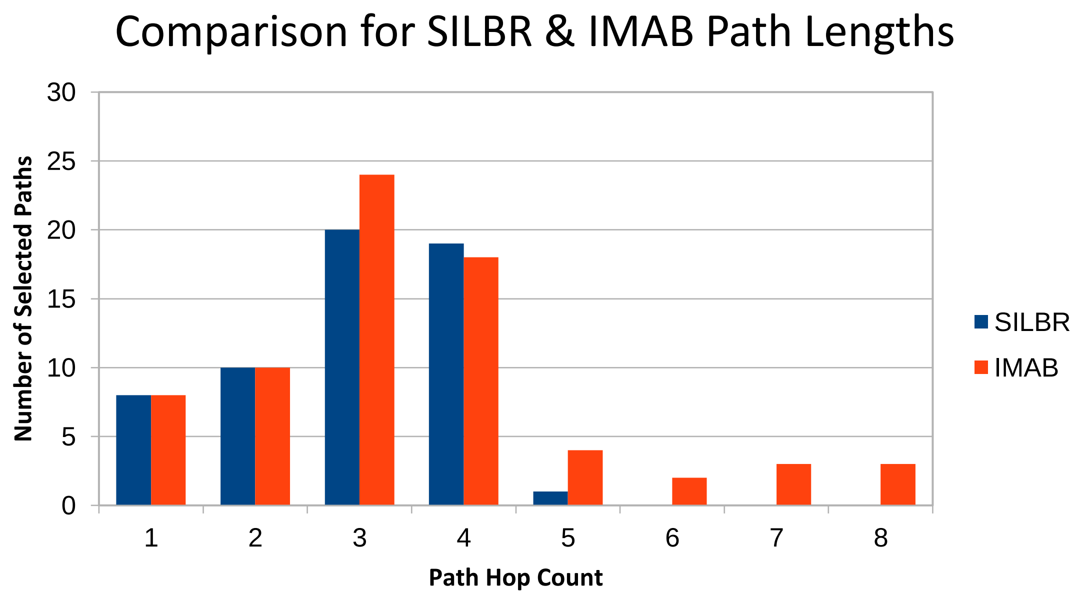


Figure 37 SILBR & IMAB Path Length Comparison 1 Mbps Host Uplink Connections – 5 Mbyte 200 Flows