Network Impact Modeling and Analysis: A QoS Perspective

Tarandeep K. Randhawa, *The University of Western Ontario*

Supervisor: Haque, Anwar, *The University of Western Ontario*

Abstract

The International Data Corporation (IDC) estimated that total digital data created, replicated, and consumed was 4.4 Zettabytes (ZB) in the year 2013, 8 ZB in 2015, and predicted to reach 40 ZB by 2020. This massive amount of internet traffic put a great overhead on network capacity which may impact network Quality of Service (QoS) such as latency, jitter, throughput, packet loss, and load balancing. From the Internet Service Provider's (ISP's) perspective, understanding the possible impact of the future internet traffic on its network is critical for provisioning their network capacity in a cost-effective manner while meeting network QoS requirements. In order to achieve the above goal, one needs a framework that is capable of taking input from the traffic forecast, assign traffic load over the networks, and then identify the impact on the existing traffic QoS status (latency, jitter, packet loss, throughput, etc. In this paper, we developed a network planning framework namely Network Impact Modelling and Analysis (NIMA) that uses novel methods and techniques to alert network planners on the links that are subject to a high-risk group in terms of congestion, indicates the impact on network-wide QoS metrics, and finally suggests an optimal routing strategy that can improve the overall network health. As part of this optimal routing task, we used Yen's algorithm which showed performance improvement when compared with Dijkstra's algorithm and Suurballe's k-disjoint algorithm. For simulation purposes, we used Mininet in a combination with a floodlight controller for implementation. The experiments are performed on different sized topologies to test the effectiveness of our proposed framework.

Keywords

Quality of Service (QoS), network traffic forecast, Mininet, Floodlight controller, load balancing, network congestion, multipath routing.

Summary for Lay Audience

The rapid growth in Internet-based services and applications results in a dramatic increase in internet traffic. A huge amount of data in the form of text, video, and real-time streaming results in an excessive increase in network traffic. As a result, congestion occurs over the network which impacts the network's Quality of Service (QoS). Internet Service Providers (ISPs) need cost-effective models for timely detection of risk-prone network links to handle the congestion in order to maintain network health.

In this thesis, we develop a traffic model to analyze the impact of forecasted traffic on the network's QoS metrics i.e. network health. Our proposed framework namely Network Impact Modelling and Analysis (NIMA) evaluates and analyses the congestion level of the network due to overlaying forecasted traffic on the existing network traffic. Furthermore, NIMA also evaluates the QoS parameters such as Latency, Throughput, Jitter, Packet loss, Link utilization, and Load Balancing for both current and forecasted traffic. NIMA identifies high-risk (highly utilized) links and alerts the network planners when such links are detected. In this way, NIMA can be used by network planners as an aid in decision making and policy designing. Finally, to improve the overall health of the network, we suggest an optimal routing algorithm.

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisor, Dr. Anwar Haque who provided me with the opportunity to conduct this research. I am thankful to Dr. Haque for his continuous support, advice, guidance, and encouragement throughout my Master's program at the University of Western Ontario.

I would also like to acknowledge Yaser Al Mtawa for his technical assistance in Mininet simulation and Yasmeen Ali for her participation in various technical discussions throughout this journey.

Finally, I thank my family for their constant encouragement and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

1 Introduction

The digitization of data in recent years and associated next-generation services lead to a mammoth growth in internet traffic. Every day a huge number of devices are being interlinked with the internet. Total Internet traffic has experienced dramatic growth in the past two decades. In 1992, global Internet networks carried approximately 100 GB of traffic per day. Ten years later, in 2002, global Internet traffic amounted to 100 Gigabytes per second (GB/second). In 2017, global Internet traffic reached more than 45,000 GB/second. [1] According to the Cisco VNI Global IP Traffic Forecast [1], the number of devices connected to IP networks will rise to three times more than the global population in 2022. As a result, the number of global internet users will increase from 3.4 billion in 2017 (45% of the WorldWide (WW) population) to 4.8 billion (60% of the WW population) by 2022. As per this estimate, Machine-to-Machine(M2M) connections will grow to approximately 14.6 billion and the annual global traffic will reach 4.8 zettabytes per year by 2022. The total traffic produced by wireless and mobile devices will account for 71 percent by 2022 with smartphones alone contributing 44 percent of total traffic [1]. From 2017 to 2022, the global IP video traffic growth will be estimated at about four-fold and the traffic generated from Virtual Reality (VR) and Augmented Reality (AR) will be going to increase approximately 12-fold [1]. Overall, IP traffic will increase at a Compound Annual Growth Rate (CAGR) [1] of 26 percent from 2017 to 2022. While the total human participation on the internet is limited to the World’s population size, the Internet of Things (IoT) is vastly bigger with the potential to add billions of devices with no upper limit. However, IoT does not assume a specific communication technology and is making great progress along with the integration of Wireless Sensor Networks (WSN). Sensory data generated from various sensors such as temperature sensors, humidity sensors, pressure sensors, gyro sensors, etc. plays a great role in the development of the IoT environment. This tremendous growth in network traffic makes it difficult for Internet service providers (ISPs) to plan network growth in a cost-effective manner; and poses challenges towards maintaining their offered services with the required quality of service. A correct estimate of forecasted traffic impact on network-wide latency and other

Quality of Service (QoS) parameters can be helpful for ISP's in provisioning their network capacity.

1.1 Motivation and Objective

The problem arises when the massive amount of Internet traffic puts a great overhead on network capacity which may impact Quality of Service (QoS) such as congestion, latency, throughput, jitter, packet loss, and load balancing [2,3]. To ensure uninterrupted access and an accepted level of QoS, the ISP's need to know about high-risk links that may not be able to handle the passing network traffic and hence may fail. The QoS parameters analysis based on the real-time traffic may only be helpful in identifying the links that are already at an edge of failure. The early identification of risk-prone vulnerable links may help the network planners in provisioning their network capacity in a cost-effective manner while meeting network QoS requirements. From the ISP's perspective, understanding the possible impact of the future internet traffic on its network is critical for the early identification of high-risk links. Most of the existing approaches [5] analyze the QoS parameters based on the current traffic and don't account for the impact of forecasted traffic. A framework is required that can analyze the impact of forecasted traffic on the QoS parameters and help in identifying the vulnerable links. The main objective of this thesis is to design a framework that is able to take input from the traffic forecast and able to analyze the impact on network-wide latency, throughput, jitter, packet loss, utilization and finally suggests an optimal routing strategy that can improve the overall network health.

1.2 Contribution

We developed a network planning framework, namely, Network Impact Modelling and Analysis (NIMA) that predicts the congestion level of the network, alerts the network planners on the links that are subject to the high-risk group and indicates the impact on network-wide latency, jitter, packet loss, link utilization, throughput. The novel contribution of this thesis is given below:

- To the best of our knowledge, this is the first work that measures and analyzes all major QoS parameters based on forecasted traffic by taking current network scenarios into account and suggests an optimal routing for the load-balanced network.

- We proposed utilization-based link-status categories that can be used to categorize links on a scale of low-to-high failure risk. The network planner is notified about high-risk links.

- We present a general framework that can be used by ISPs to understand the possible impact of traffic on the network status.

- We analyzed the multi-path routing based Yen's algorithm and showed how it improved the network load balance. We also compared the performance of Yen's algorithm with Dijkstra's single shortest path algorithm and Suurballe's k-disjoint algorithm.

We initialized our experimental setups with current traffic scenarios and evaluated the impact of the new traffic on the network by analyzing the before and after scenarios focusing on the QoS metrics such as packet loss, latency, jitter, throughput, link utilization, and load balancing. After routing the new traffic in the network, evaluating and analyzing the individual QoS parameters is helpful in identifying high-risk links that need attention by the network planners. For Link Utilization, we proposed a link status category based on utilization level and evaluated the change in the status when forecasted traffic is simulated on the network initialized with current traffic parameters. We also monitor other QoS parameters such as latency, packet loss, throughput, and jitter as part of the NIMA framework. For load balancing, we applied a multi-path routing algorithm known as Yen's algorithm. We also compared the performance evaluation of Yen's algorithm with Suurballe's k-disjoint algorithm taking into consideration average latency, average throughput and load balancing as metrics.

1.3 Thesis Structure

In chapter 2, we provide a literature review of the network concepts used throughout the study. In chapter 3, we formally defined the problem statement and discuss in detail the existing solutions proposed in the past to address the problem. In chapter 4, we discuss our proposed methodology in detail. The proposed framework design and methods used to collect and compare various statistics are discussed in detail. In chapter 5, we discuss our experiments and results. We showed and analyzed the statistics calculated using our proposed framework. Also, we showed the comparison of our proposed framework with the existing methods. In chapter 6, we conclude our findings and shed light on future directions.

Chapter 2

2 Review of Network Concepts

ISP’s rely on accurate QoS parameters analysis for network resources and financial planning. A reliable and accurate measure of QoS parameters based on the forecasted traffic can be helpful in identifying high-risk network links that need attention/possible capacity upgrade. To address the need for identification of highly-utilized links, we proposed a model that evaluates the majority of the QoS parameters for forecasted traffic. In this chapter, we briefly discuss the various network concepts that are used throughout this study.

2.1 Network Traffic

The amount of data that moves across a network at a given point of time is referred to as network traffic [6]. In general, network topologies are implemented based on network traffic. Data is encapsulated in packets to pass as traffic over the network links. In our experiments, we generated the network traffic as the Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) packets. To ensure the QoS, proper organization of network traffic is desired in a given network. In our experiment we use two different types of traffic flows – one flow as probe packets that employs ICMP and second is the traffic generated using TCP. TCP sequence diagram [7] shown in figure 2.1 represents the three phases of TCP between client and server. The client begins the communication with the server by first establishing a connection with the client by sending the SYN packet to initiate the connection establishment phase. The connection establishment phase performs a three-way handshake. Once the connection is established, the client performs data transfer. Once the data transfer phase is over, the client sends FIN packet to the server indicating initiation of the connection termination phase. If the server has no more data to send, then the server also sends FIN packet to the client indicating connection close from its end. Figure 2.2 shows the working of the ICMP protocol generated using the ping tool. The objective of ICMP [8] is to measure the Round-trip time. ECHO request is sent by the client and the ECHO reply is the response by the server.

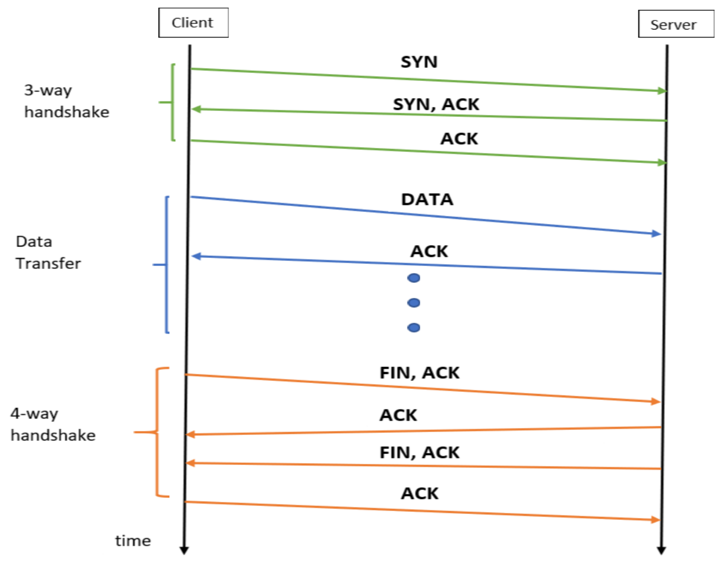


Figure 2.1: TCP sequence diagram

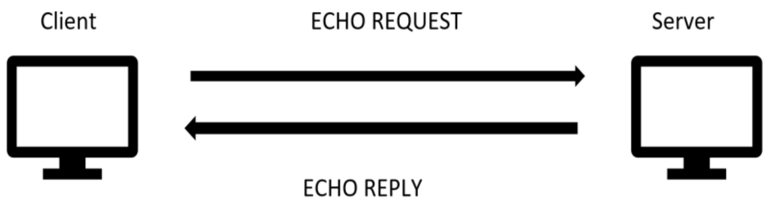


Figure 2.2: ICMP diagram

2.2 Congestion

With the development of technology, the number of internet users, as well as devices, is one the rise. This results in the exponential growth of internet traffic that can further cause congestion. Congestion arises when network traffic inflow is more than the underlined limited bandwidth a link can support. Link failures in the network can also cause congestion at other links. The routing technique also plays an important role in handling congestion. Basic routing techniques based on the single shortest path are more prone to congestion and hence link failures because all packets from a source to a sink try to follow the same path. This enforces more load on the same links every time and increases the problem of network congestion. It is important to prevent congestion because important data packets may be lost and never be recovered [9]. We evaluated the impact of forecasted traffic on the network to identify the highly congested links and also showed how a multi-path routing approach helped in handling the congestion.

2.3 Quality of Service (QoS)

QoS is defined as the measure of service quality that the network offers to the end-users [4]. The impact of internet traffic on network performance can be analyzed by measuring the Quality of Service parameters. Various QoS parameters are explained as follows:

2.3.1 Packet Loss

Packet loss occurs where IP packets are unable to reach from source to destination. Packet loss is defined as a percentage of lost packets in comparison to the total transmitted packets [5]. There are various reasons for packet loss in a network that include congestion, inefficient routing techniques, limited memory at nodes, etc. The packet loss is calculated as:

$$PacketLoss = \frac{Packets\ transmitted - Packets\ received}{Packets\ transmitted} \times 100 \tag{2.1}$$

2.3.2 Latency

Latency is the transit time taken by a packet to reach from source to destination and is calculated as the ratio of a packet size to the link bandwidth [5,10,11]. The formula for latency is:

$$Latency = \frac{Packets\ size}{Link\ bandwidth} \tag{2.2}$$

The various types of delays are:

a. Packetization delay: It is a delay caused during the formation of IP packets in a process.

b. Queuing delay: It is the time a packet waits in a queue while the router is handling packet transmission along with a network.
c. Propagation delay: When a packet is traveling in a transmission medium like copper, coax, etc., the amount of time it takes to travel from the sender to receiver is called propagation delay.
d. Transmission delay: It is also known as store-and-forward delay, the time required to push all the bits of the packets onto the network.
e. Processing delay: The time a host or router needs to process an incoming packet by reading a packet header and searching for a routing table in order to determine the next node for packet forwarding.

2.3.3 Jitter

Jitter is defined as the variation of the delay (in seconds) i.e. when the end to end delay is not constant [5,11,12]. Jitter can be caused because of the variation in traffic load or congestion in the network. The formula for Jitter is given as:

$$Jitter = \frac{sum\ of\ variation\ of\ delay}{sum\ of\ total\ packet\ received} \tag{2.3}$$

2.3.4 Throughput

It is defined as the number of packets processed through the network in a unit of time (in bps) [5,10,14]. Throughput is related to the bandwidth available. Higher link bandwidth results in higher throughput. The throughput is calculated as:

$$Throughput = \frac{Total\ data\ sent}{Delivery\ time} \tag{2.4}$$

2.3.5 Utilization

Monitoring network utilization can help us understand whether the network is idle, normal or busy. Network utilization is the ratio of current network traffic to the maximum traffic that the port can handle. When network utilization exceeds the threshold value, it will cause low transmission speed, delay and so on [11]. The formula for calculating utilization (%) is:

$$Utilization = \frac{data\ bits}{bandwidth \times time\ interval} \times 100 \tag{2.5}$$

2.4 Load Balancing

The exponential growth of network traffic results in a scarcity of network resources and difficulty in meeting the requirements of Quality of Service. Load Balance is another main factor that gets affected by the excessive amount of internet traffic. It is a technique where the workload is divided into multiple resources in order to prevent congestion and overcrowding on any particular resource [13]. However, load balancing does not mean to equally distribute load among all the nodes of the network, rather its emphasis on the current status of the network in order to balance the load on particular nodes and links. In order to distribute the information flows in a network load balancing makes the use of already existing parallel paths between input and output nodes. Load balancing benefits in high throughput, reliable, efficient, disruption-free network, optimizing traffic, lesser down the time of response and many more. There are different methods to achieve load balancing in a network that are categorized as static load balancing and dynamic load balancing. In static load balancing, traffic is divided equally to each resource within the server. The load is distributed during compile time. This technique is suitable for systems with low load variations and can be inefficient as it is not capable of predicting the behavior of the user. In dynamic load balancing, network traffic is dynamically distributed. Dynamic load balancing is more efficient because the load is distributed during run time.

2.5 Software-Defined Network

The main idea of the Software-Defined Network (SDN) [14] is to manage and monitor the network in a software-defined environment. Unlike, traditional networks, SDN networks are more flexible by allowing the separation of control plane and data plane in network devices. The control plane consists of the controller and makes a decision about the handling of traffic. Data plane includes switches which assist in forwarding traffic according to control plane decisions. In SDN, instead of using distributed control protocols, a centralized controller is used to configure forwarding elements remotely. The architecture of SDN consists of three layers:

1. Application layer: The Application layer consists of applications of networks in which new network features are introduced like management, security features, forwarding schemes, etc.

2. Control Layer: The control layer consists of a controller. The software application acts as a centralized entity to control the network. This allows the management of all functions of the network like topology management, handling of traffic between switches, security, etc.

3. Forwarding Layer: The forwarding layer consists of Network devices such as switches, routers, and flow tables.

The communication between layers is done through Northbound interfaces and southbound interfaces. Northbound interfaces are the application program interfaces used to communicate with the application layer and control layer. Southbound interfaces are the application program interfaces used to communicate between forwarding devices such as hubs and switches and controlling devices i.e. between forwarding plane and control plane. In this thesis, we used SDN based Floodlight controller for all our network simulations.

2.6 Network Topology

To design and implement the framework, we first create a network topology. Network topology or network architecture is a structural arrangement of nodes in a network that decides how nodes are connected and communicated with each other. Topologies can be physical or logical topologies. In physical topologies, nodes are connected and communicated through wired medium and transmit data through actual cables. Logical topology nodes communicate without the actual presence of the physical medium. There are different types of topologies based on how nodes are structurally arranged as discussed below:

2.6.1 Bus Topology

In a Bus topology, there is a central cable and all nodes are connected to the central wire. The bus network is easy to install and more appropriate for small networks. However, the whole network shuts down if wire breaks or a node fails.

2.6.2 Star Topology

In a Star topology, there is a central node called a hub and all other nodes are connected to the central hub in star-like fashion. It is useful because the failure of one node doesn't affect other nodes. The failure of the central node results in failure of the whole network.

2.6.3 Ring Topology

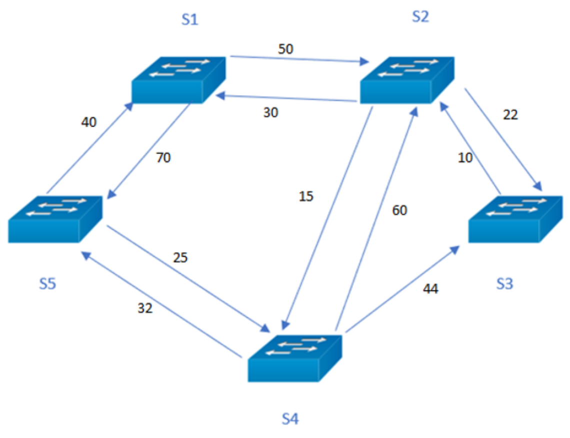
In a Ring topology, nodes are arranged in a circular order and information is passed to each node in one direction. This type of node arrangement is easy to install and handle. Ring topology is also better for handling the high-volume traffic. On the other side, it is difficult to add nodes to this type of arrangement and if one node fails the whole network goes down.

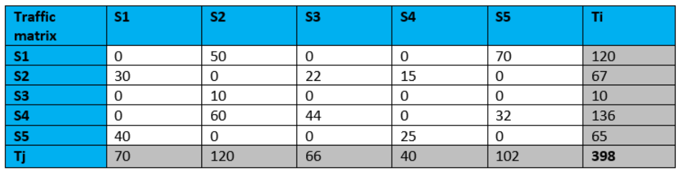
2.6.4 Mesh Topology

In Mesh topology, nodes are interconnected with each other. There are two types of Mesh topologies: full-mesh topology and partial-mesh topology. In full-mesh topology, each node is connected to every other node in a network. Every device has a direct connection with each other so the network can handle a high volume of traffic. In this type of topology, if there is an event of failure of one or two nodes, they won't affect the rest of the network. Unlike full-mesh topology, in partial-mesh topologies, each node is not connected to every other node in the network. However, these are less expensive than the full-mesh network and have less redundancy [15]. For our experiments, we used five different sized topologies having a difference in the number of nodes and interconnected as a full-mesh and partial-mesh network.

2.7 Traffic Matrix

It is the amount of data transferred between every pair of network entry/exit points. The entry and exit points are also known as source-destination pairs. The entry and exits points can be routers or Point of Presence (POP) or exit and entry links. The traffic matrix acts as an important input element for network planning. It is a big factor to obtain the current status of the network, to build the network by deciding the capacity design and topology structure and to fulfill many network engineering tasks like predict the future trends of the network, to do network optimization, designing of protocols and many more. The traffic matrix also gives information about the routing by checking the volume of demand on each link pair. It is also important to have an accurate traffic matrix in order to do failure management, service mediation, balancing of load, etc. The common method to estimate the traffic matrix is by obtaining information about link traffic and routing. Link traffic information is obtained easily from the routers using Simple Network Management Protocol (SNMP) and information about the routing of each traffic matrix on various links is obtained through the network design phase or by various routing simulation techniques [16].





Traffic matrix	S1	S2	S3	S4	S5	Ti
S1	0	50	0	0	70	120
S2	30	0	22	15	0	67
S3	0	10	0	0	0	10
S4	0	60	44	0	32	136
S5	40	0	0	25	0	65
Tj	70	120	66	40	102	**398**

Figure 2.3: An example of traffic flow in a network and its corresponding traffic matrix

Figure 2.1 shows an example of traffic flow in a network that consists of five nodes (S1, S2, S3, S4, S5) and its corresponding traffic matrix. For any network link (S_a, S_b), where $1 \leq a, b \leq 5$, a value in a traffic matrix indicates the traffic passing through that link. Any cell (S_a, T_i), where $1 \leq a \leq 5$, indicates total outgoing traffic from the node S_a and any cell (S_a, T_j), indicates total incoming traffic from the node S_a. Missing links have the value 'zero' in the traffic matrix.

2.8 Traffic Routing

Traffic routing in the network involves the selection of a path for traffic to travel between source and destination nodes [17]. There are a number of ways to classify and select routes between source and destination nodes. One way is to differentiate routing algorithms into centralized and distributed. In a centralized algorithm traffic routing decision is made by the central controller while in distributed algorithms routing tables/traffic matrix is distributed among all the nodes to make a decision. The second way of classifying traffic routing algorithms are using static and dynamic routing. In the case of static routing, the path between source and destination pair is fixed and it can only change in case of node or link failure whereas in dynamic routing the path can be adaptive according to traffic conditions such as congestion, link failure, etc. Network traffic routing also affects the performance measure of the whole network in terms of throughput and average packet delay. Therefore, if routing of traffic helps in keeping delay low in a network, in turn, it allows more traffic to travel between source and destination pair. The routing algorithms can be categorized as single shortest path algorithms and multi-path routing algorithms. In general, single shortest path algorithms are more troublesome as they route the packets using the same path every time that can create congestion and link failures. Multi-path routing is preferred as it enforces load balancing and tries to prevent congestion.

2.9 Network Optimization

Network optimization is the practice to improve the performance of the network. There are various designs and methodologies involve getting an optimal network such as load balancing of the network, minimizing packet loss, jitter, congestion, latency, managing bandwidth utilization. It is a technique that involves the setting of a certain set of rules to get an optimal design in a cost-effective manner which in turn is beneficial for network operators [18]. In our thesis, we showed an improvement in load balancing by implementing Yen's multipath routing algorithm in our experimental setup.

Chapter 3

3 Problem Statement and Related Work

This chapter formally defines the problem statement and discuss the existing solutions proposed to address the problem. We also identified the gaps in the related studies and set the rationale for our selection of the proposed methodology.

3.1 Problem Statement

Let $G\ (V, E)$ denotes a network, where V is the set of n number of nodes and E is a set of links in the network. $E_{i,j}$ denotes a directional link between any two nodes i and j, where $1 \leq i, j \leq n$. Suppose a traffic matrix $T_{Current}$ gives a current traffic pattern and another traffic matrix $T_{Forecast}$ gives a forecasted traffic pattern. The objective is to design a network planning framework that simulates a network G with a traffic pattern $T_{Current}$ and measure the QoS parameters, pass the forecasted traffic and re-measure the QoS parameters, and identify the high-risk links (links with high utilization).

Based on the link utilization, we benchmarked the link status into three categories: “OK”, “WARNING”, and “FAILED”. The objective is to identify the links that get the major change in link utilization and are prone to failure. The benchmark rules to identify the high-risk (highly utilized) links are as follows:

- Status “OK”: link utilization up to 50%
- Status “WARNING”: link utilization between 50% and 80%
- Status “FAILED”: link utilization more than 80%.

We also analyzed other QoS parameters (throughput, latency, jitter, and packet loss) for both the current and the forecasted traffic to analyze the impact of increased traffic on the QoS.

For any network G, we propose the Load Balancing Percentage (LBP) value that can be used to quantitatively measure (in percentage) the network load balance. More the LBP value means more equally distributed traffic. The LBP values for current traffic (single shortest path routing using Dijkstra’s algorithm) and forecasted traffic (multi-path routing using Yen’s algorithm) are to be calculated. LBP value is to be used to test the effect of the multi-path routing on the load balancing. The LBP value for n number of links is calculated as:

$$LBP = \; 100 - \left(\frac{\sum_{i=1}^{n} |AverageUtilization - LinkUtilization_i|}{n}\right) \tag{3.1}$$

The following assumptions are made:

- Packets are routed using multipath routing (Yen’s algorithm with 3-shortest paths between every source-destination pair).
- Load balancing is assumed to be enforced using multi-path routing.
- All network links are bidirectional.

3.2 Related Work

3.2.1 QoS Parameters

The various QoS parameters such as packet loss, jitter, delay, etc. and factors that contribute to the network performance are discussed in [5]. The authors in [5] proposed recommendations such as checking the network installation, changing network topology, increasing the bandwidth capacity, etc. to improve the network performance which leads to improving the QoS. Santhi et al. [10] compared Ad- Hoc On-Demand Distance Vector (AODV), and Ad-Hoc On-Demand Multipath Distance Vector (AOMDV) multipath routing protocols for a heterogeneous network using end to end delay, bandwidth, routing overhead and packet delivery ratio by introducing different traffic load in the network. The multipath routing protocols resulted in a decrease in delay, packet loss, and congestion, and helped in better bandwidth utilization. Jasem et al. [11] studied the Additive Increase Multiplicative Decrease (AIMD) mechanism for congestion avoidance and control and proposed a new modified AIMD mechanism in order to reduce the average queue length which in turn decrease the end to end delay, helps in increasing bandwidth utilization and results in avoidance of network congestion. Dahmouni et al. [12] discussed the jitter requirements for new multimedia applications. The author analyzed the impact of jitter constraints on the network and proposed the jitter calculation model in order to solve the optimal routing problem. Authors in [12] considered two types of flow for optimal routing using jitter and delay constraints and concluded that jitter constraint flows are different from delay constraints. Mtawa et al. [13] provided a comprehensive review of various QoS parameters. We considered this paper as the basis of our analysis for calculating QoS parameters. Atole et al. [19] analyzed the performance of QoS in WSN for multiple topologies with a difference in the number of nodes. Packet delivery ratio, throughput, delay, routing overheads, average energy consumed, average residual energy, etc. were the performance metrics to analyze QoS of wireless sensor networks. In the proposed method AODV routing protocol was used in the NS-2 simulation environment. It is concluded that as the number of nodes increases throughput increases initially but when the density of the network reached 50, it decreases because of congestion. QoS can be improved by using load balancing techniques. Shivapur et al. [20] compared cluster-based, protocol-based and algorithm-based techniques for load balancing and listed out their respective limitations. In [21], the authors proposed a model implemented in the NS2 simulator to predict and minimize the congestion for wireless sensor networks by predicting nodes that depend on the placement of sensor nodes. The model can predict both periodic and event data generation, but the results are based on estimated data and could be inaccurate as stated in the study. The inability to simulate for different types of sensors with varying transmission rates is another limitation listed in this study.

3.2.2 Multi-Path Routing

In [23], J Yan et al. proposed a scheme called HiQoS that uses multiple paths between origin and destination and queuing mechanisms to guarantee QoS for different types of traffic. This method also provides QoS guarantee reducing delay and increasing throughput. As a result, it is easier to recover from link failure by rerouting traffic from a failed path to another available path. In [24], Hui Dai et al. proposed a node-centric load balancing algorithm for wireless sensor networks. The constructed load-balancing routing trees were found to be experimentally better than from breadth-first search and shortest path routing using Dijkstra. Single shortest path-based routing techniques are not reliable and may be troublesome. When every time packets are transmitting using the shortest path between the origin and the destination node, the same route is not usable after some time. The frequent usage of the same path creates congestion due to an extensive increase in the number of packets. Continuous usage of the same path creates a load imbalance in the whole network [25, 26]. So, multi-path routing techniques are better suited for load balancing. Sandor et al. [27] did a performance analysis of wireless sensor networks by taking jitter and packet loss as performance indicators in peer-to-peer communication. The experiments were done using different cases: with or without Acknowledgment, different sizes of packet and period, with and without parallel competitive traffic. The simulation results showed that the parallel data transmission technique increases the throughput of the sensor network. In [28], the authors investigated the response of IoT infrastructure when a huge amount of data is generated by IoT devices by analyzing throughput, delay, and load. In [26], the authors proposed a congestion predictor model by considering parameters like network energy consumption, packet loss rate and percentage of packets delivered to the destination. The model aims in preventing congestion, reducing the number of packet loss and increases the priority of delivered packets. The successful models showed the importance of multi-path routing in achieving the load balancing and better QoS. [29] proposed a modified version of Dijkstra's algorithm by using random weights instead of equal-weighted links. This approach works better than default single shortest path routing but is not fault-tolerant when dealing with bigger mesh topologies with higher link utilization. Eghbali et al. [30] showed the use of a multipath routing algorithm using a multipath directed diffusion protocol. In the proposed method multiple paths were calculated between the source and the sink; as a result, a lifetime of network connection increased which in turn lead to load balancing of the network. In [31], the authors proposed the multipath-generalized topology model. When a certain node fails or links become over-utilized, the traffic can spread on multiple/alternate paths which helps in balancing the load, enhances the degree of fault tolerance and network connection. This method has two phases: first is the discovery of route and second is the maintenance of the route. The performance of the proposed model has been analyzed using packet Delivery Ratio, Average end-to-end delay, Throughput as QoS parameters in comparison with standard Dynamic Source Routing. [32] also improved results using weighted cost multipathing. [30-32] showed that the use of multi-path routing algorithms for Wireless sensor networks achieved efficient load balancing by minimizing the probability of communication disruption. ISP’s mostly use the Border Gateway Protocol (BGP) for routing which selects only the best available path over a session [33]. Despite the challenges [34], the trend is shifting towards the internet-wide multipath routing [35]. We used Yen’s algorithm [36,37] in our experiments. Yen’s algorithm is an approach to get the k-shortest paths from a source to a destination. Secondary paths can be efficiently obtained by calculating deviations from the previously calculated paths. This can be achieved by dividing the algorithm into two parts: 1) determining the first shortest path in the network and 2) temporarily redefining the distance between some of the edges contained in the last shortest paths to $+\infty$, then re-applying a shortest-path algorithm to the modified network graph. We used k=3 to calculate the three shortest paths for every source-destination pairs. Packets are rerouted to the next shortest path when the shortest path becomes highly utilized. We also compared [38] Yen’s algorithm with Suurballe’s k-disjoint algorithm. Our results showed that Yen’s algorithm outperforms other algorithms in comparison most of the time.

3.2.3 Comparison of Floodlight Controller with other Controllers

After selecting Mininet simulator for our experiments and Yen’s algorithm for multi-path routing based load balancing, we reviewed various controllers. In [39,40], different QoS parameters are considered to evaluate and compare the performance of the Open-Daylight controller with respect to the Floodlight controller. The comparison in [37] is done using latency and throughput as QoS parameters. [40] used delay and packet loss as QoS parameters on different size topologies and varying network loads. It was concluded that Floodlight outperforms the Open-Daylight controller in case of heavy traffic load for tree topology. In [41] the performance of POX and Floodlight controllers was considering different topologies. Emulation results showed that the floodlight controller outperforms the POX controller in terms of packet transmission rate as well as throughput. However, due to inbuilt java files, the floodlight controller occupies more memory space than the POX controller using python. Laissaouui et al. [42] compared four SDN controllers Floodlight, Beacon, Pox, Ryu using latency and throughput. The experiments were done on different network sizes considering latency and throughput as QoS parameters and it was concluded that Floodlight performed better among these four controllers. A thorough analysis covering most of the QoS parameters (Latency, Bandwidth utilization, Packet transmission rate, jitter, and packet loss) for TCP and UDP traffic was done to evaluate the performance of four SDN controllers in [43]. The experimental results in [43] showed that the Floodlight controller worked fine in all considered scenarios while the performance of the Pox controller and Open-Daylight controller degraded when the number of hosts and number of links increased. In [44] authors compared the Open-Daylight and Floodlight controllers on the basis of the proposed load balancing algorithm. The open-daylight controller didn't perform well in handling the bandwidth allocation and response times. On the other hand, Floodlight performed better for both. It was concluded that the Floodlight controller with default settings is stable enough and doesn't require any changes. The Floodlight controller outperforms the other controllers in the aforementioned experimental studies [39-44].

The QoS parameters get impacted by a massive amount of internet traffic. Most of the existing studies analyze the QoS parameters impact based on the current network traffic scenarios. These analyses may be of little to no use as the future internet traffic may bring new unexpected challenges. From ISP's perspective, understanding the possible impact of the future internet traffic on its network is critical for provisioning their network capacity in a cost-effective manner while meeting network QoS requirements. A network planning framework is required that can take input from the traffic forecast and analyze its impact on network-wide latency by evaluating the most of QoS parameters, and finally suggests an optimal routing strategy that can improve the overall network health.

The majority of existing analyses [12,18,19] are based on two or three QoS parameters and do not cover most of the parameters. So, existing results may be misleading because of incompleteness in terms of covering the majority of QoS parameters. Though many QoS parameters analyses have been performed in the past, to the best of our knowledge no study predicts the impact of forecasted traffic based on the current network scenarios and only a few analyze the majority (five or more) of QoS parameters. Our technique is novel in initializing the network as per current traffic and predicting the network status using QoS parameters by simulating forecasted traffic.

Chapter 4

4 Methodology

The increasing demand for various Internet Service has led to an exponential growth of Internet traffic in the last decade, and that growth is likely to continue. This massive growth of traffic impacts the Quality of Service which in turn effects the ISP's and network planners. It is very important to understand the possible impact of the future growth of internet traffic. Existing approaches lack such kind of framework that analyzes the possible impact of forecasted traffic on Network QoS. In our approach, we introduce NIMA (Network Impact Modelling and Analysis) as a novel framework that analyzes the impact of forecasted traffic on QoS parameters by initializing the simulating setup based on the current network traffic. Our proposed methodology can be used as a network planning tool that is able to predict the congestion level in the network and alert the network planners for any high-risk links.

4.1 NIMA Methodology

The flowchart shown in Figure 4.1 outlines the NIMA methodology. We create network topology and initialize the network simulation using a floodlight controller. The NIMA model takes network traffic flow as input twice (pass 1 and pass 2). In pass 1, current traffic is simulated and QoS parameters are evaluated. NIMA uses a REST application interface of the floodlight controller to calculate various QoS parameters. It stores these parameter values for later comparisons. In pass 2, forecasted traffic is passed to the same network setup and the QoS parameters are again evaluated. A comparison of the QoS parameters for both current and forecasted traffic is done to analyze the traffic impact on the network. If there are any high-risk QoS metrics (links prone to the failure) detected, a network planner is alerted to take appropriate action. A network planner can then make an illustrated plan for policy changes such as bandwidth increase in certain links, or to introduce new links in the network, etc. Stored results are also used for graph generation that can be helpful in visualizing the impact of forecasted traffic on the QoS.

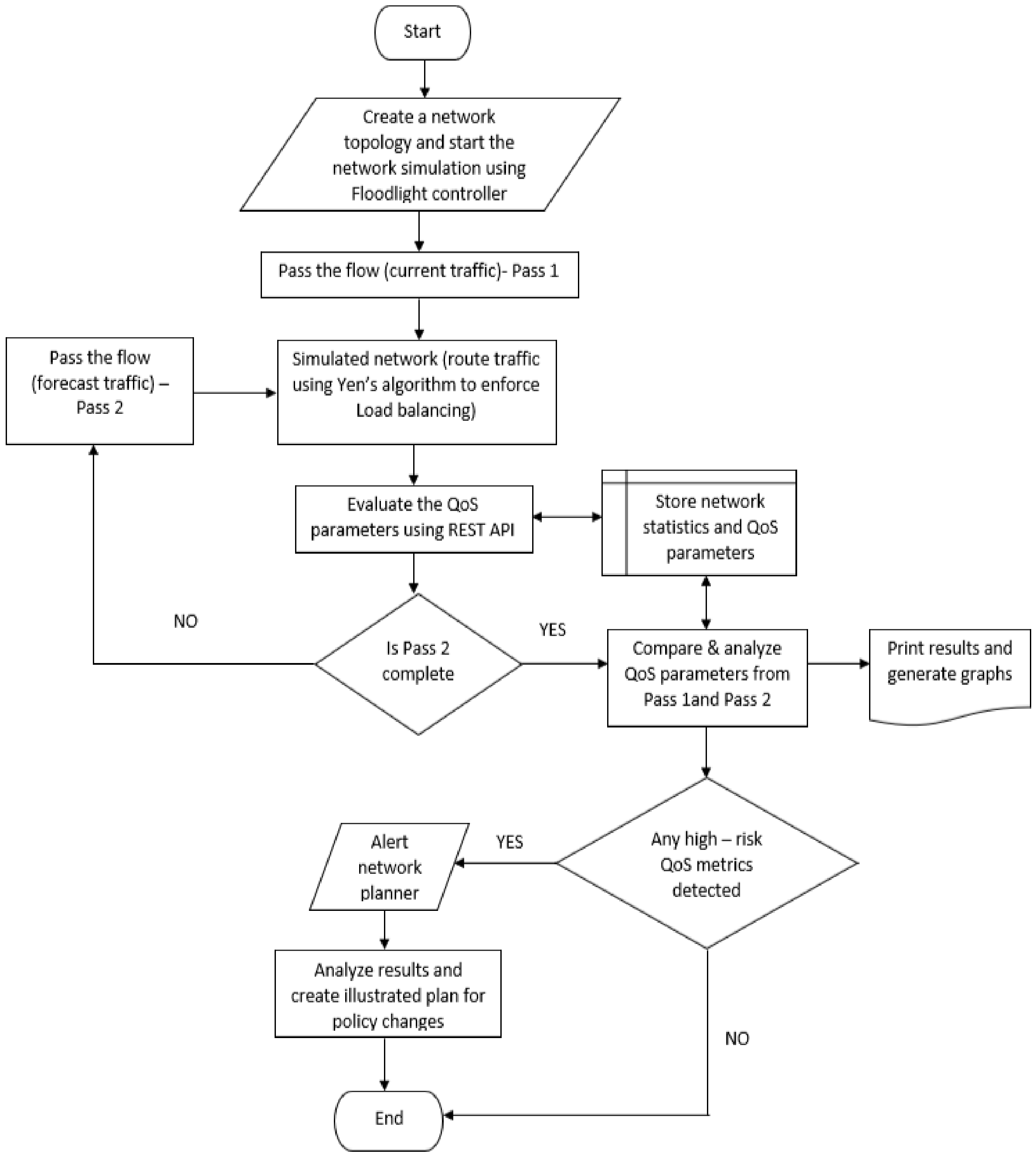

Start
Create a network topology and start the network simulation using Floodlight controller
Pass the flow (current traffic)- Pass 1
Pass the flow (forecast traffic) – Pass 2
Simulated network (route traffic using Yen's algorithm to enforce Load balancing)
Evaluate the QoS parameters using REST API
Store network statistics and QoS parameters
NO
Is Pass 2 complete
YES
Compare & analyze QoS parameters from Pass 1and Pass 2
Print results and generate graphs
Alert network planner
YES
Any high – risk QoS metrics detected
Analyze results and create illustrated plan for policy changes
NO
End

Figure 4.1: Flowchart outlining NIMA methodology

NIMA methodology is tested on five different topologies (T1-T5) shown and discussed in the next section. T1 and T2 are partial-mesh topologies, whereas T3, T4, and T5 are full-mesh topologies. The different QoS parameters used are Throughput, link-utilization, latency, jitter, packet-loss, and load-balancing. We compared different QoS parameters for both current and the forecasted traffic on all five selected topologies.

4.2 Network Topology Design

In a real-world scenario, the network is a combination of partial and full-mesh topologies. For our experiments, we considered both partial and full-mesh topologies by testing the scalability of the switches and the hosts within the limitation of the simulation environment. We used five different sized topologies (shown in Figures 4.2 – 4.6), having a difference in the number of nodes and interconnected as a full-mesh and partial-mesh network. Open kernel switches (also known as open-v-switches or OVS-switches) are used in our topology designs. Asadollahi *et al.* [45] tested the scalability of the floodlight controller using mesh topology and suggested to have 50 hosts in topology as the benchmark size for a stable network. We experimented with the number of hosts ranging from 5 to 100 by keeping the number of switches fixed to 5 with an exception of 10 switches for one topology. In accordance with the literature [45], we confirmed that the increasing number of switches makes the Floodlight controller unstable, so we used 5 switches for all of our topology with just a single exception. The topologies considered for our experiments are as follows:

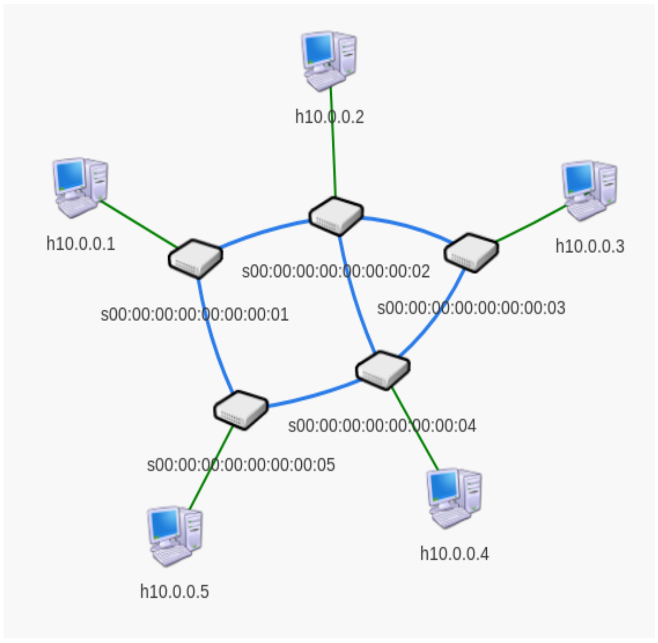


Figure 4.2: Topology T1 is organized as partial- mesh topology consisting of five switches each connected to a single host (five hosts on total)

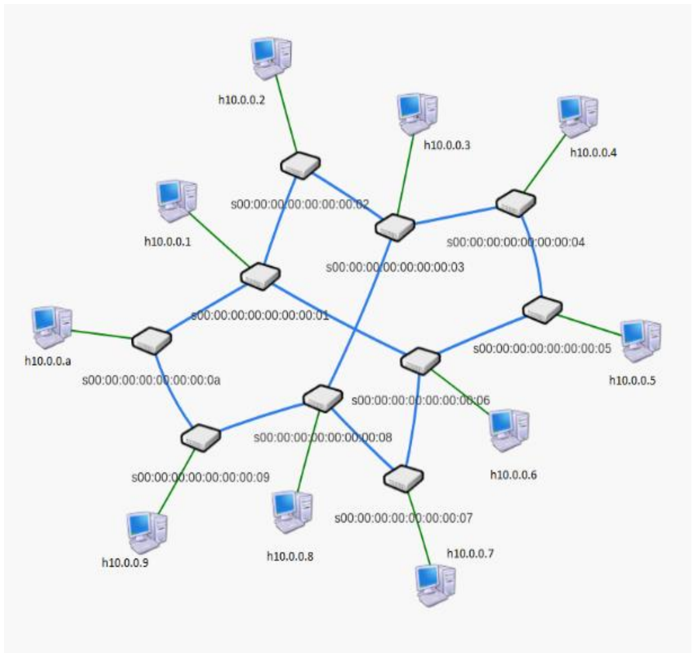


Figure 4.3: Topology T2 is a partial-mesh topology that consists of ten switches each connected to a single host (10 hosts in total)

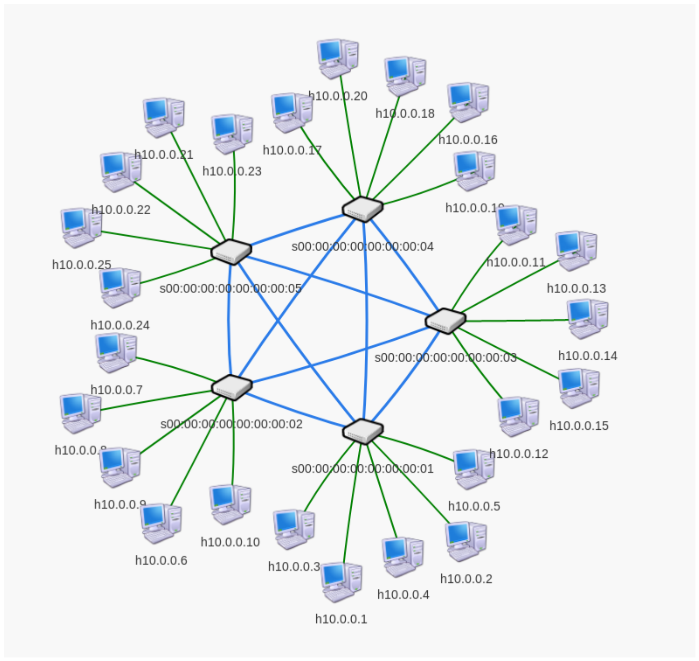


Figure 4.4: Topology T3 is a full-mesh topology that consists of five switches each connected to five hosts (25 hosts in total)

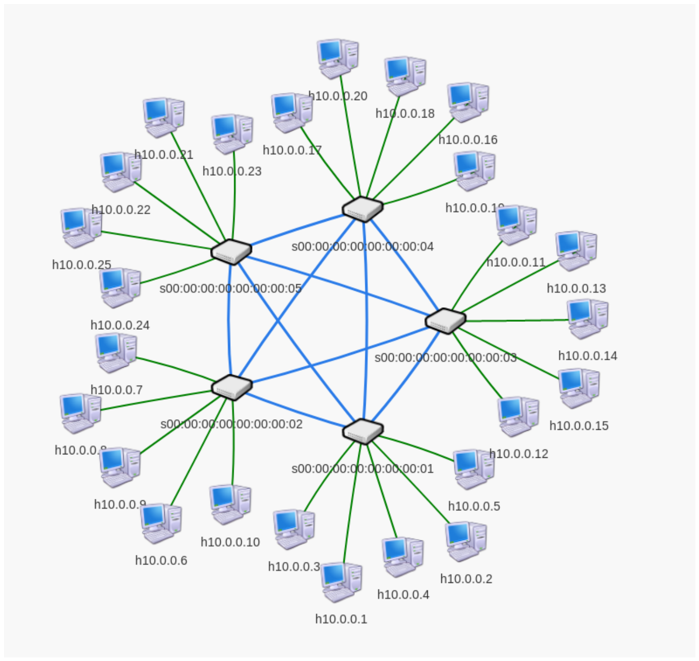


Figure 4.5: Topology T4 is a full- mesh topology with five switches each connected to ten hosts (50 hosts in total)

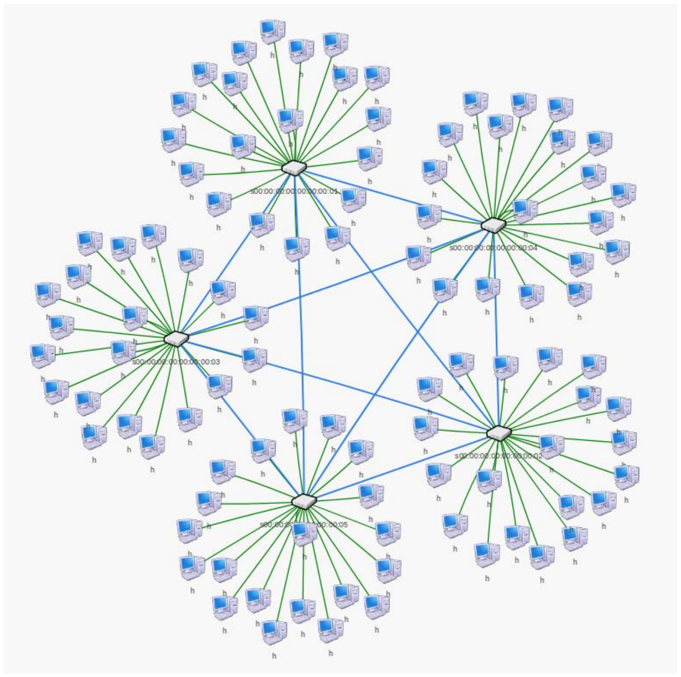


Figure 4.6: Topology T5 is a full-mesh topology with five switches each connected to twenty hosts (100 hosts in total)

4.3 Traffic and Routing

4.3.1 Traffic Generation

In our analyses, we use two different types of traffic flows - one flow as probe packets that employs Internet Control Message Protocol (ICMP) generated using the ping tool, second is the traffic generated using Transmission Control Protocol (TCP). The objective of ICMP is to measure the Round-trip time. To calculate the throughput, and the packet loss TCP is employed to generate traffic with window size 85.3 Kbytes. TCP traffic is generated using the Iperf tool. Iperf works on client-server functionality and helps in measuring the end-to-end throughput. For our simulation, the forecasted traffic is calculated as:

$$\sum_{i=1}^{n} \delta_i . t \tag{4.1}$$

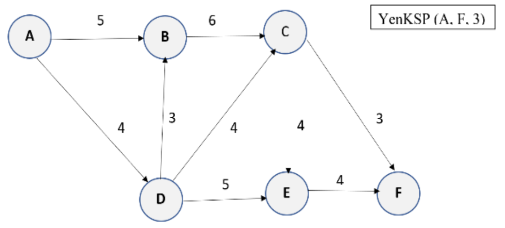
where, n is the number of nodes, δ_i is the data rate of the device (Mbps), and t is the time window (in seconds) to run the simulation. Forecasted traffic is generated by the i^{th} node can be calculated by multiplying its data rate to the time interval.

4.3.2 Traffic Routing

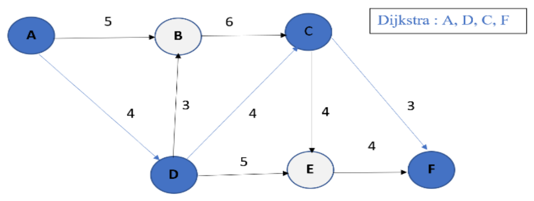
For our experiments, we use multi-path routing. Yen’s algorithm is used to compute the k-shortest paths between every source-destination pairs. We use k=3 in our experiments to find up to 3 shortest paths between two endpoints. Yen’s algorithm starts with finding the shortest path on a graph G = (V, E), from source node S to destination node T using Dijkstra’s shortest path algorithm and store the results in the results list (Yen's list X). The algorithm takes every node in the shortest path except the terminating node and calculates another shortest path (spur) from each selected node to the terminating node. For each node, the path from the start node to the current node is the root path. Two restrictions are placed on the spur path: 1) It must not pass through any node on the root path (i.e. loopless) and 2) It must not branch from the current node on any edge used by a previously found k-shortest path with the same root. Node and edge markings are used to prevent the spur paths from looping or simply following the route of a previous k-shortest path. If a new spur path is found it is appended to the root path for that node, to form a complete path from start to end node, which is then a candidate for the next k-shortest path. All such paths are stored (Yen's list Y) and the shortest remaining unselected path is selected as the next KSP and transferred to the results list (Yen's list X). The same process is repeated, calculating a spur path from each node in each new k-shortest path until the required number of k-shortest paths have been found [46].

```
function YensKSP (Graph G, Source S, Sink T, K)
|       //calculate the shortest path; Initialize Y to store the potential kth shortest path
|       X [0] = Dijkstra (G, S, T);
|       Y = [];
|       // calculate k-shortest paths
|       for k = 1: K
|       |       //calculate for all spur nodes in the previous path
|       |       for i = 0: size(X[k-1])-2
|       |       |       //retrieve the spur node from the previous kth-shortest path
|       |       |       spurNode = X[k-1]. node(i);
|       |       |       //save path from S to the spur node of the previous kth-shortest
|       |       |       //path as root path
|       |       |       rootPath = X[k-1]. nodes (0, i);
|       |       |       for each path p in X:
|       |       |       |       // Remove the edges which share the same root path
|       |       |       |       // and were included in the previous shortest paths
|       |       |       |       if rootPath == p.nodes(0, i):
|       |       |       |       |       remove p.edge(i,i + 1) from G;
|       |       |       |       end
|       |       |       end
|       |       |       for each node rootPathNode in rootPath except spurNode:
|       |       |       remove rootPathNode from G;
|       |       |       end
|       |       |       //calculate the spur path from the spur node to T
|       |       |       spurPath = Dijkstra (G, spurNode, T);
|       |       |       //save complete path
|       |       |       totalPath = rootPath + spurPath;
|       |       |       //add the potential kth-shortest path
|       |       |       Y.append(totalPath);
|       |       |       //re-add the removed edges and the nodes to G
|       |       |       restore edges to G;
|       |       |       restore nodes in rootPath to G;
|       |       end
|       |       //stop if there are no spur paths left or no more spur paths exist
|       |       if Y is empty
|       |       |       break;
|       |       end
|       |       //sort the potential k-shortest paths by cost
|       |       Y.sort();
|       |       //add the path with lowest cost as the kth shortest path
|       |       X[k] = Y [0];
|       |       Y.pop ();
|       end
|       //return k-shortest paths between S and T as result
|       return X;
end
```

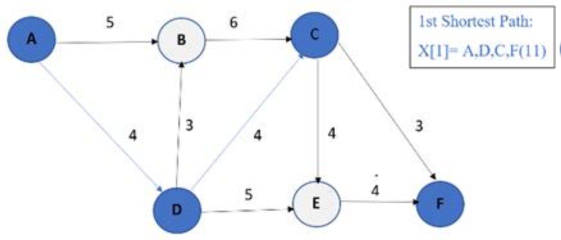
Figure 4.7 shows the steps involved in multi-path routing calculations using Yen's algorithm. We took graph G = (V, E) where V ={A,B,C,D,E,F} and E = {(A,B), (A,D) (B,C), (B,D), (C,F) (C,D) (D,E), (E,F) }.



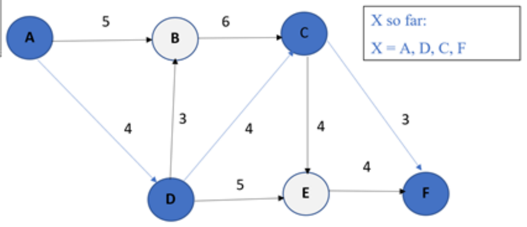
(1)



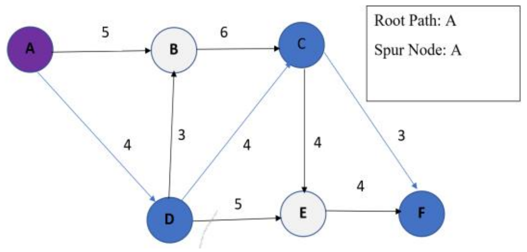
(2)



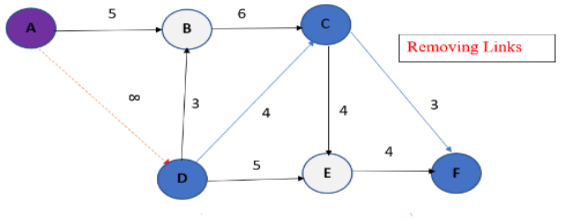
(3)



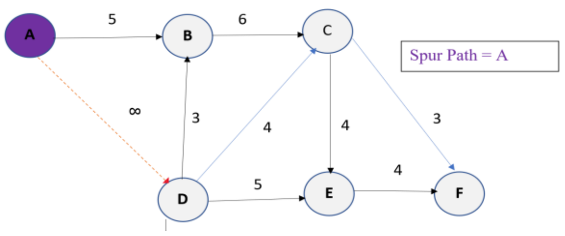
(4)



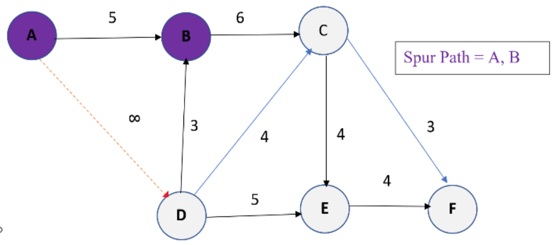
(5)



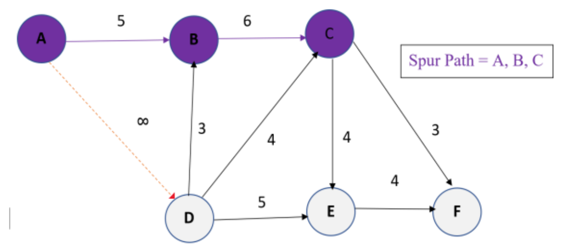
(6)



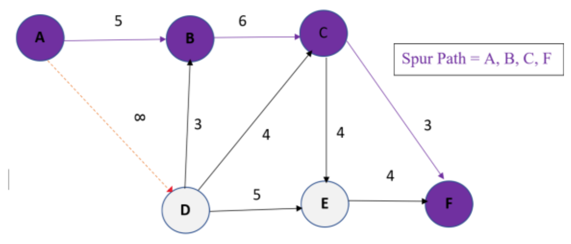
(7)



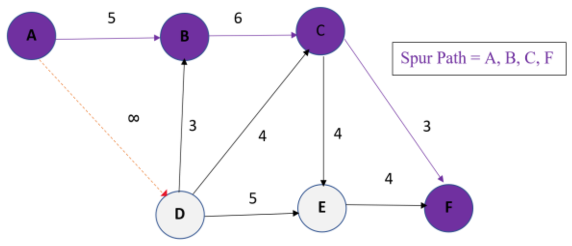
(8)



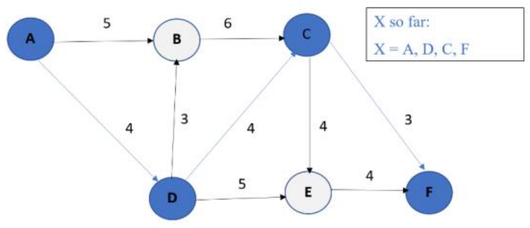
(9)



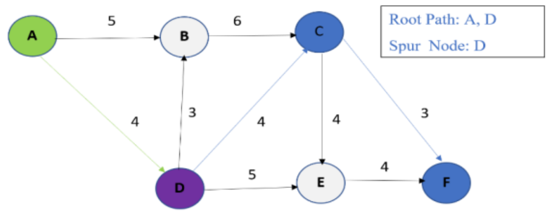
(10)



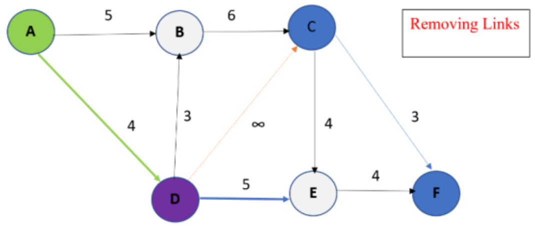
(11)



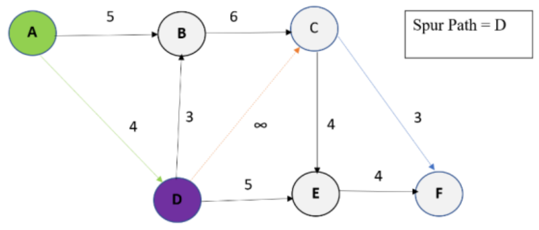
(12)



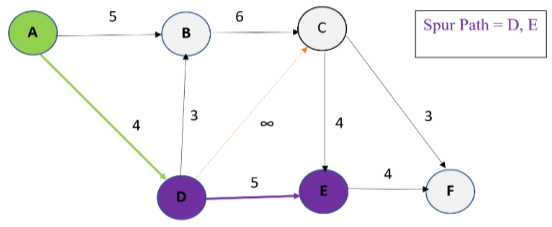
(13)



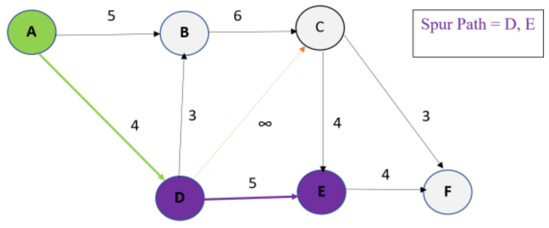
(14)



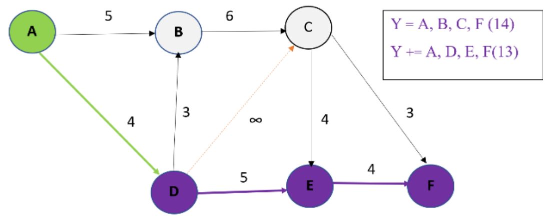
(15)



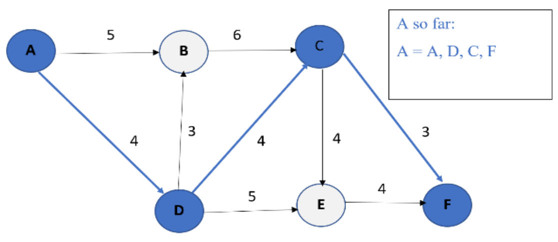
(16)



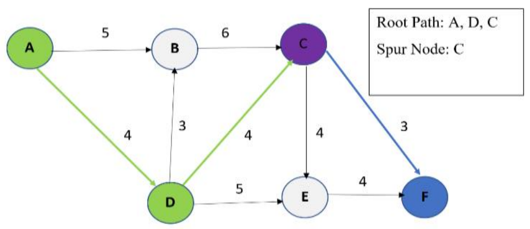
(17)



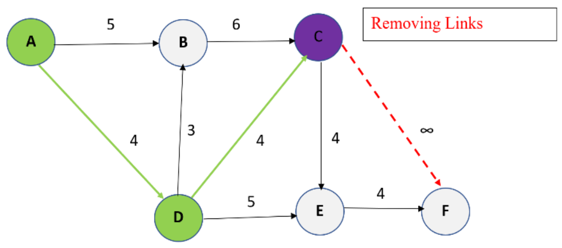
(18)



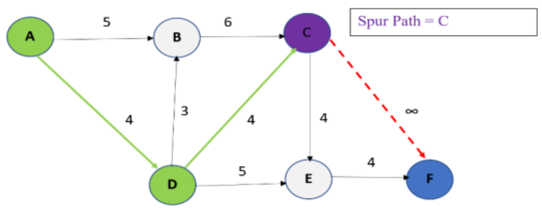
(19)



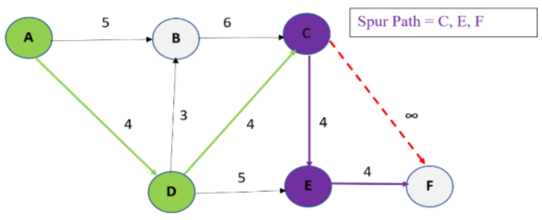
(20)



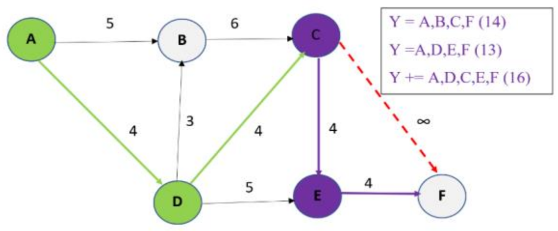
(21)



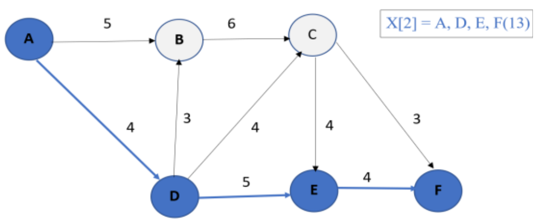
(22)



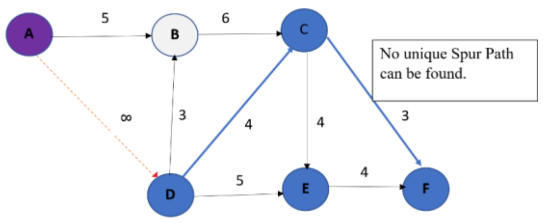
(23)



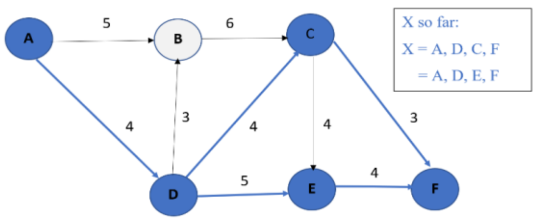
(24)



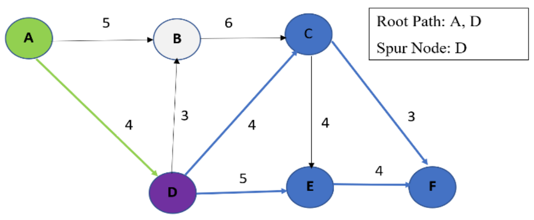
(25)



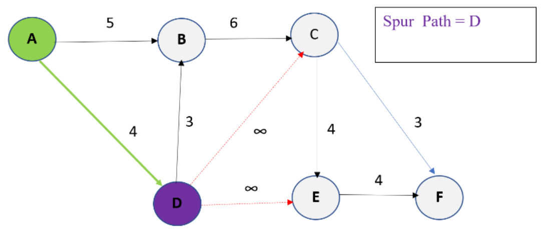
(26)



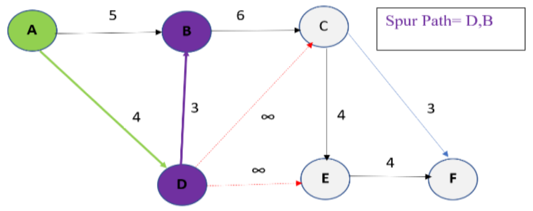
(27)



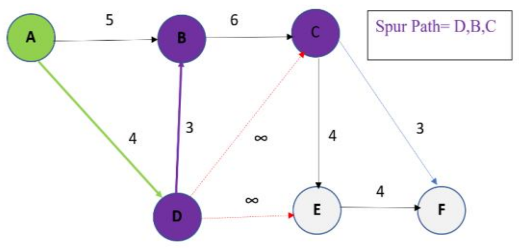
(28)



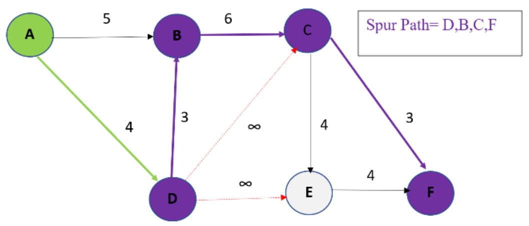
(29)



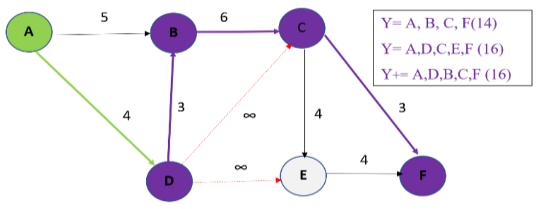
(30)



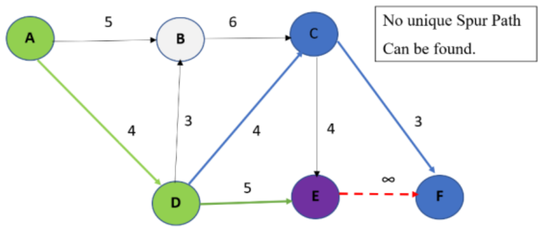
(31)



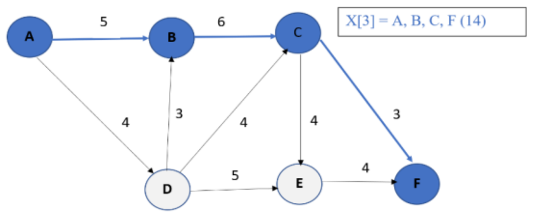
(32)



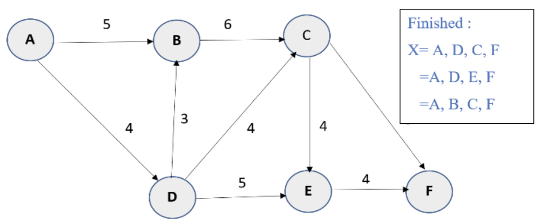
(33)



(34)



(35)



(36)

Figure 4.7: Example showing multi-path routing calculation using Yen's algorithm

Steps showing the working of Yen’s algorithm:

1. YenKSP (A, F, 3)

2. Dijkstra (A, D, C, F)

3. First Shortest Path: X [1] = A, D, C, F(11)

4. X so Far: X = A, D, C, F

5. Root Path: A, Spur Node: A

6. Removing Links

7. Spur Path A

8. Spur Path A, B

9. Spur Path A, B, C

10. Spur Path A, B, C, F

11. X so far: X= A, D, C, F

12. Root Path: A, D; Spur Node: D

13. Removing Links

14. Spur Path= D; Spur Path= D, E

15. Y= A, B, C, F (14); Y+= A, D, E, F(13)

16. A so far: A= A, D, C, F

17. Root Path: A, D, C, F; Spur Node: C

18. Removing Links

19. Spur Path = C

20. Spur Path= C, E, F

21. Y= A, B, C, F (14); Y= A, D, E, F (13); Y+= A, D, C, E, F(16)

22. X [2] = A, D, E, F (13)

23. No unique Spur Path can be found.

24. X so far: X= A, D, C, F

    X = A, D, E, F

25. Root Path= A, D

    Spur Node= D

26. Spur Path= D

27. Spur Path= D, B

28. Spur Path= D, B, C

29. Spur Path= D, B, C, F

30. Y= A, B, C, F (14)

    Y= A, D, C, E, F (16)

    Y+= A, D, B, C, F (16)

31. No Unique Spur Path can be found.

4.4 Statistics Collection

NIMA uses a REST API (REpresentational State Transfer Application Programming Interface) of the Floodlight controller to monitor the network statistics. A REST API is a mechanism by which a program can exchange data with an external entity over HTTP at runtime. The application listens to HTTP Requests along and opens a network server socket. API can be accessed by external users as well as other programs to send and receive messages to and from it. The application will perform some prescribed (recommended) tasks, based on the HTTP request. Floodlight implements a REST API within application modules (e.g. the Device Manager, the Firewall, the Topology Manager, etc.), as well as in the controller core. Application modules implement REST API for a variety of tasks such as statistics collection, adding a firewall rule, route finding, adding/removing a flow, etc. Each module implements the API for a different purpose. For example, the Topology manager uses it for traffic routing, and the Static Entry Pusher module allows a user to send flows to the Floodlight controller, as well as to query or delete the flows using an HTTP request to the published URLs. Other Floodlight modules that implement REST APIs include, but are not limited to, the Device Manager, the Firewall, and the Topology Manager. The REST API is the recommended interface to make use of features exposed by the Floodlight controller [45].

Link Discovery Module is used for link discovery and maintain the list of active links in the OpenFlow network. The link Discovery Module uses both LLDP (Link Layer Discovery Protocol) and BDDP (Broadcast Domain Discovery Protocol) to discover the available network links [48]. For direct links, LLDP packets are used to discover links between switches. BDDP is used to discover the indirect connections between switches in the broadcast domain. After link discovery, the Routing Manager configures link cost metrics based on all available paths between the given source and destination pairs and then statistics are collected. The statistics in the floodlight controller are configured by first using “Startup Configuration” in the statistics module. The startup configuration is used to enable and disable the statistics collection module. The configuration variable “net.floodlightcontroller.statistics.enable = <boolean>” is used to enable or disable the statistics collection module. The value of Boolean is true or false, true to enable the module and false to disable the module. Another configuration variable “net.floodlightcontroller.statistics.collectionIntervalPortStatsSeconds = <positive-integer>” is used to set the interval at which port stats request messages are sent and bandwidth is recomputed. The value of Positive integer is 1 or 2, '1' to issue port stats request every second, '2' to issue port stats request every 2 seconds. Statistics Service module in the Floodlight controller is used to collect the network statistics such as Link Bandwidth Utilization, Latency, Throughput, Traffic, Packet loss, jitter. Topology Manager module finds the routing information and maintains the topology information for the controller. The topology manager finds the shortest path among all source-destination pairs using Dijkstra’s algorithm, the same as when Yen’s algorithm is used for single-path. We used Yen’s algorithm for multi-path routing with k = 3 (k decides the number of multi-paths, 3-shortest paths routing in our case). To assign the link costs for finding the shortest paths, the Topology manager makes use of statistics collection. Users can select any single parameter such as utilization, hop count, etc. that should be used to calculate the shortest paths. We changed the default Topology manager module to collect and export the all available QoS parameters that are later analyzed by NIMA for the stats comparison before and after passing the forecasted traffic.

4.4.1 Traffic Matrix

Simulated network topologies are initialized using the current traffic and change in the QoS is analyzed after passing the forecasted traffic in the same network. Table 4.1 shows the traffic matrix for current traffic and Table 4.2 shows the traffic matrix for forecasted traffic used in the topology T4. It can be seen that a total of 5509.83 Mb was passed as current traffic and 7788.56 Mb as forecasted traffic.

Traffic matrix (current traffic)	**S1**	**S2**	**S3**	**S4**	**S5**	**T_i**
S1	0	175.50	324.65	470.34	119.51	1090.00
S2	175.50	0	156.74	543.51	104.16	979.91
S3	324.65	156.74	0	307.30	354.24	1129.06
S4	470.34	543.51	307.30	0	205.90	1527.05
S5	119.51	104.16	354.24	205.90	0	783.81
T_j	1090.00	979.91	1129.06	1527.05	783.81	**5509.83**

Table 4.1: Traffic matrix showing current traffic used in topology T4

Traffic matrix (forecasted traffic)	**S1**	**S2**	**S3**	**S4**	**S5**	**T_i**
S1	0	273.00	362.40	568.62	270.47	1474.49
S2	273.00	0	295.04	677.16	279.93	1525.13
S3	362.40	295.04	0	491.68	413.28	1562.40
S4	568.62	677.16	491.68	0	262.70	2000.16
S5	270.47	279.93	413.28	262.70	0	1226.38
T_j	1474.49	1525.13	1562.40	2000.16	1226.38	**7788.56**

Table 4.2: Traffic matrix showing forecasted traffic used in topology T4

Figure 4.8 shows the current traffic distribution in the topology T4. For every network link, the label follows the format Utilization-Traffic in Mb / Bandwidth in Mbps-Latency. For example, link (S1, S2) has a label 27-176/650-15 which means this link has a bandwidth of 650 Mbps, carries 176 Mb of network traffic, has link utilization of 27% and latency 15ms. The link colors show the utilization status (green for utilization less than 50 % with "OK" status, yellow for utilization between 50-80 % with "WARNING" status, and red for utilization > 80 % with "FAILED" status). For each network link, link utilization is calculated as the percentage of the link bandwidth used by the traffic going through that link.

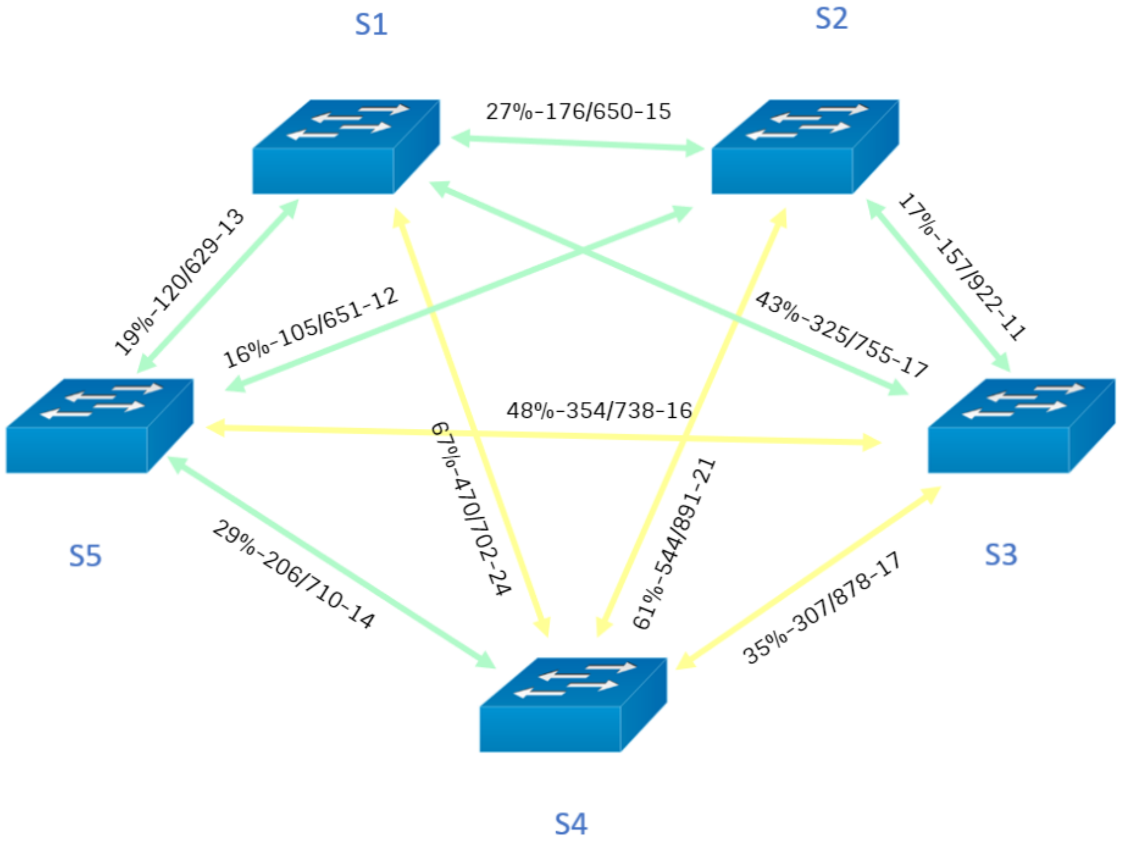


Figure 4.8: Current traffic distribution in topology T4

Figure 4.9 shows the forecasted traffic distribution in the topology T4 with the same format as used in Figure 4.8. It can be seen that the link utilization for the link (S1, S4) increased from 67 % to 81 % and hence its utilization status changed from "WARNING" to "FAILED" (visualization changed from yellow to the red color).

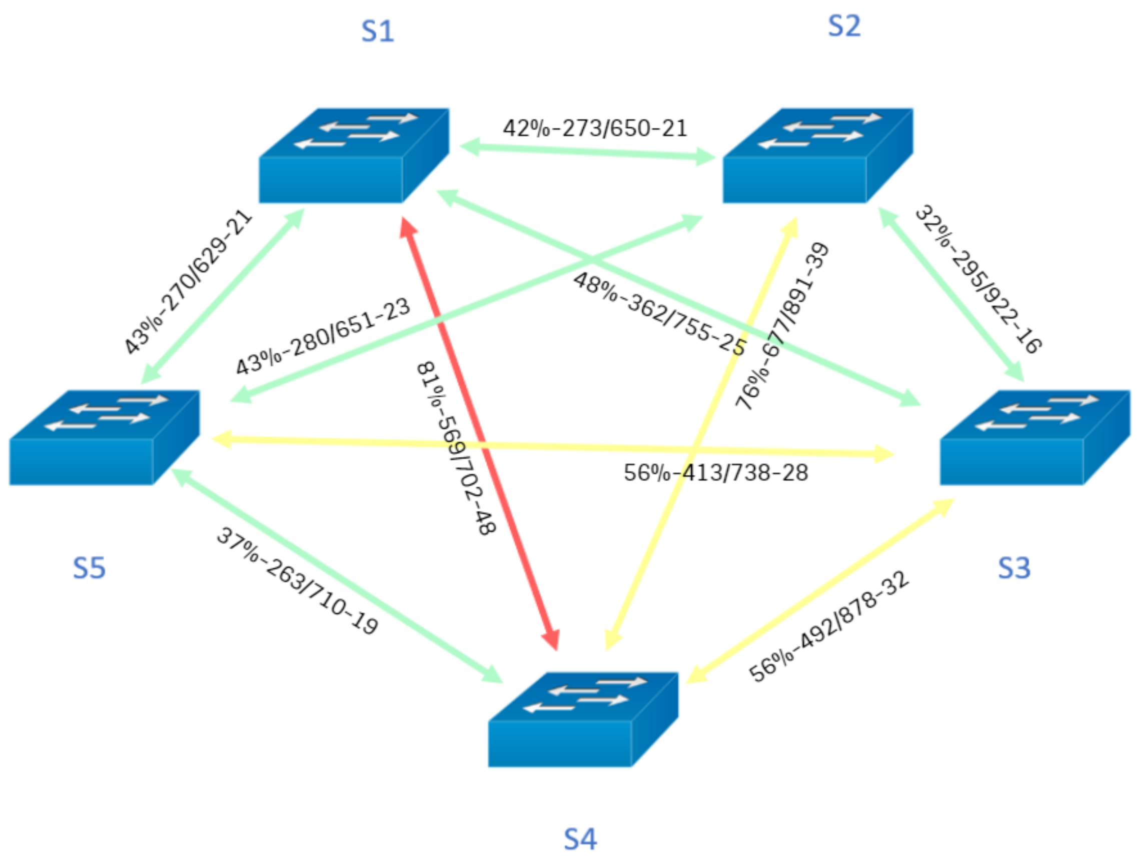


Figure 4.9: Forecasted traffic distribution in topology T4

4.4.2 QoS Stats Collection

We selected two farthest hosts for each topology to calculate the rest of the QoS parameters (throughput, jitter, and packet loss). The TCP protocol is used with window size 85.3 Kbytes and the QoS parameters are observed over the 15 seconds window for both current and forecasted traffic.

For current traffic, the shortest paths are calculated using the available bandwidth, and the hop count. Any path P of length l can be denoted as $P= \{p_1, p_2, \ldots, p_l\}$, where, any pair (p_i,p_{i+1}) is the link between two nodes p_i, p_{i+1} and $1 \leq i < l$. The available bandwidth of the path is equal to the bandwidth of the link having minimum bandwidth among all links on the path, and is calculated as:

$$Available\ bandwidth = \min_{1 \leq i < l} Bandwidth(p_i, p_{i+1}) \tag{4.2}$$

The path cost *C* is calculated using bandwidth and the hop count as follows:

$$Path\ Cost\ C = \left[1 - \left(\frac{AvailableBandwidth - (HopCount * 10)}{1000}\right)\right] \qquad (4.3)$$

where constant 10 is used as a penalty for longer paths and 1000 is used for normalization because maximum allowed link bandwidth is 1000 Mbps (all link bandwidths are randomly generated between 500 and 1000). Penalty for a larger value of hop count forces the routing algorithm to choose shorter paths (with lesser hop count).

Table 4.2 shows the shortest path calculation (with minimum cost) between host h1 (connected to switch S1) and host h50 (connected to switch S5) in topology T4 for current traffic. The first column lists all the possible paths between host h1 and host 50. As stated earlier, the routing algorithm calculates 3 shortest paths using equation 4.2 and dynamically distribute the traffic along these paths by taking utilization into account.

PATH	Available bandwidth (Mbps)	Hops	Path cost
1-5	629	2	0.39
1-2-5	650	3	0.38
1-3-5	738	3	0.29
1-4-5	702	3	0.33
1-2-3-5	650	4	0.39
1-2-4-5	650	4	0.39
1-3-2-5	651	4	0.39
1-3-4-5	710	4	0.33
1-4-2-5	651	4	0.39
1-4-3-5	702	4	0.34
1-2-3-4-5	650	5	0.40
1-2-4-3-5	650	5	0.40
1-3-2-4-5	710	5	0.34
1-3-4-2-5	651	5	0.40
1-4-2-3-5	702	5	0.35
1-4-3-2-5	651	5	0.40

Table 4.3: Three shortest paths calculated between host h1 (connected to switch S1) and host h50 (connected to switch S5) in topology T4 for current traffic

Figure 4.10 shows the traffic flow along the 3 shortest paths between hosts h1 and h50 (S1-S3-S5, S1-S4-S5, and S1-S3-S4-S5). The QoS parameters are observed over the 15 seconds for the traffic flow generated using TCP protocol with window size 85.3 Kbytes over the single shortest path (S1-S3-S5). End-to-end packet loss, jitter, and throughput are calculated at host h50. The average throughput over the 15 seconds window (with an interval size of 1 second) is 296 Mbps with jitter 1.18 ms. The total packet loss is 5 %.

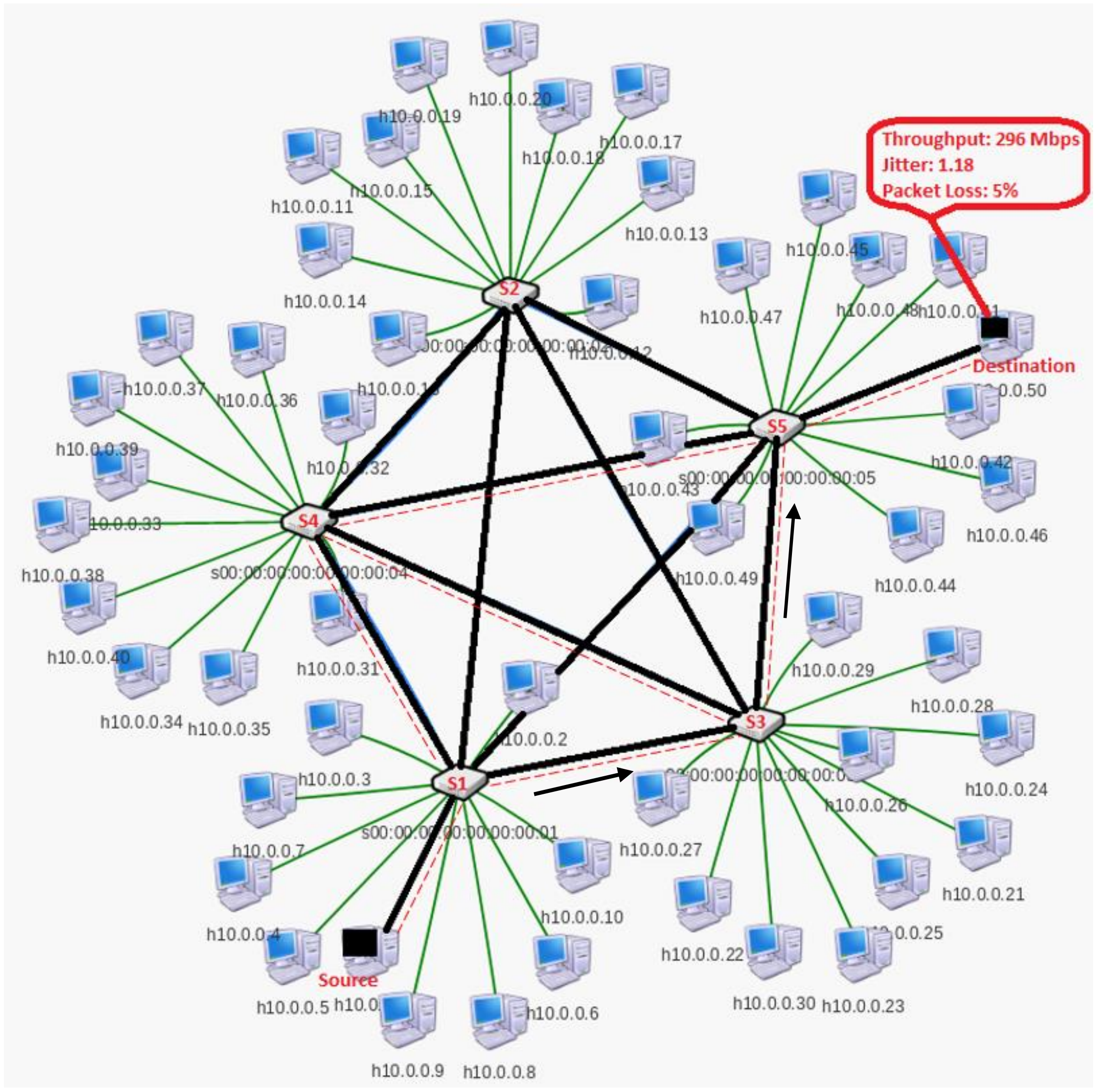


Figure 4.10: QoS parameters calculation in topology T4 for current traffic between hosts h1 and h50

After passing the current traffic, available bandwidth changes because of the link utilization that affects the path cost calculation. For forecasted traffic, the available bandwidth of a path is the minimum value of the difference between the bandwidth capacity and the used bandwidth among all links along the path. As a result, the equation 4.1 changes as follows:

$$AvailableBandwidth = \min_{1 \le i < l} BandwidthCapacity(P_i, P_{i+1}) - BandwidthUsed(P_i, P_{i+1}) \quad (4.4)$$

PATH	Available bandwidth (Mbps)	Hops	Path cost
1-5	509.49	2	0.51
1-2-5	474.50	3	0.56
1-3-5	383.76	3	0.65
1-4-5	231.66	3	0.80
1-2-3-5	383.76	4	0.66
1-2-4-5	347.49	4	0.69
1-3-2-5	430.35	4	0.61
1-3-4-5	430.35	4	0.61
1-4-2-5	231.66	4	0.81
1-4-3-5	231.66	4	0.81
1-2-3-4-5	474.5	5	0.58
1-2-4-3-5	347.49	5	0.70
1-3-2-4-5	347.49	5	0.70
1-3-4-2-5	347.49	5	0.70
1-4-2-3-5	231.66	5	0.82
1-4-3-2-5	231.66	5	0.82

Table 4.4: Three shortest paths calculated between host h1 (connected to switch S1) and host h50 (connected to switch S5) in topology T4 for forecasted traffic

Using equation 4.3, updated three shortest paths (with minimum path cost) for forecasted traffic in topology T4 between hosts h1 and h50 are S1-S5, S1-S2-S5, and S1-S2-S3-S4-S5. Figure 4.11 shows the traffic distribution along these three paths. The QoS parameters are calculated at the host h50 following the same procedure as the current traffic scenario. The average throughput is 349 Mbps with jitter 1.33 ms. The packet loss is increased to 8 % (3 % more than the current traffic scenario), the reason is the increase in the link utilization.

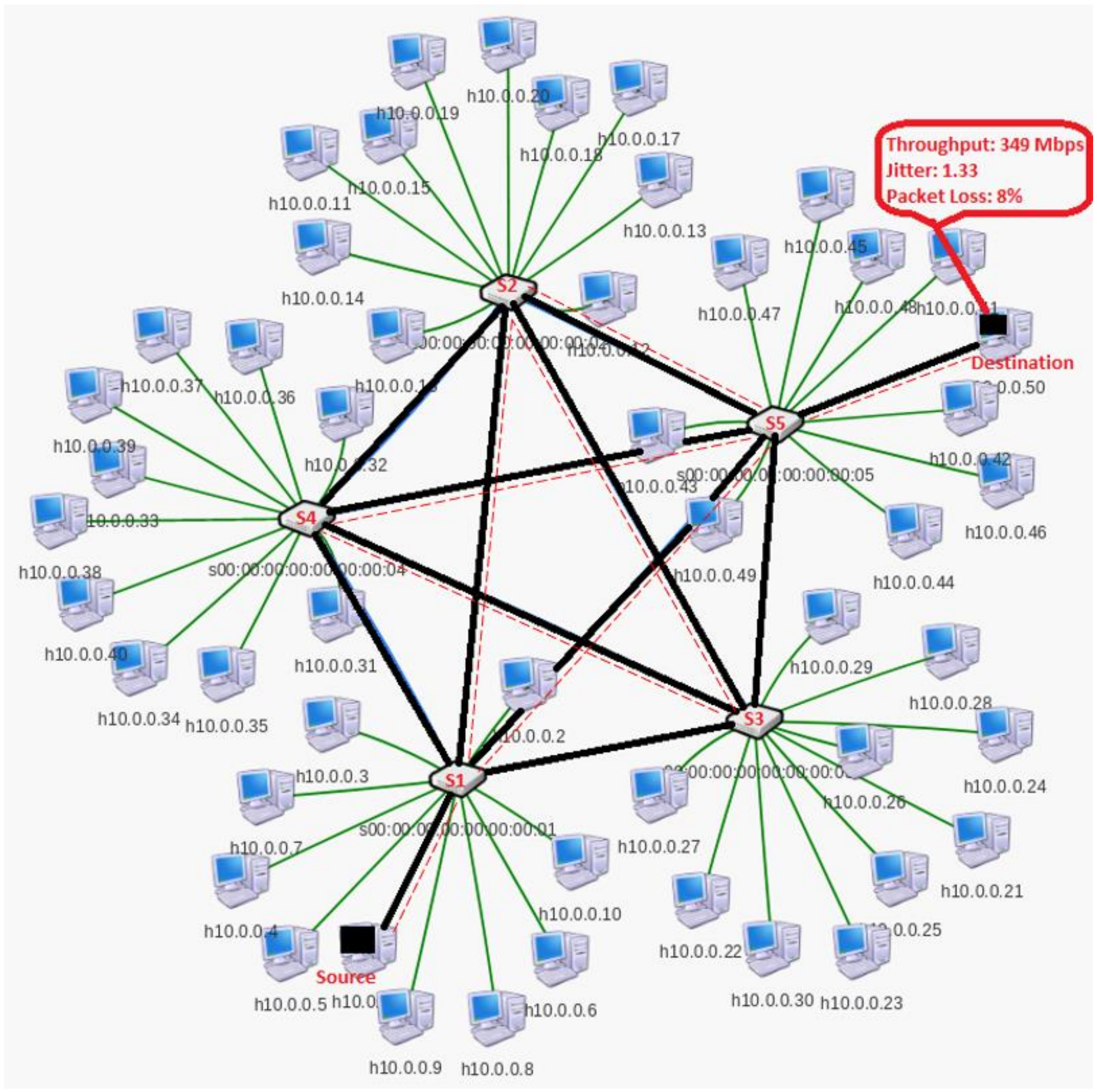


Figure 4.11: QoS parameters collection for forecasted traffic in topology T4

The same procedure is repeated for QoS statistics calculation for all other topologies and the calculated parameters are analyzed for both the current and the forecasted traffic.

Chapter 5

5 Implementation and Results

NIMA is a novel framework that analyzes the impact on QoS parameters on forecasted traffic by initializing the setup with simulating the current network status. This analysis is helpful in identifying the potential high-risk links subject to failure, predicting the congestion level of the network, and indicating the impact of network-wide latency. Analysis can also help ISPs in planning the network extension and upgrades so that they are always ready to handle the future network traffic. In this chapter, we outline the NIMA methodology and discuss the QoS parameters results in detail.

5.1 Experimental Setup

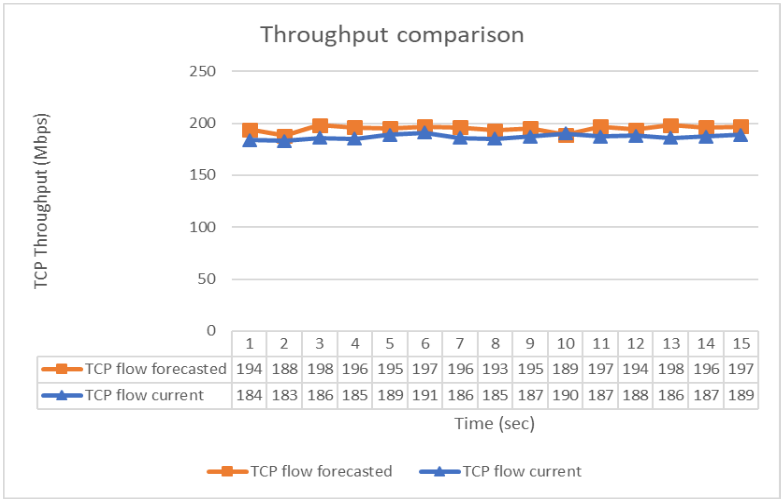
For our experiments, we use the Mininet emulator [22] in combination with the Floodlight controller [28]. These applications (e.g., Mininet, and Floodlight) run on an Oracle VirtualBox 5.2.12 virtual machine using 64-bit Ubuntu 18.04 LTS with 4GB of RAM. The virtual machine is installed in Lenovo Flex 5 laptop with Intel® Core™ i7-8500U CPU @ 2.00 GHz, 16 GB of RAM and 64-bit Windows 10 operating system. To minimize any randomness coming from the virtual machine setup so that results are reproducible with minimal variation, all the experiments are done in a dedicated, uninterrupted, and stable platform with fixed Mininet parameter settings for CPU, memory allocation, etc. Mininet has an inbuilt implementation of all switches and hosts. We use the default settings of OpenFlow and the Floodlight controller. Each experiment is run 10 times and QoS parameters like link utilization, latency, packet loss, throughput, and jitter are calculated at the end of each run. We then calculate the average of all QoS parameters for all topologies and compare the before and after scenarios to check the impact of forecasted traffic.

The network topology of our experiment is emulated by Mininet 2.3.0 [22] with Open vSwitch 2.5.4 supported by OpenFlow version 1.3 [45]. Mininet is a widely used tool that can be used as both an emulator to emulate the network architecture, and a simulator to simulate multiple experimental scenarios [48]. In the proposed framework, we first emulate our topology using Python. Though Mininet provides inbuilt command-based functionality to quickly create straight-forward and simpler topologies, we customized our topology design to mimic real-world scenarios where different network links can have different bandwidth. We then use Mininet as a simulator to pass traffic flows and measure the QoS metrics: latency, jitter, and packet loss, throughput, link Utilization. In this simulator, we run a Python program agent to simulate traffic flows of each host. Furthermore, Floodlight 1.2, a Java-based controller [49] is used as the SDN controller for all of our experiments. Floodlight supports physical and virtual switches and has a large developers’ community. Floodlight is included in the architecture during the emulation of the network. In our experiments, we used only one controller. However, there could be multiple controllers, especially in large-scale networks.

5.2 Comparison of QoS parameters

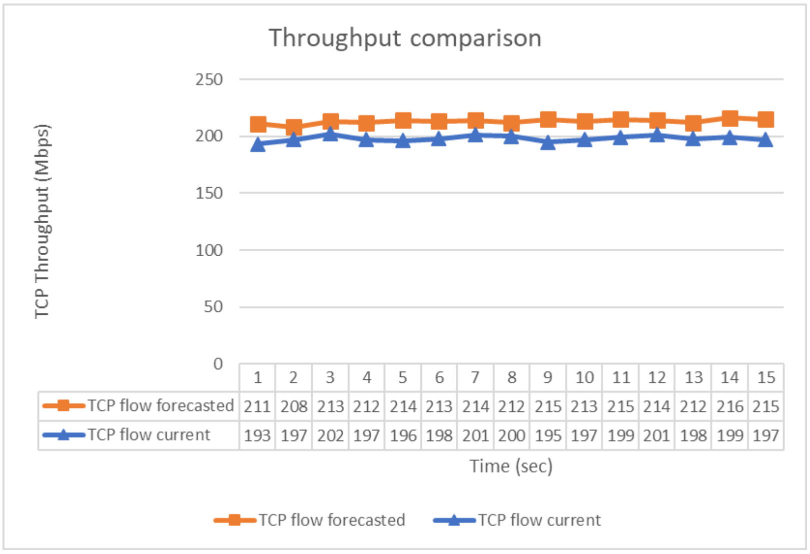
5.2.1 Throughput

We selected two farthest hosts for each topology to calculate the network throughput. The TCP protocol is used with window size 85.3 Kbytes and the throughput is observed over the 15 seconds window for both current and forecasted traffic. The following figures (Fig 5.1 – 5.5) show the throughput comparisons for the respective topologies (T1-T5).



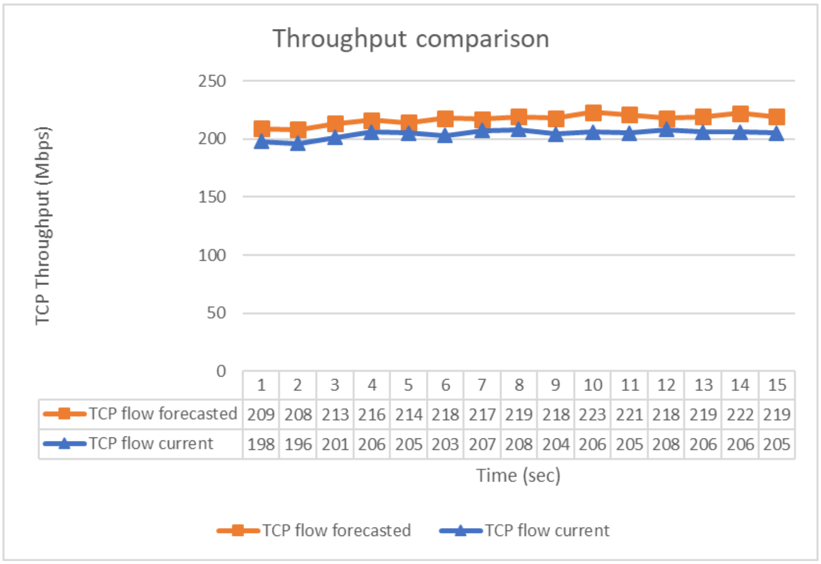
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TCP flow forecasted	194	188	198	196	195	197	196	193	195	189	197	194	198	196	197
TCP flow current	184	183	186	185	189	191	186	185	187	190	187	188	186	187	189

Figure 5.1: Throughput Comparison for current and forecasted traffic (Topology T1)



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TCP flow forecasted	211	208	213	212	214	213	214	212	215	213	215	214	212	216	215
TCP flow current	193	197	202	197	196	198	201	200	195	197	199	201	198	199	197

Figure 5.2: Throughput Comparison for current and forecasted traffic (TopologyT2)

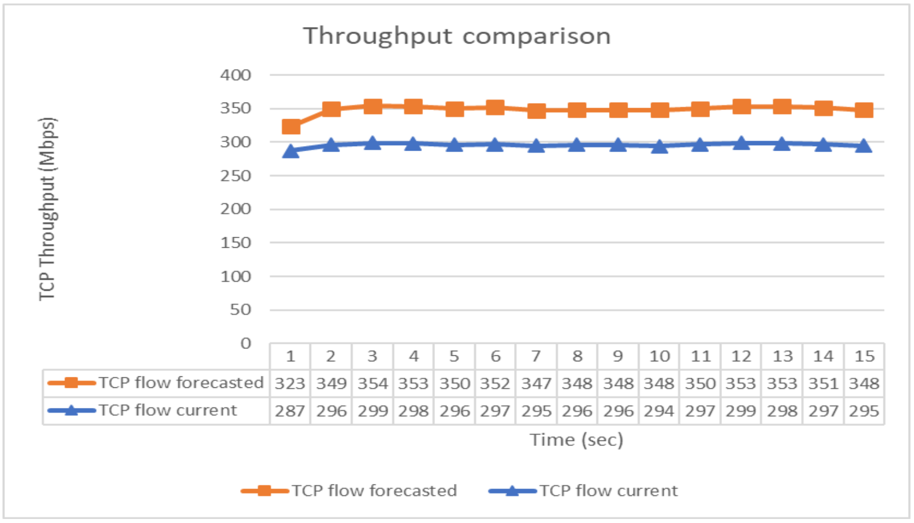


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TCP flow forecasted	209	208	213	216	214	218	217	219	218	223	221	218	219	222	219
TCP flow current	198	196	201	206	205	203	207	208	204	206	205	208	206	206	205

Figure 5.3: Throughput Comparison for current and forecasted traffic (TopologyT3)

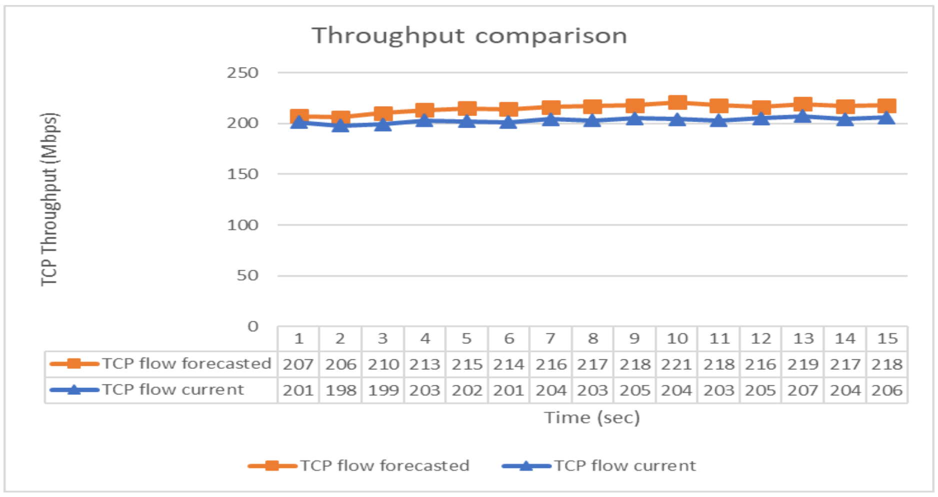
We selected two farthest hosts (h1-h10.0.0.1 and h5-h10.0.0.5 in case of topology T1, h1-h10.0.0.1 and h10-h10.0.0.10 in case of topology T2, h1-h10.0.0.1 and h25-h10.0.0.25 in case of topology T3, h1-h10.0.0.1 and h50-h10.0.0.50 in case of topology T4, and h1-h10.0.0.1 and h100-h10.0.0.100 in case of topology T5), to calculate network throughput.

We observed that the average throughput increased for the forecasted traffic when compared to the current traffic, from 187 Mbps to 195 Mbps for T1, 198 Mbps to 213 Mbps for T2, 204 Mbps to 217 Mbps for T3, 296 Mbps to 349 Mbps for T4, and 203 Mbps to 215 Mbps for T5.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TCP flow forecasted	323	349	354	353	350	352	347	348	348	348	350	353	353	351	348
TCP flow current	287	296	299	298	296	297	295	296	296	294	297	299	298	297	295

Figure 5.4: Throughput Comparison for current and forecasted traffic (Topology T4)

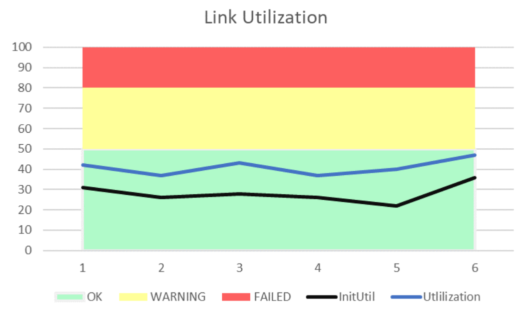


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TCP flow forecasted	207	206	210	213	215	214	216	217	218	221	218	216	219	217	218
TCP flow current	201	198	199	203	202	201	204	203	205	204	203	205	207	204	206

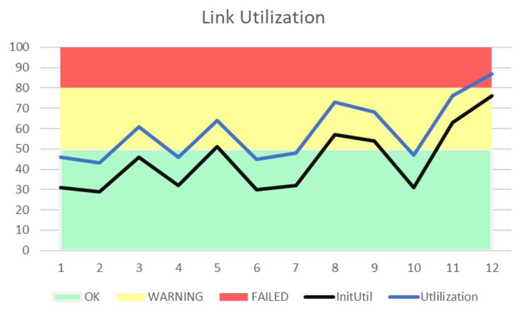
Figure 5.5: Throughput Comparison for current and forecasted traffic (Topology T5)

5.2.2 Link Utilization

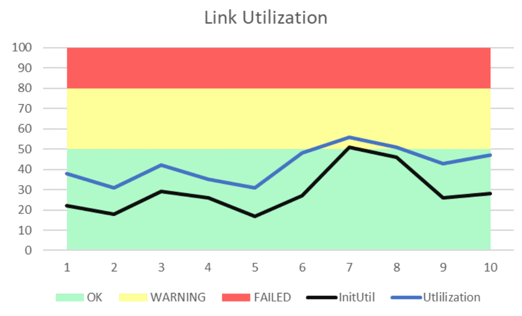
NIMA can monitor the links and raise a flag for high-risk links that may fail in the future (as per forecasted traffic data). Figure 5.6 shows link utilization (in %), for both current and forecasted traffic in topologies T1-T5. We benchmarked link status into three categories: OK (shown in green, link utilization up to 50%), WARNING (shown in yellow, link utilization between 50% and 80%), and FAILED (shown in red, link utilization more than 80%). It is found that the link utilization is elevated for forecasted traffic almost following the same pattern as current traffic. The links on the border range of "OK" status moved to the "WARNING" status and the links on the border range of "WARNING" status moved to the "FAILED" status. No link failure occurred in topologies T1 and T3. Link 12 in T2 (between switches s3 and s8) and link 7 in T4 (between switches s1 and s4) failed for forecasted traffic. Topology T5 was most impacted with two failed links (link 6 between switches s1 and s3, and link 7 between switches s1 and s4).



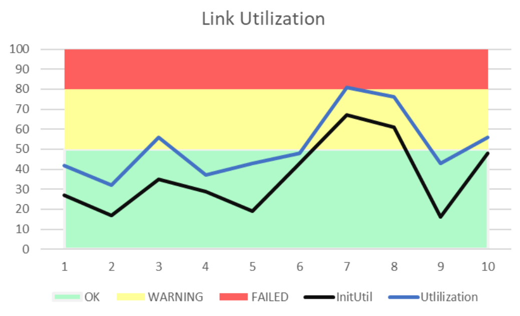
(a) Topology T1



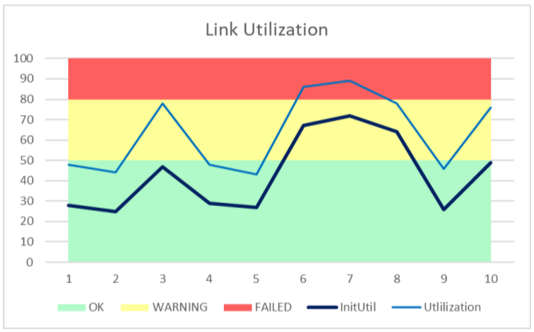
(b) Topology T2



(c) Topology T3



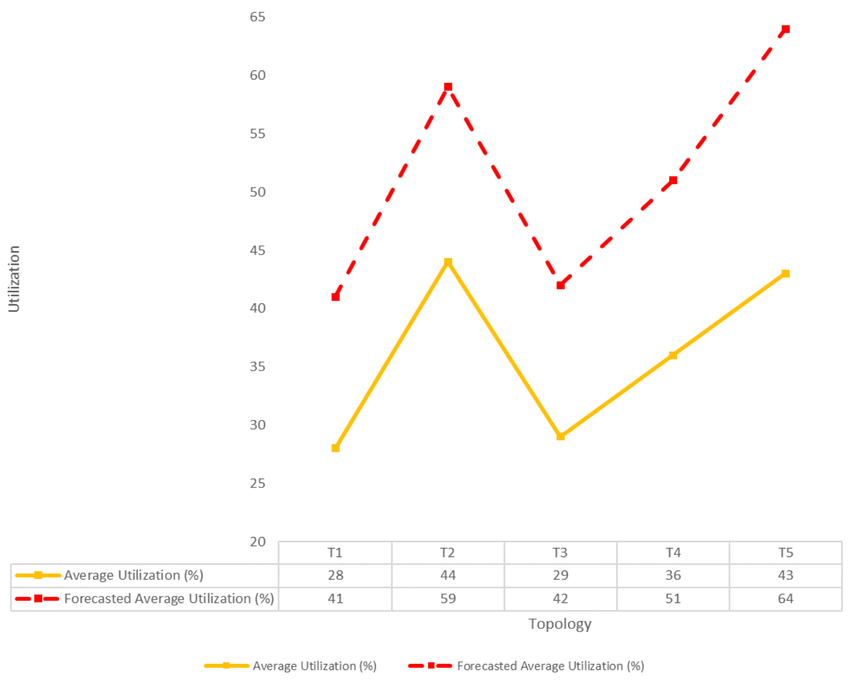
(d) Topology T4



(e) Topology T5

Figure 5.6: Link Utilization for current and forecasted traffic

As shown in figure 5.7, overall link utilization went up for the forecasted traffic because of an increase in network traffic. Figure 5.8 shows a comparison of average utilization in topologies T1-T5 for current, and forecasted traffic. The average utilization is in the range of 28-44% for current traffic and in the range 41-64% for forecasted traffic.



	T1	T2	T3	T4	T5
Average Utilization (%)	28	44	29	36	43
Forecasted Average Utilization (%)	41	59	42	51	64

Figure 5.7: Comparison of average utilization in topologies T1-T5 for current and forecasted traffic

5.2.3 Latency

Tables 5.1-5.5 summarize the network statistics for topologies T1-T respectively. The latency, bandwidth, link utilization with current and forecasted traffic, amount of passing traffic, change in utilization and Network status are shown. NIMA auto-updates the stats and raise the flags for potential troublesome links (with Warning or Failed status).

Link	Latency	Bandwidth (Mbps)	Current Utilization	Current Traffic (Mb)	Current Status	Forecasted Utilization	Status	Traffic (Mb)	Utilization Change
s1-s2	29	650	31	201.50	OK	42	OK	273.00	11
s2-s3	26	922	26	239.72	OK	37	OK	341.14	11
s3-s4	27	878	28	245.84	OK	43	OK	377.54	15
s4-s5	25	710	26	184.6	OK	37	OK	262.70	11
s5-s1	24	629	22	138.38	OK	40	OK	251.60	18
s2-s4	31	755	36	271.80	OK	47	OK	354.85	11

Table 5.1: Latency, Bandwidth, Link Utilization with current and forecasted traffic, and network status are shown for topology T1

As shown in Table 5.1, for topology T1, all links were under normal utilization ("OK" status with utilization under 50%) both before and after passing the forecasted traffic. The latency is in the range 24-31ms (milliseconds).

Link	Latency	Bandwidth (Mbps)	Current Utilization	Current Traffic (Mb)	Current Status	Forecasted Utilization	Status	Traffic (Mb)	Utilization Change
s1-s2	29	650	31	201.50	OK	46	OK	299.00	15
s2-s3	28	922	29	267.38	OK	43	OK	396.46	14
s3-s4	30	878	46	403.88	OK	61	WARNING	535.58	15
s4-s5	29	710	32	227.20	OK	46	OK	326.60	14
s5-s6	32	629	51	320.79	WARNING	64	WARNING	402.56	13
s6-s7	28	755	30	226.50	OK	45	OK	339.75	15
s7-s8	31	702	32	224.64	OK	48	OK	336.96	16
s8-s9	34	891	57	507.87	WARNING	73	WARNING	650.43	16
s9-s10	32	651	54	351.54	WARNING	68	WARNING	442.68	14
s10-s1	29	738	31	228.78	OK	47	OK	346.86	16
s1-s6	39	746	63	469.98	WARNING	76	WARNING	566.96	13
s3-s8	42	683	76	519.08	WARNING	87	FAILED	594.21	11

Table 5.2: Latency, Bandwidth, Link Utilization with current and forecasted traffic, and Network status are shown for topology T2

For topology T2, latency is in the range 28-42ms see table 5.2. Highly utilized links showed an elevation in the latency. As shown in Table 5.3, the latency range is in the range 17-28ms, which aligns with the majority of links showing normal utilization.

Link	Latency	Bandwidth (Mbps)	Current Utilization	Current Traffic (Mb)	Current Status	Forecasted Utilization	Status	Traffic (Mb)	Utilization Change
s1-s2	18	650	22	143.00	OK	38	OK	247.00	16
s2-s3	21	922	18	165.96	OK	31	OK	285.82	13
s3-s4	17	878	29	254.62	OK	42	OK	368.76	13
s4-s5	19	710	26	184.60	OK	35	OK	248.50	9
s5-s1	21	629	17	106.93	OK	31	OK	194.99	14
s1-s3	31	755	27	203.85	OK	48	OK	362.40	21
s1-s4	28	702	51	358.02	WARNING	56	WARNING	393.12	5
s2-s4	26	891	46	409.86	OK	51	WARNING	454.41	5
s2-s5	22	651	26	169.26	OK	43	OK	279.93	17
s3-s5	21	738	28	206.64	OK	47	OK	346.86	19

Table 5.3: Latency, Bandwidth, Link Utilization with current and forecasted traffic, and Network status are shown for Topology T3

Link	Latency	Bandwidth (Mbps)	Current Utilization	Current Traffic (Mb)	Current Status	Forecasted Utilization	Status	Traffic (Mb)	Utilization Change
s1-s2	21	650	27	175.50	OK	42	OK	273.00	15
s2-s3	16	922	17	156.74	OK	32	OK	295.04	15
s3-s4	32	878	35	307.30	OK	56	WARNING	491.68	21
s4-s5	19	710	29	205.90	OK	37	OK	262.70	8
s5-s1	21	629	19	119.51	OK	43	OK	270.47	24
s1-s3	25	755	43	324.65	OK	48	OK	362.40	5
s1-s4	48	702	67	470.34	WARNING	81	FAILED	568.62	14
s2-s4	39	891	61	543.51	WARNING	76	WARNING	677.16	15
s2-s5	23	651	16	104.16	OK	43	OK	279.93	27
s3-s5	28	738	48	354.24	OK	56	WARNING	413.28	8

Table 5.4: Latency, Bandwidth, Link Utilization with current and forecasted traffic, and Network status are shown for the Topology T4

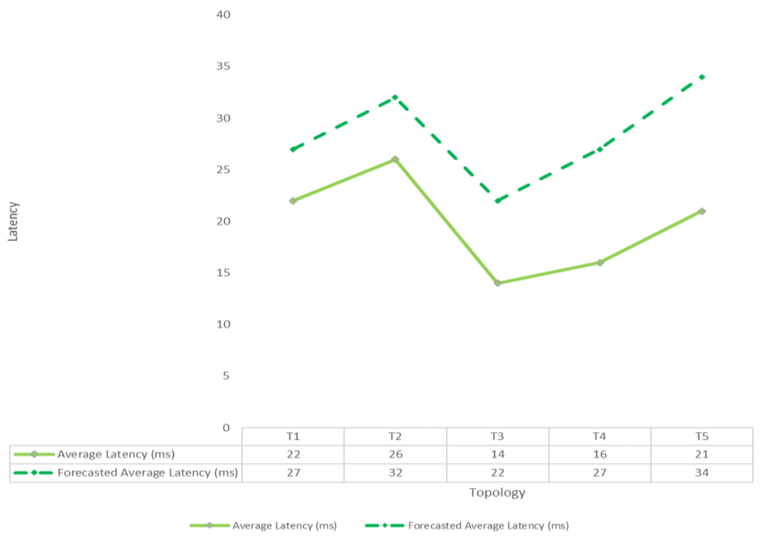
As shown in Tables 5.4, 5.5, and It is observed that the link failures in T4 and T5 resulted in an increase in latency, but the impact on the overall network was minimal because both T4 and T5 are full-mesh topologies and packets were rerouted through alternative paths.

As expected, troublesome links (highly utilized links which are prone to failure) showed an increase in latency for the forecasted traffic.

Link	Latency	Bandwidth (Mbps)	Current Utilization	Current Traffic (Mb)	Current Status	Forecasted Utilization	Status	Traffic (Mb)	Utilization Change
s1-s2	32	650	28	182.00	OK	48	OK	312.00	20
s2-s3	29	922	25	230.50	OK	44	OK	405.68	19
s3-s4	37	878	47	412.66	OK	78	WARNING	684.84	31
s4-s5	30	710	29	205.90	OK	48	OK	340.80	19
s5-s1	28	629	27	169.83	OK	43	OK	270.47	16
s1-s3	38	755	67	505.85	WARNING	86	FAILED	649.30	19
s1-s4	43	702	72	505.44	WARNING	89	FAILED	624.78	17
s2-s4	36	891	64	570.24	WARNING	78	WARNING	694.98	14
s2-s5	31	651	26	169.26	OK	46	OK	299.46	20
s3-s5	33	738	49	361.62	OK	76	WARNING	560.88	27

Table 5.5: Latency, Bandwidth, Link Utilization with current and forecasted traffic, and Network status are shown for the topology T5

Figure 5.8 shows a comparison of average latency in topologies T1-T5 for current, and forecasted traffic. The average latency is in the range 14-26ms for current traffic, and in the range 22-34ms for forecasted traffic.



	T1	T2	T3	T4	T5
Average Latency (ms)	22	26	14	16	21
Forecasted Average Latency (ms)	27	32	22	27	34

Figure 5.8: Comparison of average Latency in topologies T1- T5 for current, and forecasted traffic

We observed the direct impact of highly utilized failed links on the percent increase in the packet loss. Links in “WARNING” or “FAILED” status resulted in an increase in packet loss, average utilization, and latency for the forecasted traffic.

5.2.4 Jitter

Tables 5.6 and 5.7 summarize the current traffic statistics, forecasted traffic statistics respectively, for all 5 topologies considered in our experimental setup. It is observed that in the case of a link failure, the partial-mesh topologies had a sharp increase in latency in comparison to full-mesh topologies. As we increased the number of hosts in full-mesh topologies (T3, T4, T5), they become more failure-prone; an increase in latency and jitter was seen. T4 gave better throughput than T3 and T5, suggesting optimized hosts limit per switch.

Topology	**Packet Loss (%)**	**Average Latency (ms)**	**Average Jitter (ms)**	**Average Throughput (Mbps)**	**Average Utilization**
T1	**0**	**22**	**1.23**	**187**	**28**
T2	**2**	**26**	**1.28**	**198**	**44**
T3	**2**	**14**	**1.12**	**204**	**29**
T4	**5**	**16**	**1.18**	**296**	**36**
T5	**7**	**21**	**1.21**	**203**	**43**

Table 5.6: Comparison of QoS parameters in topologies T1-T5 for current traffic

Topology	**Packet Loss (%)**	**Average Latency (ms)**	**Average Jitter (ms)**	**Average Throughput (Mbps)**	**Average Utilization**
T1	**0**	**27**	**1.34**	**195**	**41**
T2	**5**	**32**	**1.42**	**213**	**59**
T3	**6**	**22**	**1.24**	**217**	**42**
T4	**8**	**27**	**1.33**	**349**	**51**
T5	**12**	**34**	**1.48**	**215**	**64**

Table 5.7: Comparison of QoS parameters in topologies T1-T5 for forecasted traffic

Figure 5.9 shows a comparison of average jitter in topologies T1-T5 for current, and forecasted traffic. Among all topologies, observed jitter is in the range 1.12ms-1.28ms for the current traffic and in the range 1.24ms-1.48ms.

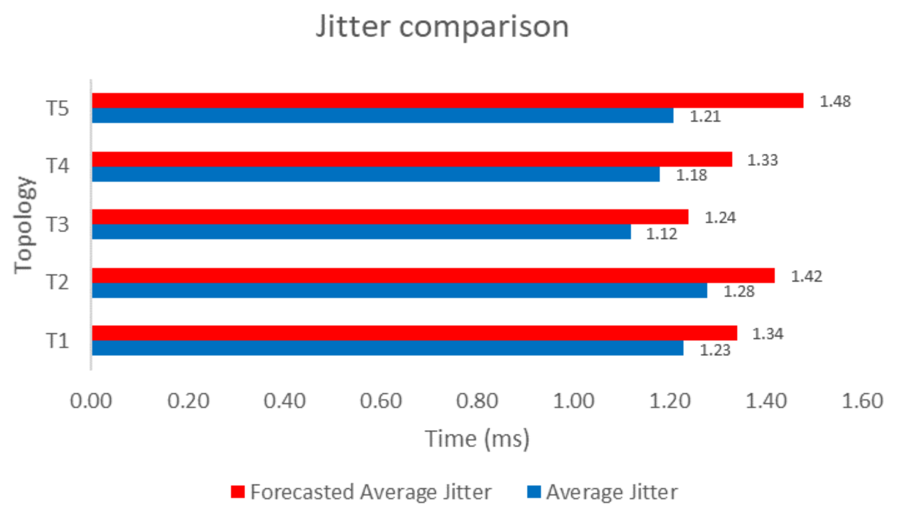


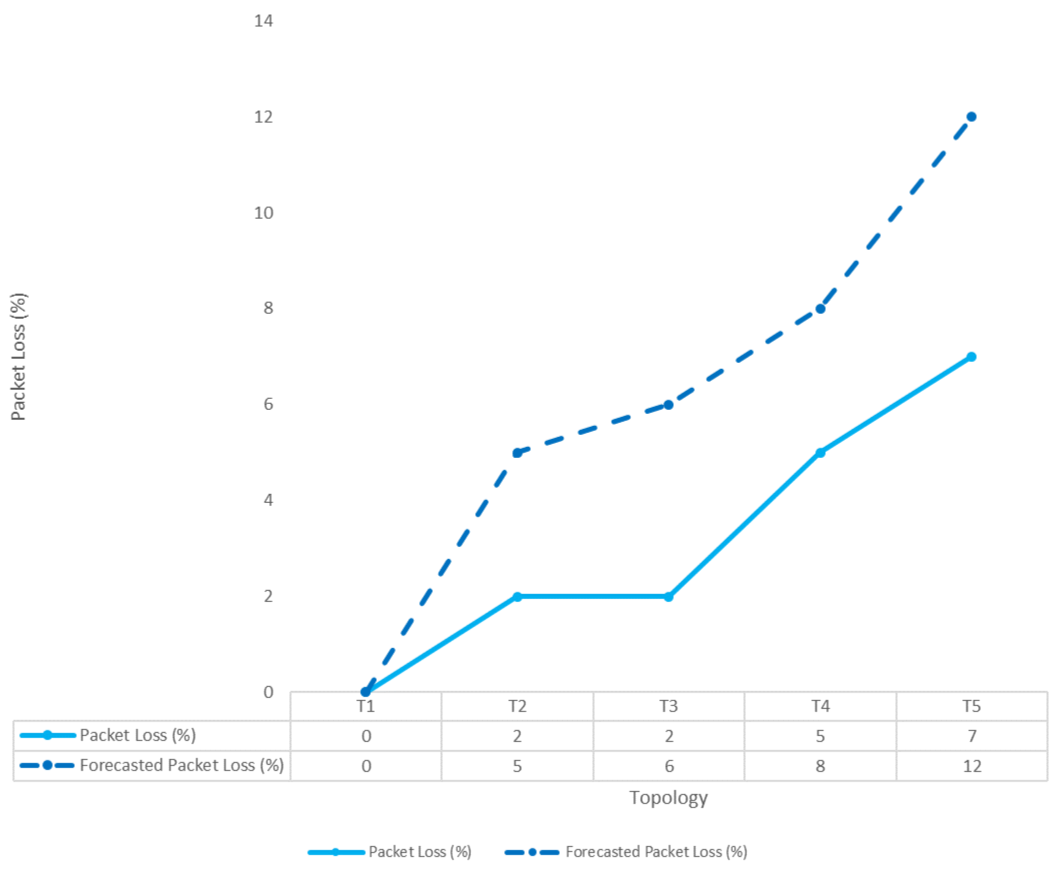
Figure 5.9: Comparison of average jitter in topologies T1-T5 for current, and forecasted traffic

5.2.5 Packet Loss

Scalability in topologies also led to an increase in packet loss. Topology T5 being the largest topology had the maximum packet loss. The maximum packet loss percentage observed is 12%. The packet loss is in correlation to the number of highly utilized links (near failure and failed links). Table 5.8 shows the percent change in link status based on utilization for all considered topologies. For topology T1, all links remained utilized under 50% (status: "OK") when forecasted traffic is passed by initializing the network to the current traffic. T4 and T5 showed highly utilized links with T5 showing 20% of links going from WARNING to FAILED status. T4 and T5 showed 30% links in total, that remained in WARNING status.

Link Status / Topology	T1	T2	T3	T4	T5
Remains OK	100%	50%	80%	60%	50%
OK to WARNING	0%	8.5%	10%	20%	20%
Remains WARNING	0%	33%	10%	10%	10%
OK to FAILED	0%	0%	0%	0%	0%
WARNING TO FAILED	0%	8.5%	0%	10%	20%

Table 5.8: Utilization based change in link status for topologies T1-T5



	T1	T2	T3	T4	T5
Packet Loss (%)	0	2	2	5	7
Forecasted Packet Loss (%)	0	5	6	8	12

Figure 5.10: Comparison of average packet loss in topologies T1-T5 for current, and forecasted traffic

Figure 5.10 shows a comparison of average packet loss in topologies T1-T5 for current, and forecasted traffic. It is observed that the increase in packet loss aligns with an increase in utilization. The highly utilized links directly impact the percent increase in packet loss.

5.2.6 Load Balancing

Table 5.9 shows the Load Balancing Percentage (LBP) for current traffic (single shortest path routing using Dijkstra’s algorithm) and forecasted traffic (multi-path routing using Yen’s algorithm). More the LBP value, more equally distributed traffic. It is seen that the multi-path routing improved the load balancing, with an exception of topology T5 where highly utilized and failed links resulted in a minimal reduction in LBP value. The LBP value for ‘n' number of links is calculated as:

$$LBP = \ 100 - \left(\frac{\sum_{i=1}^{n} |AverageUtilization - LinkUtilization_i|}{n} \right) \tag{5.1}$$

Topology	LBP (Current traffic)	LBP (Forecasted traffic)	Variation
T1	**96.44%**	**97.00%**	**+0.56%**
T2	**86.50%**	**87.17%**	**+0.67%**
T3	**92.20%**	**93.20%**	**+1.00%**
T4	**85.16 %**	**87.32 %**	**+2.16%**
T5	**83.60%**	**82.20%**	**-1.4%**

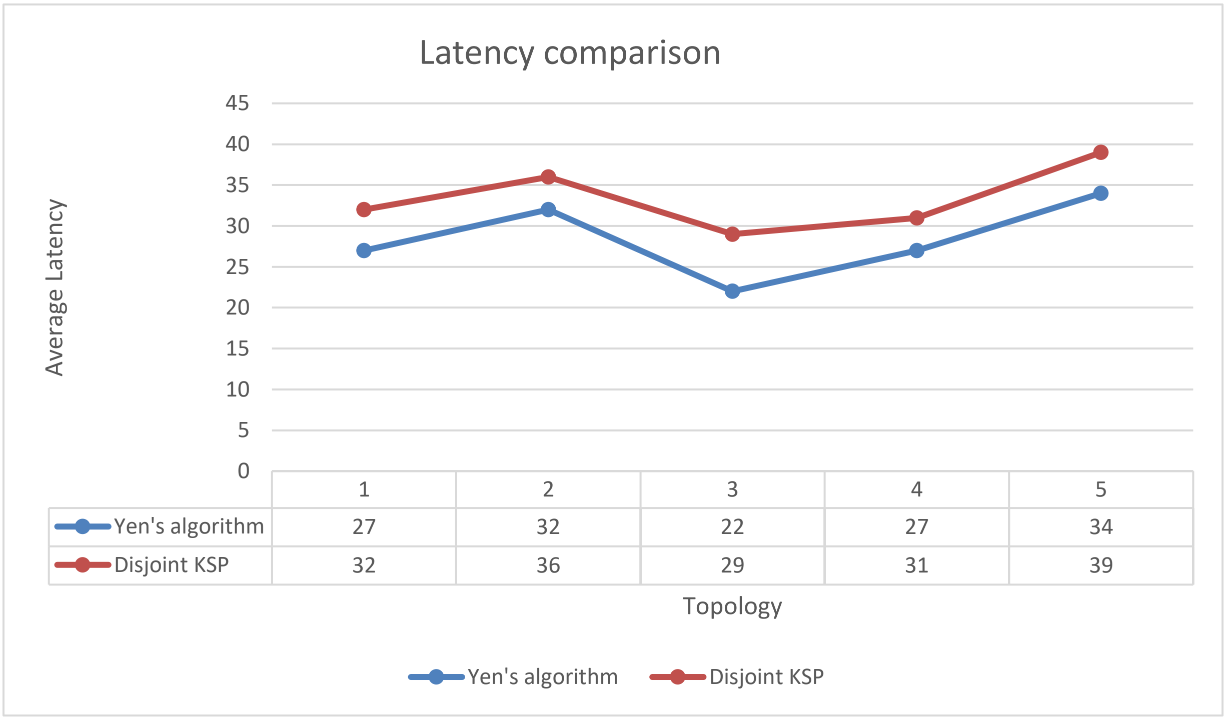
Table 5.9: LBP values for current and forecasted traffic

Overall, full-mesh topologies showed better throughput and lower latency and jitter as multi-path routing using Yen's algorithm helped in selecting alternate paths in case of link failures.

5.3 Comparison of multi-path Routing Algorithms

Floodlight controller uses Yen's algorithm for multi-path routing. Many algorithms have been proposed in the past for multi-path routing problems, such as Yen's algorithm, Disjoint k-Shortest Path (KSP) algorithm, etc. [38]. Yen's algorithm uses multiple iterations over Dijkstra's algorithm to find multiple "k" shortest paths. The Disjoint KSP algorithm only works with acyclic directed graphs. Suurballe and Tarjan proposed an algorithm that finds a pair of link disjoint paths between a source and all the other graph nodes. Santos [38] compared various multi-path routing algorithms using the Floodlight controller and showed that the Yen's algorithm outperforms the link-disjoint-KSP algorithms and the Self-adaptive Multiple Constraints Routing Algorithm (SAMCRA) for most of their experiment test scenarios. The link disjoint KSP algorithm was second best in terms of throughput and latency comparisons.

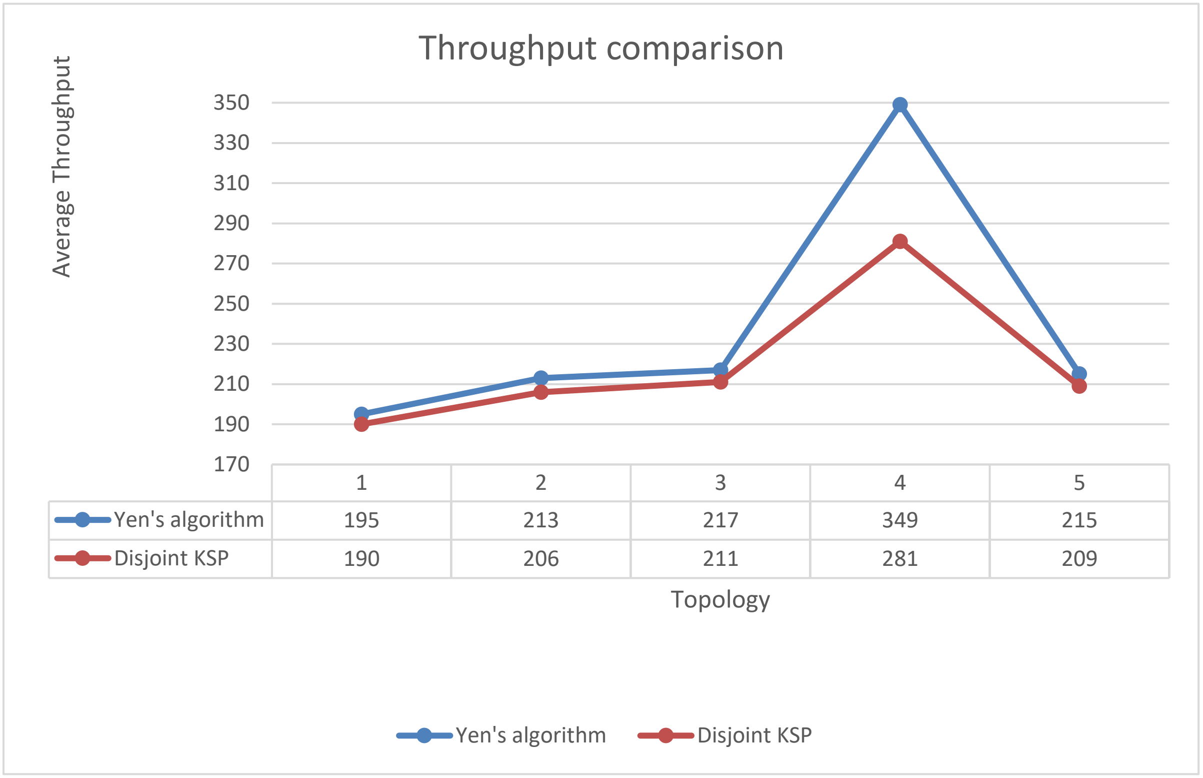
We first simulated our networks (all five topologies) with current traffic routed using Dijkstra's algorithm and then compared the performance of Yen's algorithm, and the Suurballe's k-disjoint shortest multi-path algorithm on the simulated networks by routing the forecasted traffic. We re-run our experiments using Suurballe’s k-disjoint shortest path algorithm instead of Yen’s algorithm and keeping all other experimental settings unchanged. Figure 5.11 shows the latency comparison of algorithms for our topologies T1-T5. We also did throughput comparison as shown in Figure 5.12 and load balancing comparison as shown in figure 5.13 for both algorithms. Our results are in accordance with the findings of Santos. Yen’s algorithm outperforms the k-disjoint shortest path algorithm in terms of QoS parameters latency and throughput both. The performance difference is significant when traffic and the number of links increase. Yen’s algorithm also achieved a better load balancing score for all of our experimental topologies T1-T5.



	1	2	3	4	5
Yen's algorithm	27	32	22	27	34
Disjoint KSP	32	36	29	31	39

Figure 5.11: Latency comparison of Yen’s algorithm and Suurballe's k-disjoint shortest path algorithm

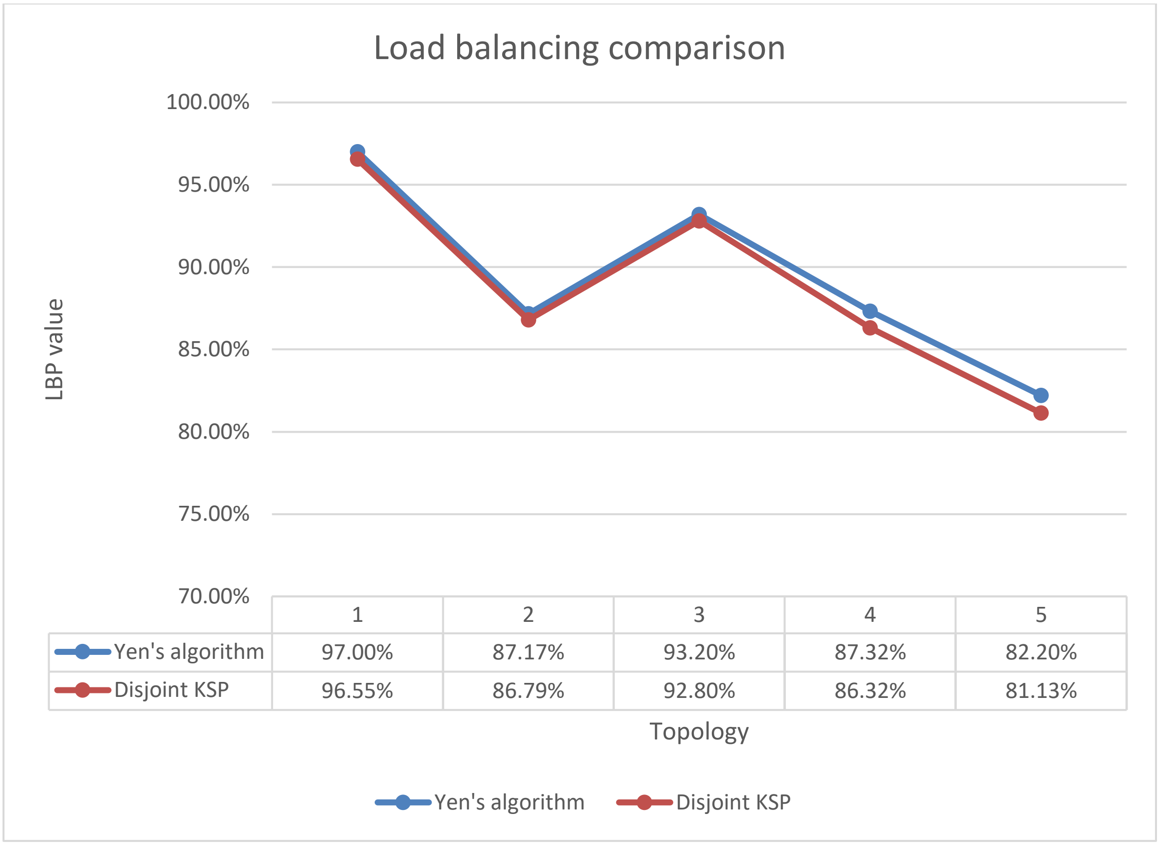
As shown in figure 5.11, the experimental results clearly indicate the performance of Yen’s algorithm is better than the Disjoint KSP algorithm for the forecasted traffic. We compared the average latency of all links for all topologies over 15 seconds window. Disjoint KSP takes more time to transfer the packets from a given source to destination for all topologies. For example, as shown in Figure 5.11, the average latency of topology T3 among multi-path algorithms is recorded as 22 ms for Yen's algorithm and 29 ms for Disjoint KSP for forecasted traffic.



	1	2	3	4	5
Yen's algorithm	195	213	217	349	215
Disjoint KSP	190	206	211	281	209

Figure 5.12: Throughput comparison of Yen's algorithm and Suurballe's k-disjoint shortest path algorithm

As shown in the above figure 5.12 the average throughput over a time interval of 15 seconds is observed. Yen's algorithm outperforms the Disjoint KSP algorithm when evaluated for forecasted traffic under the same experimental conditions. The highest throughput is observed in topology T4 which uses five switches with each switch connected to ten hosts (50 hosts in total). The forecasted traffic is simulated, the average throughput rises to 349 Mbps using Yen's algorithm and 281 Mbps using Disjoint KSP. This clearly indicates the outperformance of Yen's algorithm in comparison to Disjoint KSP.



	1	2	3	4	5
Yen's algorithm	97.00%	87.17%	93.20%	87.32%	82.20%
Disjoint KSP	96.55%	86.79%	92.80%	86.32%	81.13%

Figure 5.13: Load Balancing comparison Yen's algorithm, and Suurballe's k-disjoint multi-path routing algorithm

The Load Balancing comparison between two multipath algorithms (Yen’s and Suurballe’s Disjoint KSP) is shown in figure 5.13. The figure clearly indicates that the Load Balancing Percentage (LBP) value for Yen’s algorithm is always higher than Disjoint KSP for the forecasted traffic.

5.4 Discussion

We observe that average throughput increased for forecasted traffic when compared to current traffic. The highest throughput is observed in topology T4 which uses five switches with each switch connected to ten hosts (50 hosts in total). The average throughput is 296Mbps when Dijkstra’s algorithm is used on current traffic. However, when forecasted traffic is simulated, the average throughput rises to 349 Mbps using Yen’s algorithm. It is also found that Link utilization is elevated for forecasted traffic almost following the same pattern as current traffic. No link failure occurred in topologies T1 and T3. Link 12 in T2 (between S3 and S8) and link 7 in T4 (between switch S1 and S4) failed for forecasted traffic. Topology T5 was most impacted with two failed links (link 6 between switches S1 and S3, and link 7 between switches S1 and S4). It is observed that the link failures in T4 and T5 resulted in an increase in Latency, however, the impact on the overall network is minimal as both T4 and T5 are full-mesh topologies and packets are re-routed through alternative paths during forecasted traffic. We found that in the case of a link failure, the partial mesh topologies had a sharp increase in latency in comparison to full-mesh topologies. As we increased the number of hosts in full-mesh topologies, they became more failure-prone and variation in the end-to-end delay also increased. Scalability in topologies also led to an increase in packet loss. Topology T5 being the largest topology had the maximum packet loss. The maximum packet loss percentage observed is 12%. The highly utilized links directly impact the percent increase in packet loss. In the case of Load Balancing, more the LBP value, more the LBP value, more equally traffic is distributed. It is seen that the multipath routing improved the load balancing with an exception of topology T5 where highly utilized links resulted in a minimal reduction in LBP Value. It is also observed that the floodlight controller witnessed the freezing issues (instability) when the number of switches is increased to 10. So, results for topology T3 may be unreliable as the controller was not able to handle the simulation most of the time. We recommend using up to 5 or 6 switches with the experimental setup deploying floodlight controller.

Chapter 6

6 Conclusion

We developed a network planning framework (NIMA) that is helpful for network planners and ISP in estimating the congestion level of the network and alerts them on the links that are at the high-risk group and indicates the impact on network-wide Quality of Service. We showed that the multi-path routing improved the network load balance. We initialized our experimental setups with current traffic scenarios and evaluated the network performance on the forecasted traffic. QoS parameter evaluation on the forecasted traffic is helpful in identifying high-risk links that need attention. We proposed a link status category based on utilization level and evaluated the change in the status when forecasted traffic is simulated on the network initialized with current traffic parameters. The main contribution of our research is summarized as below:

- To the best of our knowledge, our study is the first to measure and analyze almost all QoS parameters based on forecasted traffic by taking current network scenarios into account.
- We proposed three categories of network utilization to place network links based on the scale of utilization.
- We proposed the formula for Load Balancing Percentage (LBP) and calculated the LBP values for both current and forecasted traffic.
- We developed a general framework that can be used by ISPs and network planners to understand the possible impact of overgrowing traffic on the network in the near future.
- We implemented the multi-path routing algorithm in our experimental setup and found that Yen’s algorithm improved the network load balance.

- Lastly, we compared the Yen’s algorithm with Suurballe’s k-disjoint algorithm. The performance comparison using average latency, average throughput, and load balancing showed that Yen’s algorithm outperformed the Suurballe’s k-disjoint algorithm.

6.1 Limitations

There are some challenges that we encountered while working on this research project. In accordance with the existing studies, we also found that the increasing number of switches made floodlight controller unstable. So, though our results are useful on smaller-scale experiments and they can act as a basis for future studies, there is a scope of improvement by testing a variety of other controllers. Also, the use of a single controller can cause a bottleneck with a sudden increase in the traffic, so our experiments are limited to the amount of traffic our current experimental setup can support.

6.2 Future Work

As already stated in the section above, future experiments can address the identified limitations of our research. At present, most of the available controllers support a limited number of switches in the network simulations. Our study can be extended to mimic the real-work scenarios by testing with a variety of controllers that support a larger number of switches as they become available. Also, the inclusion of multiple controllers will be useful in addressing the traffic bottleneck issues caused because of a single controller. Complete prototype development can also be used as a future plan for the extension of the NIMA framework.

References

[1] Cisco Visual Networking Index: Forecast and Trends 2017- 2022, Cisco Systems, 2019, pp.1-38, [online] Available at https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2] H. Hawilo, M. Jammal, and A. Shami, "Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud," in IEEE Journal on Selected Areas in Communications, vol. 37, no. 3, pp. 643-655, March 2019.

[3] Shami, Xiaofeng Bai, C. M. Assi, and N. Ghani, "Jitter performance in ethernet passive optical networks," in Journal of Lightwave Technology, vol. 23, no. 4, pp. 1745-1753, April 2005.

[4] Chen Dazhi, and Pramod K. Varshney, "QoS Support in Wireless Sensor Networks: A Survey", International Conference on Wireless Networks, 2004.

[5] W. Sugeng, J.E. Istiyanto, K. Mustofa, and A. Ashari, "The Impact of QoS Changes towards Network Performance", International Journal Computer Networks and Communications Security, vol. 3, no. 3, pp. 48-53, February 2015.

[6] D. Mistry, P. Modi, K. Deokule, A. Patel, H. Patki, and O. Abuzaghleh, "Network traffic measurement and analysis", 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, pp. 1-7, 2016.

[7] Raghunath Deshpande,"TCP Proxies Chaining: Performance Implications", 10.13140/RG.2.1.3486.9205, 2015.

[8] A.Forouzan, Behrouz," Data Communications and Networking", (Fourth ed.) Boston: McGraw-Hill. pp. 621–630, 2007.

[9] N. Halim, N. Yaakob, and A. Isa, "Congestion control mechanism for internet-of-things (IoT) paradigm", pp. 337-341, 2016.

[10] S. Santhi, and G. S. Sadhasivam, "Performance Analysis of Different QoS Routing Protocols to Minimize Power in Heterogeneous Network", International Conference on Process Automation, Control and Computing, Coimbatore, pp. 1-6, 2011.

[11] H.N. Jasem, Z.A. Zukarnain, M. Othman, and S. Subramaniam, “Evaluation Study for delay and link utilization with the new-additive increase multiplicative decrease congestion avoidance and control algorithm”, Sci. Res. Essays, 5 (23), pp. 3719-3729, 2010.

[12] H. Dahmouni, A. Girard, M. Ouzineb, and B. Sanso, “The Impact of Jitter on Traffic Flow Optimization in Communication Networks”, IEEE Transactions on Network and Service Management, vol. 9, no. 3, pp. 279-292, 2012.

[13] Y. Al Mtawa, A. Memari, A. Haque, and H. Lutfiyya, “Evaluating QoS in SDN-Based EPC: A Comparative Analysis”, 2019 15th International Wireless Communications & Mobile Computing Conference, Tangier, Morocco, pp. 1279-1286, 2019.

[14] A. A. Neghabi, N. Jafari 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.

[15] Lim Francis, “A review-analysis of network topologies for microenterprises”, Advanced Science and Technology Letters, vol. 135, pp. 175-180, 2016.

[16] T. Takeda, and K. Shiomoto, “Traffic matrix estimation in large-scale IP networks”, 17th IEEE Workshop on Local & Metropolitan Area Networks, Long Branch, NJ, pp.1-6, 2010.

[17] Dimitri Bertsekas, and Robert Gallager, “Data Networks”, Prentice-Hall Inc., Upper Saddle River, NJ, 1987.

[18] Y. Zhang, N. Ansari, M. Wu, and H. Yu, “On Wide Area Network Optimization”, IEEE Communications Surveys & Tutorials, vol. 14, no. 4, pp. 1090-1113, 2012.

[19] Akshay M. Atole, Apurva D. Asmar, Nikhil R. Nemade, Kailash P. Tambe, and Vivek Deshpande, “Performance Analysis of QoS Parameters of WSN by Varying Density of the Network”, Indian Journal of Science and Technology, vol. 9, no. 29, pp. 1-6, 2016.

[20] Seema Shivapur, Suvarna G. Kanakaraddi, and A.K. Chikaraddi, “Load Balancing Techniques in Wireless Sensor Networks: A Comparative Study”,

International Journal of Emerging Technology in Computer Science and Electronics, Vol. 14, No. 2, pp. 21-28, 2015.

[21] Ga-Won Lee, Sung-Young Lee, and Eui-Nam Huh "Congestion Prediction Modeling for Quality of Service Improvement in Wireless Sensor Networks", Sensors, 14, 7857-7880, 2014.

[22] "Mininet: Network Emulator/Simulator", [Online] Available: http://mininet.org/, [Accessed: 30-Mar-2018].

[23] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "Hiqos: an SDN-based multipath QoS solution", China Communications, vol. 12, no. 5, pp. 123– 133, 2015.

[24] Hui Dai, and R. Han, "A node-centric load balancing algorithm for wireless sensor networks", IEEE Global Telecommunications Conference, San Francisco, CA, pp. 548-552 Vol.1, 2003.

[25] R.A. Uthra, and S.V.K. Raja, "Energy efficient congestion control in Wireless Sensor Network", Recent Advances in Intelligent Informatics Advances in Intelligent Systems and Computing, 235, 331-341, 2014.

[26] Najme Tanzade Panah, Reza Javidan, and M.R Kharazmi "A New Predictive Model for Congestion Control in Wireless Sensor Networks", Journal of Engineering Science and Technology, vol.12, no.6, pp.1601-1616, 2017.

[27] H. Sandor, P. Haller, and Z. Gal, "Performance analysis of wireless sensor networks", Procedia Technol., 19, pp. 842-849, 2015.

[28] W. Rafique, and M. Ali Shah, "Performance evaluation of IoT network infrastructure", 22nd International Conference on Automation and Computing, Colchester, pp. 348-353, 2016.

[29] Kherota Yalda, Diyar Jamal Hamad, and Ibrahim Taner Okumus, "Design and Implementation of an Intra-domain routing module for an SDN controller for Traffic Engineering in SDN environment", International Conference on Advances in Software, Control and Mechanical Engineering, Antalya, Turkey, 2015.

[30] A. N. Eghbali, and M. Dehghan, "Load-balancing using multi-path directed diffusion in wireless sensor networks", In Proceedings of the 3rd International Conference on Mobile ad-hoc and sensor networks, pp. 44–55, 2017.

[31] S. Pratheema, K. G. Srinivasagan, and J. Naskath, “Minimizing end-to-end delay using multipath routing in wireless sensor networks”, International Journal of Computer Applications, 21(5): 20–26, 2011.

[32] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, “WCMP: weighted cost multipathing for improved fairness in data centers”, European Conference on Computer Systems, p. 5, 2014.

[33] IP Routing: BGP Configuration Guide, Cisco IOS Release 15M&T, Cisco Systems, 2019, [online] Available at https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/15-mt/irg-15-mt-book.html

[34] Jiayue He, and Jennifer Rexford, “Toward internet-wide multipath routing. Network”, IEEE. 22. 16 - 21. 10.1109/MNET.2008.4476066, 2008.

[35] Petr Lapukhov, and Jeff Tantsura, “ Equal-Cost Multipath Considerations for BGP”, Internet Engineering Task Force, [online] Available at https://tools.ietf.org/id/draft-lapukhov-bgp-ecmp-considerations-02.html

[36] Jin Y. Yen, “An algorithm for finding shortest routes from all source nodes to a given destination in general networks”, Quarterly of Applied Mathematics, 27 (4): 526–530, 1970.

[37] Jin Y. Yen, “Finding the k Shortest Loopless Paths in a Network”, Management Science, 17 (11): 712–716, 1971.

[38] Ricardo Ramalho dos Santos, “Development of an Openflow application for enhanced path”, MSc Thesis, The University of Coimbra, 2015.

[39] Zuhran Khan Khattak, Muhammad Awais, and Adnan Iqbal, “Performance Evaluation of OpenDaylight SDN Controller”, IEEE, 2014.

[40] S. Rowshanrad, V.Abdi, and M. Keshtgari, “Performance evaluation of SDN controllers: Floodlight and OpenDaylight”, IIUM Engineering Journal, vol. 17, no. 2, pp. 47–57, 2016.

[41] C. Fancy, and M. Pushpalatha, “Performance evaluation of SDN controllers POX and floodlight in Mininet emulation environment”, International Conference on Intelligent Sustainable Systems, Palladam, pp. 695-699, 2017.

[42] C. Laissaouui, N. Idboufker, R. Elassali, and K.El Baamrani, “A measurement of the response times of various OpenFlow/SDN controllers with CBench”, IEEE, 2015.

[43] Mittal S., “Performance Evaluation of Openflow SDN Controllers”, Intelligent Systems Design and Applications, Advances in Intelligent Systems and Computing, vol 736. Springer, Cham, 2017.

[44] J.P. Duque, D.D. Beltrán, and G.P. Leguizamon, “OpenDaylight vs. floodlight: Comparative analysis of a load balancing algorithm for software defined networking”, International Journal of Communication Networks and Information Security. 10. 348-357, 2018.

[45] S. Asadollahi, and B. Goswami, “Experimenting with scalability of floodlight controller in software defined networks”, International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques, Mysuru, pp. 288-292, 2017.

[46] A. Brander, and M. Sinclair, “A comparative study of K-shortest path algorithms”, In Proceedings of 11th UK Performance Engineering Workshop, 370-379, 1995.

[47] “Using OpenFlow - Open vSwitch 2.9.90 documentation”, [Online]. Available: http://docs.openvswitch.org/en/latest/faq/openflow/, [Accessed: 30-Mar-2018].

[48] A. J. Pinheiro, E. B. Gondim, and D. R. Campelo, “An efficient architecture for dynamic middlebox policy enforcement in SDN networks,” Comput. Networks, vol. 122, pp. 153–162, Jul. 2017.

[49] “OpenFlow Controller - Project Floodlight.” [Online]. Available: http://www.projectfloodlight.org/floodlight/. [Accessed: 08-May-2018].

Curriculum Vitae

Name:	Tarandeep Kaur Randhawa
Post-secondary Education and Degrees:	The University of Western Ontario London, Ontario, Canada 2017 - 2019 M.Sc. (Computer Science)
	Guru Nanak Dev University Amritsar, Punjab, India 2013 - 2015 M.Tech. (Software System)
	Baba Banda Singh Bahadur Engineering College Fatehgarh Sahib, Punjab, India 2009 - 2013 B.Tech. (Computer Science)
Honors and Awards:	Western Graduate Research Scholarship (WGRS) 2018 - 2019
Related Work Experience	Teaching Assistant (CS-1032 Information Systems and Design) University of Western Ontario 2017 - 2019

Publications:

1. Tarandeep K Randhawa, Anwar Haque, "NIMA (Network Impact Modeling and Analysis): Some studies on evaluating internet traffic impact and network QoS", Manuscript in preparation.
2. Tarandeep Kaur, Amit Chabbra, "Genetic Algorithm Optimized Neural Network for Handwritten Character Recognition", International Journal of Computer Applications 119(24):22-26, June 2015.