2023

Optimizing Flow Routing Using Network Performance Analysis

AL-SAADI, MUNA

https://pearl.plymouth.ac.uk/handle/10026.1/20996

http://dx.doi.org/10.24382/5051
University of Plymouth

UNIVERSITY OF PLYMOUTH

OPTIMIZING FLOW ROUTING USING NETWORK PERFORMANCE ANALYSIS

by

MUNA AL-SAADI

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Engineering, Computing and Mathematics

June 2023

Acknowledgement

First of all, I would like to thank **Allah Almighty** for giving me health and the ability to go on this long and difficult path to the end and complete it with patience and wisdom to finish my PhD and perseverance in completing it satisfactorily. I thank him very much for his countless blessings and without his help; this work would not have been possible.

I would like to extend my heartfelt thanks to Director of Studies **Dr. Asiya Khan** for her prompt and supportive guidance that inspired me with her passion and willingness really to continue with this research. I would also like to extend my sincere thanks to **Dr. Vasilios Kelefouras**, the second supervisor, for his assistance, patience, and support throughout the duration of the study. I must also thank my third supervisor, **Dr. David Walker**, who provided helpful and valuable advice throughout my study.

I owe a debt of gratitude to my beloved sister (**Bushra**) and my brothers (**Deyaa**), (**Taha) and (Asaad)** for their constant love, encouragement, and support. Any success that can be achieved, with optimism, should help me make them proud and happy.

I should not forget to thank **my family** who has been supportive without any hesitation, a big thank to all of them.

My infinite love and appreciation should go to **my best friends** as I wish the potential success of this study would make up for some of what they missed.

I thank my colleagues in the College of Engineering, Computer and Mathematics (College of Science and Engineering) for their encouragement and friendship throughout the duration of the research.

Finally, I would like to express my honest thanks to the Republic of Iraq and in particular, the Ministry of Higher Education and Scientific Research and the University of Information and Communication Technology (UoITC) for granting me a scholarship and sponsoring a PhD study.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the Author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee.

Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

This study was financed with the aid of a scholarship from the Ministry of Higher Education and Scientific Research and the University of Information and Communication Technology (UoITC) in the Republic of Iraq.

Relevant conferences were attended at which work was often presented and several papers were published in the course of this project.

- Muna Al-Saadi, Bogdan V Ghita, Stavros Shiaeles, Panagiotis Sarigiannidis. A novel approach for performance-based clustering and management of network traffic flows, IWCMC, ©2019 IEEE.
- M. Al-Saadi, A. Khan, V. Kelefouras, D. J. Walker, and B. Al-Saadi: Unsupervised Machine Learning-Based Elephant and Mice Flow Identification, Computing Conference 2021.
- M. Al-Saadi, A. Khan, V. Kelefouras, D. J. Walker, and B. Al-Saadi: SDN-Based Routing Framework for Elephant and Mice Flows Using Unsupervised Machine Learning, Network, 3(1), pp.218-238, 2023.

Word count of main body of thesis: 37,561 words

Signed: Muna Yousif Saghir

Date: 14/06/2023

Abstract

Optimizing Flow Routing Using Network Performance Analysis

MUNA AL-SAADI

The main task of a network is to hold and transfer data between its nodes. To achieve this task, the network needs to find the optimal route for data to travel by employing a particular routing system. This system has a specific job that examines each possible path for data and chooses the suitable one and transmit the data packets where it needs to go as fast as possible. In addition, it contributes to enhance the performance of network as optimal routing algorithm helps to run network efficiently. The clear performance advantage that provides by routing procedures is the faster data access. For example, the routing algorithm take a decision that determine the best route based on the location where the data is stored and the destination device that is asking for it. On the other hand, a network can handle many types of traffic simultaneously, but it cannot exceed the bandwidth allowed as the maximum data rate that the network can transmit. However, the overloading problem are real and still exist. To avoid this problem, the network chooses the route based on the available bandwidth space. One serious problem in the network is network link congestion and disparate load caused by elephant flows. Through forwarding elephant flows, network links will be congested with data packets causing transmission collision, congestion network, and delay in transmission. Consequently, there is not enough bandwidth for mice flows, which causes the problem of transmission delay.

Traffic engineering (TE) is a network application that concerns with measuring and managing network traffic and designing feasible routing mechanisms to guide the traffic of the network for improving the utilization of network resources. The main function of traffic engineering is finding an obvious route to achieve the bandwidth requirements of the network consequently optimizing the network performance [1].

Routing optimization has a key role in traffic engineering by finding efficient routes to achieve the desired performance of the network [2]. Furthermore, routing optimization can be considered as one of the primary goals in the field of networks. In particular, this goal is directly related to traffic engineering, as it is based on one particular idea: to achieve that traffic is routed according to accurate traffic requirements [3]. Therefore, we can say that traffic engineering is one of the applications of multiple improvements to routing; routing can also be optimized based on other factors (not just on traffic requirements). In addition, these traffic requirements are variable depending on analyzed dataset that considered if it is data or traffic control. In this regard, the logical central view of the Software Defined Network (SDN) controller facilitates many aspects compared to traditional routing. The main challenge in all network types is performance optimization, but the situation is different in SDN because the technique is changed from distributed approach to a centralized one. The characteristics of SDN such as centralized control and programmability make the possibility of performing not only routing in traditional distributed manner but also routing in centralized manner. The first advantage of centralized routing using SDN is the existence of a path to exchange information between the controller and infrastructure devices. Consequently, the controller has the information for the entire network, flexible routing can be achieved. The second advantage is related to dynamical control of routing due to the capability of each device to change its configuration based on the controller commands [4].

This thesis begins with a wide review of the importance of network performance analysis and its role for understanding network behavior, and how it contributes to improve the performance of the network. Furthermore, it clarifies the existing solutions of network performance optimization using machine learning (ML) techniques in traditional networks and SDN environment. In addition, it highlights recent and ongoing studies of the problem of unfair use of network resources by a particular flow (elephant flow) and the possible solutions to solve this problem.

Existing solutions are predominantly, flow routing-based and do not consider the relationship between network performance analysis and flow characterization and how to take advantage of it to optimize flow routing by finding the convenient path for each type of flow. Therefore, attention is given to find a method that may describe the flow based on network performance analysis and how to utilize this method for managing network performance efficiently and find the possible integration for the traffic controlling in SDN. To this purpose, characteristics of network flows is identified as a mechanism which may give insight into the diversity in flow features based on performance metrics and provide the possibility of traffic engineering enhancement using SDN environment. Two different feature sets with respect to network performance metrics are employed to characterize network traffic. Applying unsupervised machine learning techniques including Principal Component Analysis (PCA) and k-means cluster analysis to derive a traffic performance-based clustering model. Afterward, thresholding-based flow identification paradigm has been built using pre-defined parameters and thresholds. Finally, the resulting data clusters are integrated within a unified SDN architectural solution, which improves network management by finding the best flow routing based on the type of flow, to be evaluated against a number of traffic data sources and different performance experiments. The validation process of the novel framework performance has been done by making a performance comparison between SDN-Ryu controller and the proposed SDN-external application based on three factors: throughput, bandwidth, and data transfer rate by conducting two experiments. Furthermore, the proposed method has been validated by using different Data Centre Network (DCN) topologies to demonstrate the effectiveness of the network traffic management solution. The overall validation metrics shows real gains, the results show that 70% of the time, it has high performance with different flows. The proposed routing SDN traffic-engineering paradigm for a particular flow therefore, dynamically provisions network resources among different flow types.

Contents

List of Figures

List of Tables

Chapter One: Introduction

1.1 Introduction

Network Performance is commonly specified by the speed of the network. It controls the quality of the service (QoS) provided to the user. In order to make an estimation of the network performance, it is essential to analyze its behavior. In that way, the quality of the network link and the data transmission should be determined. For this reason, there is a group of mechanisms pertaining to managing and insuring that a network works at its best [5]. This is named Network Performance Management (NPM). It includes finding a network operations strategy, procedures, and policies to prevent, and resolve network performance problems. The performance management of the network comprises improving the functions of the network by maximizing its capacity, minimizing latency, and raising the reliability of the network regardless of bandwidth availability and appearance of failure [6]. Many functions of network performance management such as traffic measurement, traffic modeling, planning, and network optimization have been used to insure the speed of transit traffic, required capacity for transition, and high reliability that is expected by the network applications. An efficient NPM needs to study the network traffic and choose relevant performance metrics. This can be accomplished by analyzing the performance metrics of each network component [7]such as Packet Loss, Latency, and Throughput. To guarantee a good performance of any type of network, a detailed analysis of the previous parameters is a crucial step.

In view of this, the analysis of network performance is the use of network data to comprehensively understand trends in network performance. A network performance analysis has benefits such as baseline establishment and understanding the limits of the network performance. Further, By conducting the performance analysis, the long-term problems or emerging issues of network performance can be exposed [8]–[10].

The common types of network traffic analysis are traffic characterization and traffic classification. Traffic characterization is related to identifying the network services that are provided to the users such as web browsing, file transfer, sending and receiving an email, etc. While application classification is interested in identifying a specific application such as browser, a chat application, etc. [11].

Two major techniques of traffic characterization have been presented for traffic analysis according to network [12]. The first technique is payload-based which needs information about the packet or flow and the protocols. The second is a feature-based technique, which depends on the patterns of network traffic and statistical analysis of packets. Numerous of the statistical features related to the packets of flow such as the size of the packet, number of packets, inter-arrival time, duration, and so on, are gained by statistical analysis of packets. For traffic characterization, machine learning (ML) techniques are utilized to make use of these features to classify network traffic. Unsupervised machine learning is the common technique that creates models to distinguish between patterns in the data without recourse to use unlabeled training datasets [13].

In datacenter networks (DCNs), which are defined as high-performance networks, the data flows can be categorized into elephant flows and mice flows based on different features. Elephant flow is defined as a flow with a large volume, long duration, and bandwidth-hungry in the network. In contrast, a flow with a small volume, short life, and latency-sensitive is known as mice flow [14]–[16]. Elephant flow exhausts the network resources like bandwidth and buffers, which leads to throughput degradation and delays mice flow. Therefore, the differentiation between elephant and mice flow needs to be taken into consideration to improve the performance of the network [17].

Software-Defined Networking (SDN) as a recent network innovation has contributed to improve the performance significantly. Despite the architecture of SDN introducing more facilities into a network through its characteristics such as centralized controlling and programmability, the above problem still exists [16]. Additionally, the challenges have increased in terms of network management and traffic engineering with the ever increasing in data traffic. Therefore, it is necessary to find an intelligent framework that can propose a strategy for identifying and forwarding different types of flow [18].

Traffic engineering (TE) is the process that is associated with network traffic measurement and management. Its main purpose is to design the logical mechanism for directing traffic in a network, which can be used for improving the employment of network resources and providing network quality of services (QoS) requirements [19]. In other terms, the TE aims to optimize the utilization of resources and improve network performance by diminishing congestion, energy consumption, latency or delay, and packet loss [20][21]. Multipath routing is a TE approach that deals with large flows by dividing the load over the available paths. However, the multipath solutions do not distinguish the flow as elephant or mice and do not take into consideration the delay of routes [19][22]. Therefore, an efficient TE multipath routing solution according to flow types and delay of the route needs to be found.

The rest of the chapter is organized as follows. Section 1.2 presents the motivations behind this project. Section 1.3 highlights the research aims and objectives. The summary of this thesis is shown in Section 1.4.

1.2 Motivations

Nowadays, the need to find efficient network management has become imperative because of the increasing complexity in the structure of the network. One of the essential areas of network management is performance management. It provides a comprehensive vision and actionable perception and allows proactive handling of network performance issues. Consequently, analyzing and evaluating the performance metrics of network infrastructure is a requirement of network performance management. Moreover, the analysis of network performance is an important aspect to understand the behavior of the network, where network data can be used to define the network performance tendency. Traffic modeling has an essential role in the network analysis phase and an accurate traffic modeling enhances the understanding of complex network behavior and its properties [23]. Furthermore, in order to build an optimal model of network performance, characterizing network traffic accurately is a critical issue. This is because there is a possibility to identify traffic wrongly or the estimation of network performance was unsound [24]. On the other hand, the essential portion in the network management, which has a real effect on the network performance, is flow routing. The main purpose of flow routing is to access required data as fast as possible, which is the obvious advantage that the routing process can provide to improve the performance of a network [2][3]. In DCN, inefficient routing of flows elephants and mice can cause degradation of the performance of the network, where the imbalance in network load leads to network congestion [25]–[28]. Therefore a differentiation between elephant and mice flow has an important role to improve network performance [23], [29]–[31]. Accordingly, an effective approach that successfully performs characterizing, identifying, and routing of the flows need to be designed. This project includes building a framework that consists of three models: The network performance parameters-based characterization model that will be built leveraging unsupervised machine learning (ML) algorithms, thresholding-based elephants and mice flow identification model, and the developed Dijkstra algorithms-based routing model. To promote the proposed method, SDN environment has been used for the implementation of this project[32]–[34].

1.3 Research Aim and Objectives

In this research, the aim is to investigate and evaluate the application of unsupervised ML in network traffic management and routing. Furthermore, the research aims to develop a unified architectural flow routing solution that integrates characterization and identification of flows with SDN to enhance the system of network management by improving bandwidth utilization and reduce congestion across the network. To achieve this aim, the research work has the objectives as follows:

1. Undertake a fundamental review of the state-of-the-art in network performance management and network performance analysis.
2. Investigate the procedures of traffic characterization and classification, identification and routing of elephants and mice flow.
3. Propose a method for characterizing network traffic flows based on the performance metrics of the network using unsupervised machine learning techniques.
4. Design an automated method for identifying the characterized flows as elephants and mice according to pre-decided thresholds using the thresholding technique.
5. Design and develop a unified architectural framework to improve network management by integrating unsupervised machine learning with the SDN.
6. Evaluate the effectiveness of the proposed framework by (i) Different data sources of traffic. (ii) A comparison between the performance of the proposed application and the SDN-Ryu controller.

1.4 Thesis Organization

This thesis is organized as follows. Chapter 2 shows a comprehensive review of the state of the art in network performance analysis, TE, and SDN. Chapter 3 presents the experimental study of network performance analysis and traffic clustering. While Chapter 4 introduces traffic analysis-based flow identification. The proposed traffic management framework based on SDN is discussed in Chapter 5. Conclusions, achievements, and limitations of the project will be clarified in Chapter 6. A brief description of each chapter is summarized below.

Chapter 2 gives a background about network performance and all the issues that affect the network performance level. In addition, the analysis of network performance and approaches of traffic analysis were explained to determine the appropriate characterization method for conducting the classification. Reviewing of challenges and solutions for network performance was briefly introduced to find useful information that would help to make the analysis process sufficient and to avoid the uselessness of data. On the other hand, network traffic classification was summarized to identify its general goal and to know how it is essential for QoS control. Traffic engineering development was presented to explain the role of this application to improve network resources employment and provide the requirements of network quality of services (QoS). A brief background of SDN was shown to clarify the impact of SDN deployment on the network architecture's development. A literature review of flow Analysis investigation, elephant and Mice flow identification, and SDN- based flow routing optimization were placed in this chapter.

In Chapter 3, a scheme of unsupervised machine learning has been proposed to investigate into network performance analysis. The k-means clustering algorithm has been utilized for two purposes: the selection of features and the clustering process. Two feature sets have been used to clarify the impact of the optimal selection of feature sets on the clustering process. To improve the set of parameters and reduce the dimension of the dataset as well as for handling the outliers and the problem of variables correlation, the PCA analysis technique was applied.

Chapter 4 shows an innovative model design of network performance features-based flow characterization and then identifies these flows as mice or elephants based on these features. The model also proved the possibility of clustering network traffic based on their performance attributes. Consequently, each cluster contains identical flows regarding throughput, RTT, packet loss, etc. On the other hand, identifying the flows as long-lived (elephant) and short-lived (mice), which comprises

90% of traffic on the network, is still challenging. As a result, the two identified types of flows, which were characterized with respect to performance features, can be utilized for optimizing systems of network management in the future and improving the performance of networks by finding the best path for each flow. This in turn will contribute to improving the QoS too. The scheme of identifying flows will be applied as a primary stage in the next part of this project.

Chapter 5 introduces a novel framework for SDN- based flow routing. The framework introduces a developed Dijkstra algorithm that selects the best route based on flow type (elephant or mice) and takes advantage of the programmability offered by SDN/OpenFlow. A recursive update of route costs is performed, which provides more efficiency and enables consistency with real-time constraints. Consequently, an efficient and more balanced network will be gained, and higher throughput will be obtained.

Chapter 6 introduces the research conclusions, achievements, and limitations of the project. Furthermore, future work is suggested in this chapter.

Chapter Two: Network Performance and Traffic Engineering

2.1 Introduction

The network performance aims to define the service quality offered by the underlying network. One of the significant issues in networking field is network performance monitoring. It is both a quantitative and qualitative process, and it defines and measures the performance level. This process guides the administrators of the network to measure and improve the network services. Furthermore, they need to assess the performance of the network regularly to guarantee there are no overloaded devices in the network, and they are capable to leverage monitoring Measurements prior to congestion[35]. In other terms, the process of evaluating, analysing, reporting, and tracking the performance of a network is known as network monitoring. Its objective is to enable those concerned to follow the overall performance and service quality of the underlying network through the analysis and review of collective network statistics.

The performance of the network is measured by statistics and metrics drawn from specific network components, namely:

- Network bandwidth or capacity: The network applies bandwidth meters to limit the maximum Available data transfer to certain flows.
- Throughput: The amount of data that is successfully transmitted over the network in a given time.
- Latency: measures the time it takes for data to reach its destination and ultimately make a roundtrip.
- Packet loos: happens when one or more transmitted data packets fail to reach at their intended destination.

Network technologies and computer systems have undergone rapid development in the last few decades, therefore, the need to monitor, evaluate and control network performance precisely has become essential and urgent [36].

Consequently, understanding the performance of the network is vital for those concerned with networking because it helps to develop complex operations and protocols of a network [37]. Furthermore, it is important to realize that the best network performance is achieved through acquiring all the information with respect to the behaviour of the network. Also, precise information is necessary for the assessment, troubleshooting, and enhancement of the network to attain the required performance level[38][39]. Beneficiaries of networks, whether they are users or engineers, have tried to understand the metrics related to network performance in order to determine the reasons for performance retraction for optimization [40]. Because, the metrics of network performance depend on the network's conditions [41].

Network traffic is also called data traffic, which can be defined as the amount of data moving across devices at a given time. In addition, it is fragmented into data packets and sent through a network before being regathered by the receiving device. Moreover, the characteristics of network traffic require a deep understanding to obtain a comprehensive view of the possibilities for evaluation. It is important to specify the requirements for the enhancement of network functionality and infrastructure development [42]. Thus, it is necessary to discover the factors that define network performance [36][41]. Network monitoring and management function depends on an accurate characterization of network traffic, which is produced by diverse applications and network protocols [43].

Accurate characterization of network traffic is significant for building optimal models of network performance, and also for avoiding the possibility of inaccurate traffic or wrong estimation of network performance [24]. Moreover, efficient management of the network resources can be achieved by precise traffic analysis and characterization [44]. Traffic classification is one of the approaches in traffic analysis that aims to classify network traffic into predefined groups [25]. It is noteworthy that a wide variety of machine learning methods are used to classify network traffic, using statistical analysis. Different statistical features, regarding the flow (packets) like the size of the packet, number of packets, inter-arrival time, duration, etc. are used by ML [45][46].

From the aforementioned, it is clear that the use of traffic analysis seems to be more efficient for network management. Using performance metrics can provide an accurate characterization of flow with perfect identification for flow types providing a more effective framework for flow routing. Therefore, as, to introduce an insight into network performance management and network management, this chapter explains the network performance analysis, flow characterization, identification of flow types, traffic engineering, and SDN environment.

The rest of this chapter is organized as follows: Section 2.2 gives an overview of network traffic analysis. Section 2.3 summarizes ML classification techniques and the latest developed learning techniques, while the employment of these techniques is presented in Section 2.4. Section 2.5 shows the development of TE. The background of SDN is clarified in Section 2.6. Section 2.7 highlight a comprehensive literature review of flow analysis research, modern studies for elephant and mice identification, and the proposed mechanisms for flow routing optimization in the SDN environment. Conclusions were drawn in Section 2.8

2.2 Network Traffic Analysis

Traffic analysis is the vital procedure that involves monitoring the network activities, revealing particular patterns, and gathering useful information from network data. One of the factors that has an important role in the stage of analysis is traffic modelling; accurate modelling of traffic enhances the understanding of sophisticated network behaviour and its features [37].

There are many types of tools for traffic modelling, such as analytic, testbed, simulation, and operational analysis. Measurement and estimation of the real system can also be provided by operational analysis. Analytic modelling tools have the capacity to describe a model in terms of mathematical expressions such as using Analytic modelling for decomposing the queuing system. Testbed is a platform created for analysing the behaviour of the system under different conditions to get a comprehensive image of it, for example, a testbed has been used to compare three different control models for SDN routing [47].

The simulation tool is used to test the system without any effect on the real system employing programming language for instance the simulation model was developed to evaluate the initial experiments for transmitting messages, and those that resulted from the reduction in quality of service [48].

Many network performance modelling and estimation tools have been provided, but operational analysis can give a comprehensive overview of actual performance, as well as a prediction of performance patterns [49]. This kind of tool can be used when the monitoring of network performance is enabled by compatible hardware and software equipment, which introduces substantial experimental outcomes. Consequently, these tools provide the ability to get the information from the real system whereas analysis of this information can give good insights into the future of network behaviour. In addition, the results of the analysis play a significant role in improving network performance [50].

The solutions to network performance problems require improved, effective monitoring techniques and measurement of network performance which will be used in operation and communication of network management [37][51][52]. Management of the Network depends on network traffic analysis, which consequently depends on monitoring and collected information, to take the required action on the network. This action is taken according to measurements and metrics of network [53].

Measuring the parameters of network performance, which are packet loss, delay, and the error rate, can be achieved by using two techniques, active or passive, which provide a possibility to define the distribution of whole end-to-end losses and delays between network parts as in Table 2.1. In addition, to packet losses and retransmissions, there is an important performance parameter in client-server communication, called the server response time. All of these parameters affect network services, such as emails, transfer data files, web pages, and requests for domain names from their servers [37]. The availability, loss and error, delay, and bandwidth are four groups represented by the metrics that have the greatest effect on the network performance, and are advantageous for estimating the level of service offered for IP flow forwarded through the network [41]. The availability metrics are used to estimate the amount of time that the network is operating without any failure. Loss and error metrics measure the packet loss fraction, which is caused by the overflow of buffer in a network, or other reasons, or bits error fraction, or packets. In addition, they indicate the conditions of network congestion and/or errors of transmission and/or malfunction of equipment. Delay metrics have been used to define the conditions of network congestion or the changed routing effectiveness.

The delay of the packets transferred by a network can be measured in two ways: either a one-way delay or a round-trip time delay (RTT). Jitter, which is a variation of delay, can be measured by these metrics. Finally, the metrics are used to measure the amount of data that can be transferred through the network, in a unit of time called bandwidth metrics. In addition to the network performance metrics mentioned above, Central Processing Unit (CPU) load, memory depletion, and hardware temperature in the network are often useful metrics that can explain the reasons for the degradation of network performance. The process of monitoring may perceive these metrics and register them as being significant to reveal the degradation of service levels by their values or may prevent degradation by the improvement of equipment. Furthermore, these metrics can represent the monitoring of flow and routing groups [35].

In recent years, the growing number of users of the network has increased the need to convert large files. The use of real video and the demand for multimedia applications has also increased. All these requirements have led to a need for increased network productivity, which in turn has increased network congestion. Slow response time and slow file transfer are the results of congestion, and as a result, the productivity of the network becomes lower for users. This problem can be solved by efficient use of the bandwidth that is provided, or by increasing the network bandwidth [36]. In general, the characterization of traffic is the first significant step towards understanding and solving the problems related to network performance.

Table 2.1 Comparison between Active and Passive Measurement

Monitoring method	**Description**	**Techniques**	**Resources Requirements**
Active	- Used to analyses network performance in particular aspects. - Employed to calculate and present a description of network such as HTTP response time, jitter, packet loss, etc.	File Transfer Packet Pair Packet Train	CPU - Low Memory – small data
Passive	- Produce a comprehensive view of the performance of the network. - Employed to determine the network component with the highest consumption of bandwidth.	Flow level Packet level	CPU - high Memory- large Data

2.2.1 **Network traffic Characterization**

This section clarifies two distinct methods for the characterization of network traffic.

2.2.1.1 *Packet level Characterization*

In packet-level characterizations, traffic flows can be expressed in terms of inter-packet time and packet size. This level of characterization can be used for analysing network traffic. The main advantage of applying packet-level traffic characterizations is to generate and simulate traffic that allows to measure and study the parameters of the network [54]. Some of the problems can take advantage of a finer granularity that is provided by packet-level. This method of analysis is characterized by flexibility and conciseness. All the packet-level methods for modelling and predicting are seeking to provide a mechanism that has the ability to fast response to any rapid change in network conditions in a real-time. Indeed, packet-level analysis can provide precise traffic characteristics, which might be not provided by executing analysis in aggregated coarser-grain [55]. For example, packet-level characterization can be used in the analysis of prediction to assess the peculiarities of network traffic produced by mobile apps [56]. While the traffic characteristics of network devices can be analyzed based on packet-level characterization in order to give precise insights into the characteristics of the operations and behavior of these devices. This can provide efficient security and performance mechanisms for the network [57][58].

2.2.1.2 *Flow level Characterization*

The evolution of the network is related to a comprehensive knowledge and traffic characteristics understanding, which indicate the kind of deployed techniques in order to perfectly match the requirements of the user and the constraints of the network. Consequently, the development of monitoring-based tools, improvement of technologies to collect network traffic information, and analysis of their characteristics are currently essential topics for network engineering and research.

Flow level characterization is the second most popular approach to modelling traffic. In this context, traffic flow is "a sequence of packets sent from a particular source to a particular unicast, any cast, or multicast destination that the source desires to label as a flow" as defined in RFC 3697. A further definition of a flow is given by [59][60] who describes it as “a uni-directional traffic stream with a unique <source-IP-address, source port, destination-IP-address, destination port, IP-protocol> tuple”.

A flow record contains packet-level characteristics, which are found in the header of every packet of the flow, and flow-level characteristics, which are calculated depending on the collected values from all the packets in a flow. Average packet size, flow data rate, and flow duration are good illustrations of flow-level characteristics[61]. In contrast to packet-level data, flow-level data is used for different network management functions. The enormous amount of flow data leads to defying the scalability problems in the classification models that are used for network traffic [62]. In recent years, state-of-the-art techniques use the flow statistical features. Therefore, a valid selection of flow-traffic features is substantial when applying flow-level analysis, which optimizes the traffic behavior representation. On the other hand, the right selection of flow-level features is necessary when data reduction is needed [63]. Moreover, by analyzing flow level statistics, the classification of network traffic can be conducted [13][29][62].

2.2.2 **Network Traffic Classification**

The classification of network traffic has a general goal, which is network performance improvement. Accurate classification of traffic is becoming increasingly essential with many applications in network such as TE, security, and QoS. The approaches for traffic classification proposed in both the academic literature and in the practical field include:

2.2.2.1 *Port-based methods*

According to this method, the initial criteria to classify traffic were transport protocol ports. Applications like P2P, for example, can use the ports of well-known protocols to hide themselves dynamically. Similarly, some protocols, such as FTP protocol, can assign ports depending on the load of traffic. This is considered to be one of the disadvantages of the port-based method. The method is described as rapid and utilizing low consumption of resources. It is supported by numerous network devices and does not use the payload of the application layer, so it has no impact on the users' privacy [42]. It is used by applications and services that employ fixed port numbers. Since the port number in the system can be changed, therefore, it facilitates fraud [64]. Moreover, this method has few restrictions. Firstly, applications for ports, for example, p2p applications, may not be registered with the Internet Assigned Numbers Authority (IANA). An application may use well-known ports to avoid access control restrictions of the operating system. In addition, some server ports are allocated dynamically, as desirable. For example, a Real Video broadcast device allows dynamic negotiation of the server port used to transfer data. This server port is negotiated on the initial TCP connection, which was created using the known Real Video control port. It has been found that the accuracy of port-based classification using the official IANA list was no better than 70% [65], or 30% to 70% [66] or only 30% of the total traffic [67]. In some cases, IP layer encryption may cause TCP or UDP header confusion, which makes it impossible to know the physical port numbers.

2.2.2.2 *Payload Inspection-based Methods*

To evade total dependence on port number indications, many existing industry products use formal session reconstruction and application information from the content of each packet. At present, the most common methods of traffic classification are based on payload inspection, owing to its comparatively high precision. Because of the use of deep inspection of packets, violation of the privacy of user data and the amount of data that is processed are the main drawbacks of this approach. Moreover, the classification based on this method cannot be applied to encrypted data [42]. Although it avoids dependency on fixed port numbers, payload inspection [40] imposes significant complications and processing load on the traffic-identification device. Extensive up to date knowledge of application protocol semantics is necessary for the payload-based inspection. In addition, it must have sufficient power to perform the analysis of a potentially large number of flows concurrently. Encrypted traffic and private protocols have made the aim of this method difficult or impossible to achieve. Furthermore, because its inspection depends on the content of the application layer, the method may infringe on the relevant privacy policies. For example, examining the factual packet payload, using a deep packet inspection, detect the services and applications in any case of port number. Therefore, this method lacks support for several applications, such as Skype. In addition, it is lazy, needs a lot of power for processing, and signatures must be up to date.

2.2.2.3 *Flow statistics-based methods*

Current modern methods tend to classify network traffic on the basis of the statistical characteristics of the flow. They perform the classification by analyzing the level of statistical flow. Statistical classification-based methods depend on analysis of characteristics such as frequencies of byte, packet inter-arrival times, and size of the packet. An alternative to payload inspection methods, the classification of traffic based on flow traces could be used [42]. In these techniques, statistical features of network flows are collected from the headers of packets. The underlying assumption of such approaches is that in the network layer, the traffic has statistical features (e.g., duration of flow, idle time of flow, packet inter-arrival time, and length of packet). These features are unique to particular classes of applications and enable the use of the diverse source applications to be distinct from each other. In [68] and [69], a relation between the traffic classes and their observed statistical characteristics was noticed, where the experimental models of connection properties, such as duration, bytes, and arrival time for a number of particular TCP applications were built and analyzed [68]–[70]. In this work, flow statistics-based techniques will be used because the art of states proves that this method manipulates all the problems of previous methods, which are port-based and payload-based.

2.3 ML Classification Techniques

Machine learning (ML) is known historically as a set of powerful techniques for data extraction and the discovery of knowledge. Machine learning is the capability of a machine to learn automatically from experiments, to edify, and to perfect its base of knowledge [71]. Learning refers to the adaptability of the system to changes, whereby it can perform the same tasks more efficiently and effectively the next time [70][72]. There is a wide range of ML applications, such as, medical diagnosis, text and handwriting recognition, search engines, and image screening. In 1990, the control of network traffic using ML techniques was proposed, with the aim of maximizing the completion of the call on a communications network with a switch circuit [70]. This was one of the highlights of the expansion of ML in its communications network applications. In addition, in 1994 ML was first used to classify internet flow in the context of intrusion detection [70]. On the other hand, the reasons behind the use of ML technologies in this area are imperative to transact with the types of traffic, different datasets, and the flow, which is the multidimensional spaces and properties of packets. There are two learning techniques in machine learning, supervised and unsupervised. The techniques affect the data collection, attributes engineering, and creating ground truth.

2.3.1 **Supervised ML Classification Technique**

Supervised ML is used to “learn” to identify behavior in the “known” training dataset. This method is applied to overcome the problems of classification and regression that are related to predicting continuous or discrete outcomes. It ensures highly precise results in trained applications [43]. The supervised learning technique concentrates on modelling the relationships between input and output. Its objective is to define a mapping from input parameters to output classes. The knowledge learnt can be displayed as a decision tree, classification rules, etc., that will be then utilized to classify unknown instances. This technique creates models by using labelled data. In supervised learning, there are two major stages (steps):

• Training: It is called the learning stage that examines the input data and creates or builds a model for classification.

• Testing (also known as classifying): It is called the classifying stage that uses the model, which has been created in the training stage, to classify new instances.

2.3.2 **Unsupervised ML Classification Techniques**

Unsupervised learning creates models that can distinguish between patterns in the data without recourse to use labelled training datasets [13]. Clustering defines an unsupervised technique of machines learning which leads to portioning a given dataset into meaningful subclasses so that members in a subclass are similar to one another and are different from the members in other subclasses. These subclasses form clusters. There are two basic clustering methods: the classic Partition clustering such as K-means algorithm; and hierarchical clustering that builds the clusters by recursively partitioning the instances. K-Means is one of the widespread simplest and fast "clustering" approaches that store particular pre-elected k centers. It is utilized to create clusters randomly based on the similarity between all input members [73].

2.3.3 **Deep learning Techniques**

Deep learning (DL) is a group of techniques that merge between machine learning (ML) and artificial intelligence (AI)[74]. Due to its capability to learn is become a popular topic in the networking context is widely utilized in different applications such as intrusion detection, traffic analysis, and classification, etc. [10][75][76]. To build a DL model, the same processing strategy of ML will be followed such as understanding and preprocessing data, building a model, training, and validating machine learning will be followed. However, designing an efficient model of DL is a challenge because of that the problems and data are variate and have dynamic nature in a real-world [77]. Based on the literature, deep learning is also defined as "deep structured learning", "hierarchical learning", "deep feature learning", and "deep representation learning" [78]. Accordingly, the deep architectures can be categorized based on their use into three types, called, "generative", "discriminative", and "hybrid deep architectures". The work concept of generative deep structures is high-order correlation properties characterization of the input data for compilation, while a discriminative architecture aims to classify patterns. Compared with the previous two types, hybrid structures works on carrying out discrimination processes by the optimized outcomes obtained from the generative structure [79].

Recently, the availability of data on traffic flows and their related labels has made the classification problem of protocol efficiently solved by deep learning models such as deep neural networks [80] and stacked autoencoders [81]. Moreover, in [82][83] and [84] that the performance of stacked autoencoders is better compared with the deep neural networks for classifying any traffic data to a pre-identified protocol. In addition, using of deep learning approaches was presented to detect anomalous and suspicious flows [85]–[87]. On the other hand, deep learning frameworks were proposed by many studies in network flow prediction, for example, aggregating multiple attributes to predict the flow of the network [88][89].

2.4 ML techniques Employments

ML approaches introduce a better strategy for network traffic characterization and classification by leveraging traffic statistical information that is independent of traffic payload. Compared to the aforementioned traditional classification methods, ML has solved the known problems of DPI and port classification for flows [90]. They have been utilized for traffic classification, as various algorithmic steps can be used to build a classifier that gathers data observations into distinct classes according to the network field, such as network management [91]and security [92].

2.4.1 **Application-based ML Classification Techniques**

Nowadays, classification approaches have an effective role in different fields of networks. The efforts of researchers focus on using ML techniques for traffic classification according to flow statistics. A powerful classification paradigm is constructed by selecting suitable features and the correlated information of TCP flows. This paradigm was proposed to enhance the classification performance using Correlation (TCC) information [93]. For identifying applications, a system using a group of seven different applications has been proposed in order to execute a set of tasks to measure the QoS of the network [94]. In addition, to cluster traffic of applications based on traffic similarity, two unsupervised ML approaches have been presented. In the proposed framework, the appropriate attributes of flow were elected using the filter method before applying K-means and Expectation Maximization (EM) [73]. On the other hand, for identifying unknown applications according to statistical features of the flow, the proposed unsupervised ML classification method is integrated with the clustering approach to create clusters of traffic[95]. Similarly, a developed semi-supervised clustering technique was suggested. A modified K-means that used flow statistics to classify network traffic was applied. The statistics of layer four of the network were input to execute the classification steps [96].

2.4.2 **User-based ML Classification Techniques**

In the last decade, internet users' activities have increased, which has had an impact on the behavior of the users themselves. For this reason, classification and pattern extraction from the data of users is very important for business support and decision-making. Many studies have used classification techniques (clustering) to minimize the gaps in studying network users' behavior patterns, They have characterized patterns of individual users' behavior, and then the clusters of users have been used to develop prediction methods of network traffic [94].

Data mining with the K-Means technique has also been used as another way to analyze users' behavior. Data mining-based k-means clustering technique was used with log activity, which is another means of studying and analyzing the behavior of users [97]. On the other hand, lack of knowledge has become a problem for most large companies, which have a huge amount of data but suffer from a knowledge famine. Therefore, they are in need of a new technique that is intelligent and has the capability to overcome this flaw. Accordingly, "data mining technique - customer clustering" has been used to characterize "high-profit", "high-value" and "low-risk" customers [98]. One of the recent challenges in online services is understanding user behavior because it is increasingly dependent on the participation of users in online social networks or crowdsourcing services. Therefore, to identify dominant user behavior by using "clickstream" data, an unsupervised technique-based system has been built, where the resultant similarity clusters are described through graphs, which are partitioned on the basis of similarity [99]. Furthermore, one of the big challenges of the present-day internet is the rapid growth of this field, which is making it a robust system to spread and restore information. Restoring useful information from this huge amount of information is, indeed, difficult [100]. The objective is to take advantage of the similarity-based grouping property of the clustering method to match the sessions of web users.

2.4.3 **Security-based ML Classification Techniques**

Network security is an area that has received a great deal of attention over the years and intrusion detection is a major field for most researchers. A new method to detect anomaly intrusion with a high detection rate by using a clustering technique with specific modifications has been innovative [101]. For the same purpose, which is to modulate the rate of anomaly detection, the clustering technique was used for data pre-processing in anomaly detection [102]. On the other hand, in order to reduce time consumption and raise the detection rate, the Artificial Neural Network (ANN) method, which has high computational resources, has been combined with a fuzzy clustering approach [103]. Analyzing raw data is a key stage in network security for identifying the causes of damage or loss; the method used for the attack; and the identity of the attacker. Clustering is considered the most powerful technique for this purpose: it can help in the detection of intrusion when the training data is unlabeled as well to detect unknown types of anomalies. Accordingly, the Simple K-Means, clustering method was used to analyze the NSL-KDD dataset, by clustering it into normal and the four major types of attack, i.e., DoS, Probe, R2L, and U2R. It is crucial to distinguish the process of attack from "benign" traffic because its role in identifying the types of attack leads to the specification of the required preventive measures [104].

2.5 Traffic Engineering Development and SDN

One of the significant network applications is traffic engineering (TE), which interacts with the measurement and management of network traffic. In addition, it designs logical mechanisms to direct network traffic to improve network resources employment and provide the requirements of network QoS; this is the generic definition of TE. The technologies of TE mainly include Internet Protocol (IP)-based TE and Multi-Protocol Label Switching (MPLS)-based TE. To solve the load-balancing problem in multipath traffic, the IP-based TE has been proposed. The problem has been solved by optimizing the algorithm of IP routing to avoid congestion of the network [105]. However, this technology has two disadvantages: first, the inability to make full use of network resources when Open Short Path First (OSPF) link weights are used to control the routing of a network. Second, network congestions, packet losses, delays, and even routing loops occur whenever links fail, or link weight changes in the topology of the network because OSPF protocol takes time to meet the new network topology. The MPLS has been proposed to overcome the drawbacks of IP-based TE [106]. Nevertheless, the complexity of the mechanism of MPLS protocol can lead to a high overhead of network performance, therefore, the requirements of data center of networks, which need to have high link bandwidth utilization, green energy saving, and high reliability, is difficult to meet. In traditional networks, control management and data forwarding are tightly coupled, where distributed devices control the whole network, and it is hard to improve the flexibility and extensibility of it. Therefore, it is imperative to evolve the architecture of the network and corresponding TE technology to solve this problem. Software-Defined Network (SDN) is an advanced architecture of networks, which was proposed by researchers at Stanford University, and it has gained widespread attention in recent years. Its main idea is based on the separation of the forwarding and control planes of a network system. In addition, it has a programmability attribute that can be used to improve the innovation capability of network applications significantly [106][107]. Figure 2.1 shows the development of TE from the past to SDN as a future solution to the above problems. The definitions of essential parts of SDN and important concepts will be discussed in the following section.

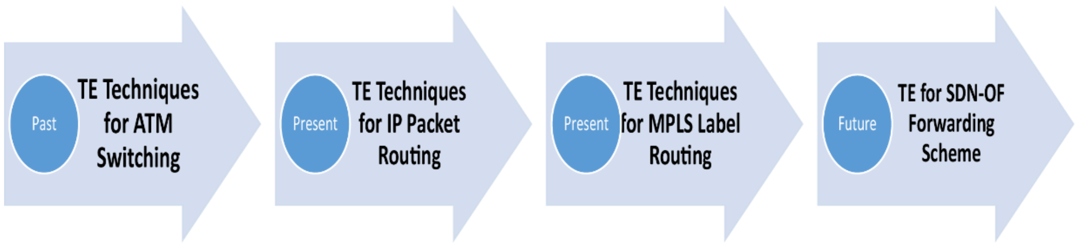

Past
TE Techniques for ATM Switching
Present
TE Techniques for IP Packet Routing
Present
TE Techniques for MPLS Label Routing
Future
TE for SDN-OF Forwarding Scheme

Figure 2.1 Traffic Engineering From Past To Future [Modified From[108]**]**

2.6 Software-Defined Networking (SDN)

SDN can be defined in general as a novel paradigm of networking that optimizes the traditional network environment through the separation of the forwarding plane and control plane [109]–[113]. In other words, SDN is an architectural approach for managing networks by providing dynamic and programmatic effective network configuring, which makes the network more intelligent and centrally controlled, to enhance the performance of networks using software applications [114][115].

The generic SDN architecture consists of two components, which are the controller and compatible switches. Additionally, SDN works according to the concept of decoupling the network devices' data plane from the control plane [116]. In specific, routers, and switches as infrastructure plane has a responsibility to forward a packet whereas the control plane has the rules for forwarding packets by the devices in the infrastructure plane. Based on the above, SDN is defined by decoupling the controller plane and infrastructure plane programmability [117]. Increased SDN employment has led to a significant evolution of today's network architectures by offering adaptability, flexibility, and scalability. In addition, SDN has the capabilities to control and efficiently transfer the data flows through the network to fulfill sufficient flow management and effective usage of network resources. It does so by separating the network control logic from the underlying switches and routers, providing logical centralization of network control, and allowing the programming of the network. In the last years, most studies have suggested integrating the ML and SDN for evolving network security, network management, and improving the design of a network [118]–[120].

2.6.1 **SDN Architecture**

Nowadays SDN is being used in numerous fields e.g., IOT, wireless networking, cloud computing, security, etc., so architecture varies with the area of usage but the basic architecture of SDN remains the same. Figure 2.2 illustrates the standard SDN Architecture. The typical structure of SDN divides operations such as configuration, allocation of resources, traffic prioritization, and traffic redirection through core devices into three layers: application, control, and infrastructure schemes.

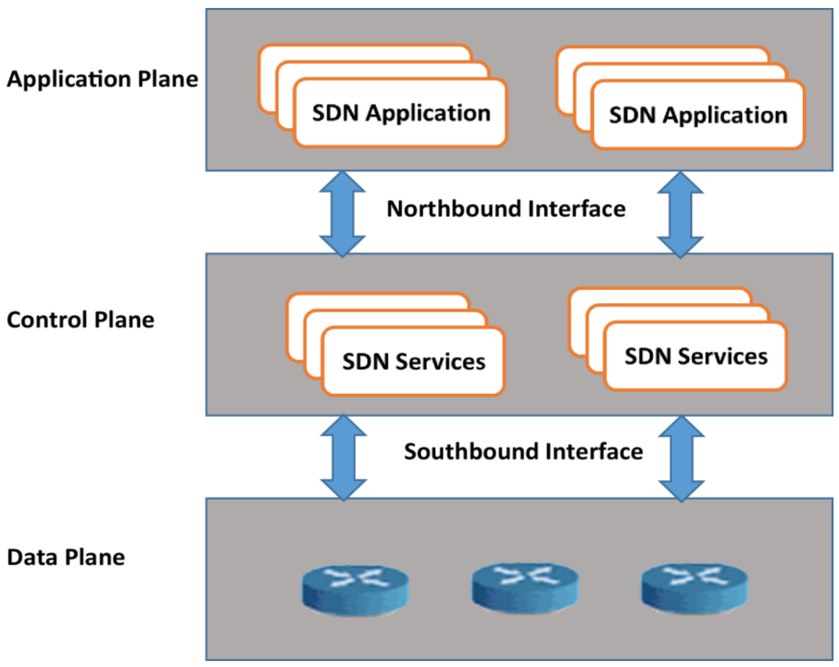


Figure 2.2 Simplified SDN Architecture [Modified From [121]**]**

The bridge that connects the application layer and infrastructure layers is a control scheme with two interfaces (i.e. south and north-bound interfaces). Each one of these interfaces has a specific function. South-bound interface represents the downward interaction with the infrastructure layer, which has limited function that gives controllers the ability to access functions that are provided by switching devices. For application layer, it connects with the controllers through the north-bound interface, which is responsible for providing access points in different forms to the services such as an Application Programming Interface (API). Through API, the SDN application can get all the status information of the network that is provided by switching devices. Moreover, network status information has been used to make decisions for setting rules to execute the forwarding of packets (switches devices) by application layers using API. Since there are many controllers, therefore, it is obligatory to coordinate the process of decision-making between them by using an "east-west" communication interface, which lies among the controllers [117]. Furthermore, SDN has an option of controlling called OpenFlow (OF) protocol, which is the predominant protocol. The architecture of OF protocol is based on three basic ideas [122]:

1. SDN data plane consists of switches, which are compliant with OF protocol.
2. One or more OF controllers are composed of the SDN control plane.
3. Provisioning of a confidential control channel to connect the switches with the control panel.

2.6.2 **Why SDN?**

Compared to the architecture of the traditional network, SDN has the following distinguishing attributes:

1. Concentricity of control: the entire network information such as the topology of the network, changes in network status, and requirements of application such as requirements of security and QoS, are stored by the SDN controller.
2. Programmability: the devices of the data-forwarding layer can be programmed dynamically to optimize the network resources distribution.
3. Openness: to communicate with the SDN controller, as the controller does not depend on the various supplier's devices, therefore, a unified interface is used by forwarding devices. Furthermore, the SDN controller can gain the status information of the network conveniently for network traffic scheduling.

In addition, SDN has characteristics that are useful to solve the current problems of network traffic engineering. These characteristics can be summed up as follows [123]:

1. Traffic measurement: the SDN, which can gather information on real-time network status and monitor as well as analyzes traffic centrally in the controller, has the flexibility of scalable measurement task deployment.

2. Scheduling and management of traffic: Globally, the requirements of traffic application can be considered, hence, the possibility of scheduling with flexibility and granularity can be achieved.
3. Multiple stream table pipelines of the OpenFlow switch provide more flexibility and efficiency for flow management.

2.7 Literature Review

The major task of a network is to hold and transfer data between its nodes. To achieve this task, the network needs to find the optimal route for data to travel by employing a particular routing system. This system has a specific job that examines each possible path for data, chooses the suitable one and transmits the data packets where it needs to go as fast as possible. In addition, it contributes to enhance the performance of the network as the optimal routing algorithm helps to run the network efficiently. The clear performance advantage provided by routing procedures is faster data access. For example, the routing algorithm takes a decision that determines the best route based on the location where the data is stored and the destination device that asking for it. On the other hand, the network can handle much traffic at one time, but it cannot exceed the allowed bandwidth as the maximum rate of data that the network is able to transfer. However, the overloading problem is real and still exists. To avoid this problem, the network chooses the route based on the available bandwidth space.

The main challenge in all network types is the optimization of network performance, but the situation is different in SDN because the technique is changed from distributed approach to a centralized one. The characteristics of SDN such as centralized control and programmability make the possibility of performing not only routing in a traditional distributed manner but also routing in a centralized manner. The first advantage of centralized routing using SDN is the existence of a path to exchange information between the controller and infrastructure devices and because the controller has the information of the entire network, flexible routing can be achieved. The second advantage is related to the dynamical control of routing due to the capability of each device to change its configuration based on the controller commands [4]. On the other hand, The essential function of TE is finding an obvious route to achieve the bandwidth requirements of the network consequently optimizing the network performance [1]. Routing optimization has a key role in TE by finding efficient routes to achieve the desired performance of the network [2]. In particular, it is based on one particular idea: to achieve that traffic is routed according to accurate traffic requirements [3]. These traffic requirements are variable depending on an analyzed dataset that considered if it is data or traffic control.

2.7.1 **Investigation into Flow Analysis**

Non-redundant and relevant features are the goal of any feature selection technique as it is essential for solving the dimensionality problem in machine learning [124][125]. The scalability, reliability, and accuracy of machine learning techniques facilitate feature selection by selecting valuable attributes. In data analysis, feature selection is considered contributory for prediction by selecting related features. There are different measures of evaluation that are used for producing a subset of good features in feature selection. Dependence measures, uncertainty measures and distance measures are the three categories of evaluation measures. Based on many studies on feature selection, evaluation function is classified into five types distance, consistency, information, classifier error and dependence. [126]–[129] prove that most of feature selection techniques have been suggested for classification methods. Many feature selection algorithms use the system of statistical measurements such as correlation, mutual information and the measurement of gain of information [73]. [127] Clarified that Filters, Wrappers and Embedded methods are the general approaches of feature selection based on measurements of evaluation. When the three methods were compared, it was found that, the filter methods have low complexity of computational, but it cannot guarantee the accuracy of the learning algorithms. In contrast, the wrapper methods select the best features, so it provides good accuracy with large computational complexity [129]. The one that is more efficient than the others is the embedded method because it runs the process of feature selection as part of the training procedure. Therefore, it is specifically used with learning techniques [127]. However, the increasing of data dimensionality is still challenge to many of feature selection techniques. Based on many studies [130]–[132], feature clustering is considered as another type of feature selection. Hierarchical algorithms and K-means are common in clustering methods.

Machine Learning (ML) is the term given to a set of powerful techniques for data extraction and the discovery of knowledge [133], [134]. Furthermore, one of the most favorable techniques to carry out network-data analysis and to automate configuration and management of networks because of its ability to make network elements 'learn' from experience by using the large quantity of data to make networks more intelligent and adaptive [31]. k-Means is a clustering algorithm, which stores particular pre-chosen k centers which it utilizes to generate clusters randomly, according to the similarity (often Euclidean distance) between all input objects [135]. There are many clustering techniques known as the best in many situations compared to the K-means algorithm[134]. However, K-means is common for many reasons. Firstly, the technique is classified as simple in its implementation. Secondly, K-means has the capability of fine-setting, therefore it is very effective as it can be integrated with better algorithms such as clustering of intensity [136], genetic algorithm [36][137], etc. Third, the limitations of K-means are known, and it was studied extensively compared to other algorithms that are less studied and have unknown limitations, so it is the preferred technique. Clustering is effective in a range of fields, including network management and security. In [138], similarity-based clustering of application traffic was executed by using two unsupervised clustering approaches: k-means and Expectation Maximization (EM). In addition, the Correlation-based Feature Selection (CFS) filter method was used to select appropriate attributes of a flow. Researchers in [26] concentrate on using ML methods for statistical flow-based traffic classification. The authors propose a new framework, which is Traffic Classification using Correlation (TCC) information, to handle the problem of very few training samples. The framework has been constructed by selecting suitable features and the correlated information of TCP flows to enhance the classification performance. In [27] a new unsupervised method for traffic classification was suggested to solve the problem of unknown applications through identifying traffic classes, based on flow statistical features, where automated flow classification and signature-based cluster aggregation were executed by finding a similarity between traffic clusters. While in [28], the C5.0 technique was used for application classification. A new set of features, which are burstiness and idle time, have been proposed to determine the type of applications that generate the traffic. The proposed features have proved their effectiveness to identify the type of applications as compared with previous studies. Another study of the use of unsupervised ML algorithms to identify applications was described in [32] . In this study, seven different application groups were concentrated. This work introduces a system, executed by a set of tasks that measures the network’s QoS. In accordance with [33], an amended k-means-based semi-supervised clustering method was used. Flow statistics-based traffic classification was applied, where layer four statistics were considered as inputs, to manipulate the classification process.

Recently, relevant deep learning techniques such as "deep neural networks", "deep reinforcement learning", "stacked auto-encoders", and Deep Boltzmann Machine have been used to solve network problems with respect to traffic analysis, security, and management. The study in [116] introduced a classification traffic method. This method starts with conducting traffic identification at the SDN switches and afterward performs a “global” classification of traffic through the SDN controller, which will be responsible for training, constructing, and refining the policies of QoS based on the information of learned traffic. Similarly, the study in [139] proposed a deep learning-based framework that unified feature extraction and classification process into one architecture. The “Deep Packet,” system can provide a traffic characterization scheme, in which the traffic of network is categorized into main classes such as FTP and P2P, and an application identification scheme, in which end-user applications such as BitTorrent and Skype are identified. In addition, the "Deep Packet" system can distinguish between encrypted and unencrypted traffic.

2.7.2 **Elephant and Mice flows identification**

The problem of precise identification of mice and elephant flows still needs to be handled [34], [140], [141]. In[142], the authors proposed a method for identifying flows as mice or elephants. Unsupervised and semi-supervised ML approaches have been used for classification flows in real time. They used three parameters as data transfers, flow rates, and durations for clustering. The accuracy of the proposed method was 90%. While the authors in [34] proposed a system that detects elephant flows in linear time through traffic volume estimation for every flow. The elephant flows are identified by counting their total bytes. They used a certain data structure (hash tables) to achieve elephant identification. This system applied two algorithms for maintaining the data structure, Median SUMming (IM-SUM) and De-amortized Iterative Median SUMming (DIM-SUM), where they use two different tables for calculation. On the other hand, other studies clarified that identification and rerouting of the flows that hold a large amount of data (elephant flows) effectively can lead to significant improvement in QoS. Authors in[143] presented a system called DARD (Distributed Adaptive Routing architecture for Datacenter networks). This system consists of three parts. The first is for detecting an elephant flow if a TCP connection has lasted for more than 10s. The second is a tracking monitor to check if all the paths connecting the source and destination switches are existing. The third part is a flow scheduler that shifts elephant flows from overloaded paths to under loaded ones. Another approach is proposed by [144], authors present a flow-scheduling algorithm, which dynamically adjusts the number of two types of paths according to the real-time traffic into low latency paths and high throughput paths respectively for the two types of flows to make full utilization of the bandwidth. As proposed in [7], a routing algorithm called Distributed Flow Scheduling (DiFS) system, defined the flow, which exceeds a threshold of 100KB as a large flow. After that, this flow will be transmitted to the destination by switching to a path with an abundant amount of bandwidth.

Recent works have been applying machine learning in network traffic classification. However, the limitations of existing studies are in the number of connections used as well as the features identified. The novelty in our work is that we have first introduced the investigation of the effect of the number of PCA components on the accuracy of the flow clustering process. Further, k-means were applied to cluster the traffic based on network performance metrics using a dataset that contained 1 million complete connections. Finally, full and precise identification of flows as elephant and mice were performed using pre-decided threshold values.

2.7.3 **SDN- based flow routing optimization**

SDN is currently utilized in numerous fields, from Internet of Things (IoT), and wireless networking, to cloud computing and datacenters, focusing on both QoS and security. In [145] SDN-based framework for the IoT environment has been proposed. This framework allows significant achievements of quality level in a heterogeneous environment of wireless networking for the task related to IoT. These achievements have been done by designing an SDN controller in IoT multi-network for providing flexible and efficient management of flows and the accessible network resources. According to [96], the authors present an application that used several ML methods to classify network traffics. This application started with gathering statistics on OpenFlow traffic from the switches in the SDN environment, which was deployed in an enterprise network. The work aimed to evaluate the performance of supervised classification approaches comparatively. Similarly, in [146] a management design, named ATLANTIC, was introduced to perform anomaly detection, classification, or mitigation. The ATLANTIC framework uses information theory for deviation calculations in the entropy of flow tables integrated with machine learning techniques for traffic classification. Thus, the design has the ability to categorize traffic anomalies and block malicious traffics by using the available information.

An application-aware multi-path flow routing framework, which integrates ML and SDN, was proposed in [115]. In this architecture, the controller prioritizes each flow using ML and determines a path depending on its priority. In the same context, the authors in [116] introduce a QoS-aware traffic classification model for SDN. The model uses the requirements of QoS to classify traffic into different classes by using deep packet inspection (DPI) and a semi-supervised machine learning algorithm. The management of networks is becoming a big challenge due to the growth of network size, the volume of traffic, and the diversity of QoS requirements; to consider this level of complexity, SDN provides flexibility and scalability of network management. Clustering traffic to provide improved network management also proved successful to a certain degree in previous studies, where the classification of traffic based on user interests [147][148], was then applied to an SDN environment [149]. Moving further, configuring a large complex network is also a challenging job. it is becoming difficult and increasingly worrisome with the passage of time, as network administrator needs to perform sophisticated actions in order to manage network tasks, thereby, to address this problem, authors in [104] proposed an event-driven network control solution based on the SDN, named Procera, to simplify various aspects related to network operation and management. The authors proposed that network operators could utilize four control domains, such as traffic flow, data usage, time, and authentication. Based on the listed studies, it is increasingly apparent that clustering of traffic and SDN are indeed likely to lead to more effective handling of traffic. However, studies tend to look at optimizing the quality of individual flows, applications, or mixes of applications as employed by end users, without considering the characteristics shared by the flows in the first instance from a performance perspective. Indeed, flows may encounter similar levels of packet loss or delay, as well as share characteristics such as file length or application requirements.

The essential reasons for network performance degradation are network congestion and imbalance in network load, which may cause by the inefficient routing of elephant and mice flows. According to [150] the load-balancing mechanism has been proposed using multipath routing of elephant flows in SDN to improve the utilization of the network. The applications of software-defined networks have attracted a lot of interest to solve the problem of network management in the last decades. In [151], the authors have applied a dynamic routing mechanism in SDN to solve the problem of inconsistent distribution of network traffic that causes congestion in network links. Knob et al. [152] have introduced a system that can give the network operator the ability to define templates that can re-route elephant flows for the specific objective to address the high impact of elephant flows on the overall network traffic. To improve network performance, researchers in [153] have proposed to detect the elephant method where the routing algorithm scans the network to find all the paths available between source and destination and calculates the link bandwidth of different paths available. Likewise, in [154], authors have presented a combination of detecting elephant flows and re-routing them to provide efficient resource utilization. In contrast, [155] suggests a method that routes mice flows efficiently using SDN technology by reducing the routing rules. In the work of [156], re-routing elephant flows (EFs) using an ant colony optimization--based technique has been presented. Load balancing in SDN links has been taken into account. This technique, known as DPLBAnt, is formulated in SDN as a shortest-path problem that can alleviate the high controller-switch load. The proposed method first detects elephant flow by employing a pair of classifiers on both the SDN controller and the switches. The switches sift through the majority of EF candidates, resulting in accurate and efficient EF detection. Then, DPLBAnt obtains the SDN's global state, from which the most optimal paths for congested links are retrieved and EF are redirected appropriately. In [157] a deep Q-learning (DQL)-based routing strategy for autonomously generating optimal routing paths for SDN-based data center networks has been proposed. Deep Q networks are trained to meet the different demands of mice-flows and elephant-flows in data center networks by achieving low latency and low packet loss rate for mice-flows, and high throughput and low packet loss rate for elephant-flows. Furthermore, port rate and flow table utilization to describe the network state were selected, taking into account traffic distribution and the limited resources of data center networks and SDN. A [158] they proposed DeepRoute which is a model-free reinforcement learning method that converts the path computation problem to a learning problem. DeepRoute learns strategies to manage arriving elephant and mice flows from the network environment to improve the network's average path utilization. In the study of [159] NNIRSS which is a neural network (NN)-based intelligent routing scheme was presented for SDN. It uses NN to achieve data flow transmission patterns and replaces the flow table with a well-trained NN in the form of an NN packet. To meet the QoS requirements of network applications, the route of data flow can be predicted based on its application type. In addition, they develop an intelligent routing mechanism based on radial basis function neural networks. In addition, an APC-K-means algorithm for determining radial basis function centers by combining APC-III and K-means was proposed. In the work of [150] a load-balancing-based dynamic multi-controller deployment scheme was proposed. They convert the flow requests into a queuing model and consider the traffic propagation delay and controller capacity as two major factors influencing the multi-controller deployment. For network planning, a modified affinity propagation algorithm (PSOAP) based on particle swarm optimization is proposed in the initial static network to solve the problem of clustering performance being affected by the initial values of the bias parameters and convergence coefficients. To achieve controller load balancing, the dynamic traffic network reassigns switches in different sub-domains using the breadth-first search (BFS) algorithm.

Overall, it is found that non-of the existing approaches for flow routing in SDN have accomplished an external application for identifying mice and elephant flows, enabling flow type and topology-aware routing. The goal of this work is to propose a framework that can accomplish this by using two blocks, namely, external applications and traffic analysis. Furthermore, it provides an algorithm for routing by using a short route for mice flows and the widest routes for elephant flows. Routes cost calculation is done recursively, which provides more efficiency and enables consistency with real-time constraints.

2.8 Conclusion

Network management is a set of actions, which guarantees the utilization of all network resources in the best possible way. Any network management system should be capable of monitoring all devices on the network and fulfilling any management procedures required by these devices. One of the functions of any network management system is to simplify the monitoring of the network performance task. The outcome of monitoring is to obtain valuable information regarding network performance, by using specific tools, which have been designed for this purpose. In addition, there are tools, which are used as analyzers. Their function is to collect statistics on the amount and type of traffic on a single network portion. Increasing volumes of network traffic, increasing speed of networks, and solutions for capturing full-packet traffic require a huge amount of storage every day. Unsupervised Machine Learning (clustering) has been used in different areas of networking, such as application, security, and classification of user behavior. In addition, clustering techniques have been used in the network performance analysis area, which will be covered in this thesis.

Nowadays, network management has become more complicated and less docile to administer manually, because of the increase in dynamicity, heterogeneity, and complexity of the network [150][160]. Therefore, the optimization and decision-making automated approaches by using ML and artificial intelligence (AI) have become necessary. The aim of unsupervised ML is to build a structure or pattern using the inputs without the need to pre-define the output class. In addition, unsupervised ML has the ability to transact with traffic types and a variety of datasets based on previous studies. Consequently, the employment of unsupervised ML has been increasing to improve the performance of networks and enhance the services such as anomaly detection, Internet traffic classification, and TE. The term TE, which includes the measurement, characterization, modeling, and control of traffic, is described as evaluating and optimizing the performance of network performance. The main aim of TE is to achieve all the requirements of a network through the optimal use of the network resources. It is also worth mentioning, that TE is responsible for designing a reasonable mechanism of routing to control the traffic of the network for improving the utilization of network resources.

Chapter Three: Investigation into Network Performance Analysis

3.1 Introduction

The analysis of network performance is the use of network data to comprehensively understand trends of network performance. With the analysis of network data, the network performance variables can be identified and measured to assess network performance, diagnose network performance issues, and exposed long-term problems. Consequently, the analysis of network performance has benefits such as baseline establishment and understanding the limits of network performance. Additionally, Analysis of the network traffic can extensively provide knowledge as regards the network resources and data infrastructure devices that are responsible for routing the packets.

In traffic analysis, the robust and effective election of features to characterize the traffic is still a real challenge. Practically, the characterization of network traffic has been categorized as either flow-based using characteristics such as flow bytes, flow duration, etc., or/and packet-based using properties like packet size, inter-arrival time, etc. Unfortunately, there is still a shortcoming of studies on the techniques of traffic characterization of network data. In the literature, the characterization process can be accomplished according to its purposes such as encrypted traffic [161][162], encapsulation of protocol [163], applications [11], or type of application like chat, or streaming [164].

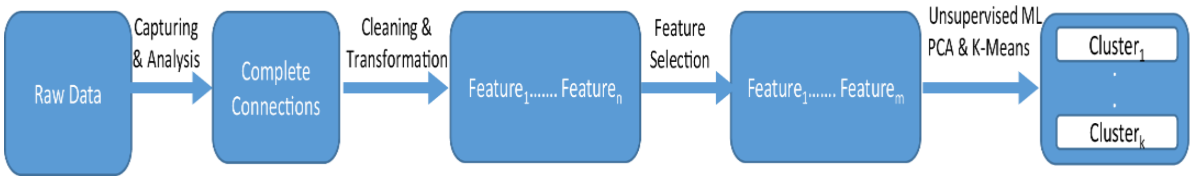
The characteristics of network traffic are still pivotal to most applications due to their influence on the QoS features of communications links. Therefore, network traffic characterization is considered an integral aspect of network services identification and communication flow management between devices. For example, in IoT environments, the characterization of network traffic contributes to management capabilities promotion, network capacity adjustment, reliability of flow packets delivery, improvement of network security, and network QoS ensure [165].

Due to the complexity of the network infrastructure and the mix of traffic, designing a network to support QoS is, therefore, not an easy task. The fundamental step is to understand the characteristics of different types of network traffic. Therefore, the modelling of data traffic becomes a decisive and indispensable step. This project will propose a comprehensive characterization of traffic with respect to network performance to manage the network efficiently.

This chapter presents an innovative traffic characterization approach based on network performance descriptors to manage the network efficiently. In addition, it takes into account the possibility of reducing the number of features for accurate characterization of network traffic. Two experiments have been conducted to investigate the efficacy of the approach. These experiments follow the methodology provided after the description of the proposed approach.

3.2 A Novel Approach for Network Performance Analysis

The proposed method seeks to develop a flow-based characterization technique to analyze and characterize network traffic. According to [166] the procedure of characterization can be carried out based on its objectives such as security (e.g. encrypted traffic and VPN or HTTPS), applications (e.g. name or type of application), and user behavior. The proposed approach introduced using common characteristics between the traffic depending on network performance features for the first time to characterize the network flow. In view of this, Unsupervised ML techniques are applied to personify similar flows and isolate them based on their features that are related to network performance such as delay, throughput, packet loss, etc. By clustering analysis, flows will be summarized, and the central flow of each cluster may be selected as the representative of that cluster. This can be performed to improve the management of networks by ensuring a fair allocation of network resources for the flows and accommodating other flows with different requirements. Figure 3.1 shows the innovative proposed unsupervised ML scheme.

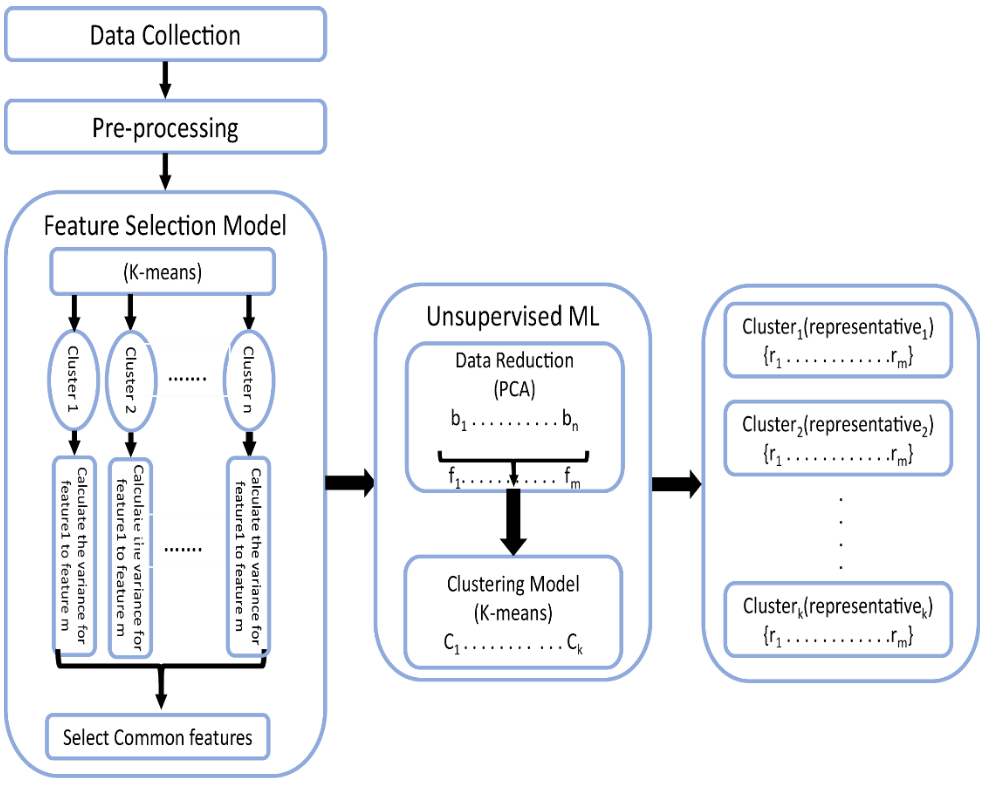


Raw Data
Capturing & Analysis
Complete Connections
Cleaning & Transformation
$Feature_1$....... $Feature_n$
Feature Selection
$Feature_1$....... $Feature_m$
Unsupervised ML PCA & K-Means
$Cluster_1$
$Cluster_k$

Figure 3.1 The Proposed Scheme

3.3 Proposed Methodology

Network traffic characterization is important in network management, and it is used to analyse and find the solutions for the problems of network performance. In accordance with this, this experiment tries to evaluate and validate the characterization of network traffic based on network performance metrics by following the methodology depicted in Figure 3.2.



Data Collection
Pre-processing
Feature Selection Model
(K-means)
Cluster 1
Cluster 2
Cluster n
Calculate the variance for feature1 to feature m
Select Common features
Unsupervised ML
Data Reduction (PCA)
b_1 .... b_n
f_1 .... f_m
Clustering Model (K-means)
C_1 .... C_k
$Cluster_1(representative_1)$ $\{r_1 .... r_m\}$
$Cluster_2(representative_2)$ $\{r_1 .... r_m\}$
$Cluster_k(representative_k)$ $\{r_1 .... r_m\}$

Figure 3.2 The Proposed Methodology

3.3.1 **Data Collection**

Recently, the activities of network users have increased rapidly, and need to keep up with the problems of analyzing huge traffic. Different solutions and techniques have been suggested for dealing with data analysis, such as data clustering, and data reduction. With a view to construct a set of connections, concentration has been placed on the TCP protocol because it provides a wide variety of features to characterize flows. The tcpdump [166] tool was used to capture the raw data at the Center for Security, Communications, and Network Research (CSCAN) at the University of Plymouth. This data is stored as a pcap file containing a collection of 19,004 records. The captured and filtered network traffic was analyzed using the tcptrace analyzer [167] to yield 11,593 completed bi-directional flows. Each one contains classical fields related to a flow such as IP addresses, port numbers, packets and byte counters, etc. These bi-directional flows provide full visibility of network performance. A total of 146 features, related to network performance, were extracted from the header field of each TCP packet to generate a dataset. The tcptrace records were afterward subjected to pre-processing before undergoing to unsupervised ML techniques as detailed in the following section.

3.3.2 **Pre-processing and Feature Extraction**

Generally, raw data is incomplete (contains missing values), noisy (there are duplicates, errors, and outliers), and inconsistent (in different formats). Therefore, it needs to be transformed into an understandable format for ML. techniques. Data preprocessing can handle these issues by performing the following tasks:

(i) Data cleaning entails filling in missing values, smoothing noisy data, identifying or removing outliers, resolving inconsistencies, and eliminating duplicates, irrelevant observations, errors, and unnecessary columns.

(ii) Data Transformation requires feature construction and data normalization.

In this step, Principal Component Analysis (PCA) has been employed to handle outliers and to reduce data as in Sub-section 3.3.4 while z- Score normalization technique has been utilized to scale features' values that fall within a specified range to ensure that unit of feature doesn't distort the closeness of cases. In the normalization step, a built-in scale() function has been used to achieve z-score normalization to convert data into 0-1 range. The resulted dataset contains 129 features (Appendix -1), and 11593 connections is ready for feature selection. All the pre-processing steps are executed using R-based script.

3.3.3 **Feature Selection Model**

High-dimensional data contains thousands of features, but not all of them are somewhat relevant or vital to achieve the goal function. Therefore, excluding the irrelevant features from the data will dispose of their negative impact on the result of the target function [25]. Furthermore, some of these features are redundant, which means that many of these features may have the same effect on the goal function outcomes. To reduce the computation of workload for high-dimensional data; one feature can represent all the redundant features [26]. Finally, it is interesting to build an efficient feature selection model, in which the goal is to identify the smallest group of independent features with the most influence in the clustering process.

In the current literature [146], feature selection algorithms are mainly categorized into three groups, i.e. (i) feature selection algorithms that are used to remove irrelevant features, (ii) feature selection algorithms that are used to remove redundant features, and (iii) feature selection algorithms that are used to remove features both irrelevant and redundant features.

The typical clustering algorithm, which is relevant to be used for large number of variables is k-means, which is simple and fast. K-means clustering has been applied as a technique to select specific features in this work by using unlabeled data (i.e., categories or groups of data are not defined). Feature selection process is implemented by excluding the control features of each cluster. The main aim of this process is to minimize the dataset and consequently diminish the size of high dimensional data and choose the essential features. A vector of 52 features is the output of this stage as will contribute to speeding up the clustering process in the future. These features illustrated in Table 3.1 as Feature set2.

Table 3.1 Feature Set 2

Seq.	Feature	Seq.	Feature
1.	first_packet	2.	avg_owin_a2b
3.	total_packets_a2b	4.	wavg_owin_a2b
5.	total_packets_b2a	6.	wavg_owin_b2a
7.	ack_pkts_sent_a2b	8.	initial_window_bytes_b2a
9.	ack_pkts_sent_b2a	10.	ttl_stream_length_a2b
11.	pure_acks_sent_a2b	12.	ttl_stream_length_b2a
13.	sack_pkts_sent_a2b	14.	throughput_b2a
15.	sack_pkts_sent_b2a	16.	RTT_samples_a2b
17.	dsack_pkts_sent_a2b	18.	RTT_samples_b2a
19.	unique_bytes_sent_a2b	20.	RTT_min_a2b
21.	unique_bytes_sent_b2a	22.	RTT_min_b2a
23.	actual_data_pkts_a2b	24.	RTT_max_a2b
25.	actual_data_pkts_b2a	26.	RTT_max_b2a
27.	actual_data_bytes_a2b	28.	RTT_avg_a2b
29.	actual_data_bytes_b2a	30.	RTT_avg_b2a
31.	rexmt_data_pkts_a2b	32.	RTT_stdev_a2b
33.	rexmt_data_pkts_b2a	34.	RTT_stdev_b2a
35.	rexmt_data_bytes_a2b	36.	post.loss_acks_a2b
37.	rexmt_data_bytes_b2a	38.	post.loss_acks_b2a
39.	outoforder_pkts_a2b	40.	ambiguous_acks_a2b
41.	outoforder_pkts_b2a	42.	ambiguous_acks_b2a
43.	sacks_sent_a2b	44.	segs_cum_acked_a2b
45.	sacks_sent_b2a	46.	segs_cum_acked_b2a
47.	avg_win_adv_b2a	48.	duplicate_acks_a2b
49.	max_owin_a2b	50.	duplicate_acks_b2a
51.	max_owin_b2a	52.	triple_dupacks_b2a.

3.3.4 **Data reduction Model**

Nowadays, network measurements have hundreds or thousands of data available for a single experiment, therefore the statistical approaches are considered challenging to deal with such high-dimensional data [168][169]. Nevertheless, this data might contain highly redundant variables and can be efficiently minimized these number of variables without losing any significant information [170][171]. The mathematical methods that are designed to achieve this reduction are called dimensionality reduction techniques [172]. Data reduction involves the transformation of high dimensional space to low dimensional space by eliminating dependent or highly correlated variables and feature selection. Furthermore, reducing the dataset leads to a group of advantages such as improving the quality of data, increasing the efficiency of the algorithm work, better accuracy achievement, and clarifying pattern design and examination researchers in [173]. Additionally, the cost of computing will be reduced, dimensions visualization enhanced, and the results improved [174][175].

Several techniques have been utilized to deal with the high-dimensionality problem in traffic analysis such as PCA, Isomap, autoencoder, etc. Choosing the convenient technique depends on the volume of the dataset, the aim of the analysis, the computational resources, and data complexity e.g., image or numerical, linear or nonlinear. PCA is one of the top-known dimensional reduction techniques and it is still preferable in the classification context [176]–[178]. PCA compared with other techniques like autoencoder is faster [179] and accelerate the convergence of the model [180]. The core difference between PCA and lies in the way they carry out dimensionality reduction. In PCA, the encoder and decoder are linear methods and might be extended to work with nonlinear data [181].

PCA utilizes orthogonal transformation on the data as a statistical procedure to transform a specific number of correlated variables into a minimal number of uncorrelated variables, which are called principal components. These components are arranged based on their variability in descending order. Being an essential approach for exploratory analysis of data, PCA takes data in n dimensions and makes visible the maximum variability in the data by rotating them. Primarily, PCA has been used for dimensional reduction, as input data is reduced without losing the important information in the data. The choice of the number of principal components is a critical issue in taking a decision. Many studies conducted on choosing the number of components such as cross-validation approximations [182]. Optimally, the reduction process is based on the proportion of the average squared projection error to the total variation in the data such that must be less than or equal to 1%. This process retains 99% of the variance as principal components. The techniques of feature selection are applied as a dimensional reduction [183], [184] in spite of there being a difference between them [185]. Feature selection is performed usually as a supervised process, which generally can be done well, but is not scalable and apt to judgment bias. On the other hand, Dimensionality reduction is an unsupervised task, as new features (dimensions) have been created instead of choosing a subset of features.

3.3.5 **Clustering traffic Model**

In the last decades, the employment of unsupervised machine learning techniques using unlabeled data has become common for network performance improvement and providing services, such as TE, optimization of QoS, anomaly detection, and Internet traffic classification[134][186][187]. These techniques make machine learning general, flexible, and automated as they provide the ability for analyzing data without having to formulate useful features manually and label them [134]. One of the powerful unsupervised learning methods is the K-means algorithm, which is a famous partitioning clustering technique, compared with other algorithms such as DBSCAN. The k-means algorithm can be applied efficiently with sparse and large datasets whereas DBSCAN fails with such datasets because it depends upon the Euclidean definition of density [188]. Furthermore, k-means requires one parameter that defines the number of clusters (K) while DBSCAN requires two parameters radius(R) and minimum points (M) [189]. Moreover, k-means works better than DBSCAN when utilized with real-time dynamic datasets because it consumes less time for computation [189]. A brief definition of K-means is an algorithm that aggregates observations with identical characteristics into k clusters. It starts deducing the model of clustering based on a statistical vector of points as an input. Then, it distributes these points into cluster form. The clustering process is performed based on the algorithm [190] illustrated as below:

K-means Algorithm

```
Input:
        P= {P1,P2,P3,........,Pn}: the set of data points and
        V = {v1,v2,.......,vc} the set of centers.
Output:
        C={c1,c2,...............ck} the set of clusters

Start algorithm:
Step 1: Select 'c' cluster centers randomly.
Step 2: Calculate of the distance between cluster centers and each data point.
Step 3: Assign data points and a corresponding cluster center based on the
            minimum distance between them.

Step 4: Recalculate of the new cluster center by:

            $V_i = \left(\frac{1}{C_i}\right) \sum_{j=1}^{c_i} P_i$
            Where, 'ci' is the number of data points in ith cluster.

Step 5: Recalculate of the distance between new cluster centers and each data
            point.
Step 6: When there is no data point to be reassigned
          then end,
Otherwise recall from step 3.
```

Furthermore, K-means has commonly used with unlabeled data where categories or groups of data are not defined. Therefore, to determine the optimization of the K-means clustering the accuracy will be calculated using two measurements Within Cluster Sum of Squares (WCSS) and Between Clusters Sum of Squares (BCSS).

In this model, K-means has been utilized to cluster network traffic flows based on network performance parameters. It started with determining an optimum 'k' value. This was carried out using the Elbow method, which limits the number of clusters to a value beyond that appending another cluster does not improve the modeling of data. The elbow technique chooses the appropriate number of clusters by repeating the k-means technique on a dataset for a range of values of k (1-10) and computing the sum of squared errors for each value of k. Two sets of features were used: Feature set1 contains 129 features as mentioned in Sub-section 3.3.2, while Feature set2 includes 52 features as shown in Sub-section 3.3.3. Feature set2 is a subset of Feature set1 where 77 features, which represent the control or driven features for each cluster, have been removed to minimize the total number of features. Moreover, this model includes the calculation of cluster convergence and profiling clusters, which identify the behavior of each cluster.

3.3.5.1 *Cluster Convergence*

Analysis of clusters is an exploratory method. Any clustering algorithm cannot achieve a perfect endpoint, but it can reach a point where the error is minimized. Convergence is a condition that rules the minimal change in cluster centers. It sets a ratio of the minimum value that the distance between initial cluster centers reaches. In other terms, true convergence is done when reaching the point that there is no possibility of further improvement. Consequently, the analysis of clusters is considered good if the condition of cluster convergence is met. The threshold that has been used to restrict criteria for convergence was between 5% and 30% for the population of all clusters overall dataset. Therefore, in this research with a dataset that contains 11593 bidirectional connections; the minimum and the maximum number of connections in any cluster should be between 500 and 4000. Whereas the cluster with a size that exceeds these limits will be considered an outlier and it needs to be handled. Further, a few outliers can be easily incorporated by cluster structure.

3.3.5.2 *Cluster Profiling*

Profiling is defined as generating the description of the clusters based on input variables that have been used for the analysis of clusters. The purpose of profiling is to understand the network behavior within a particular situation. Profiling is effective when cluster analysis has been conducted on multivariate clusters, so the clusters can be described based on multiple aspects, provide an accurate description of the network behavior, and get inferences based on the provided information. After validating the convergence of clusters, it is essential to recognize the behavior of each cluster. In this stage, the aim of cluster profiling is to accomplish an obvious description of the type of flows in each cluster, and the behavior of each flow with respect to network performance metrics. This is achieved by mapping a combination of variables with respect to network performance such as packet loss, delay, connection size, throughput, and congestion window; the output of this stage will be a representative connection for each cluster.

3.4 Experimental Setup and Results

This experiment is carried out on an Ubuntu 14.0.4 LTS Operating System with a Linux kernel (3.14.4 x64). The CPU is Intel Core i7-1165 G7, 2.8 GHz, and 16 GB RAM. The experimental procedure has been applied by using a network pcap file. This file contains network traffic that was collected utilizing the tcpdump capturing tool in the laboratory of Plymouth University. In-depth analysis was accomplished on the captured data to debrief network performance features using tcptrace analysis tool. Unsupervised Machine Learning techniques (PCA and K-means) are implemented as a final stage to reduce the dimensionality of data and find the common features between connections. These techniques are applied using two sets of features explained in Sub-section 3.3.3. The hypothesis of this work was to clarify the possibility of characterizing TCP flows based on network performance metrics. In accordance with this, two questions are suggested:

- What is the reliability of characterizing TCP flows based on network performance metrics?
- What are the factors of clustering that may affect the accuracy of characterizing TCP flows?

In order to answer these questions, the below steps were implemented.

3.4.1 **Data Reduction Implementation**

The work started with extracting key variables (in form of components) from a larger set of variables that are available in both selected datasets. PCA technique was applied for the purpose of dimensional reduction to make the clustering algorithm more effective and efficient. Furthermore, PCA has been used before the clustering algorithm to analyze the correlation between the features of Feature set1 as a second purpose. Figure 3.3 Presents the correlation of the first ten variables in Feature set1. In this figure, correlation coefficients have been colored according to the value.

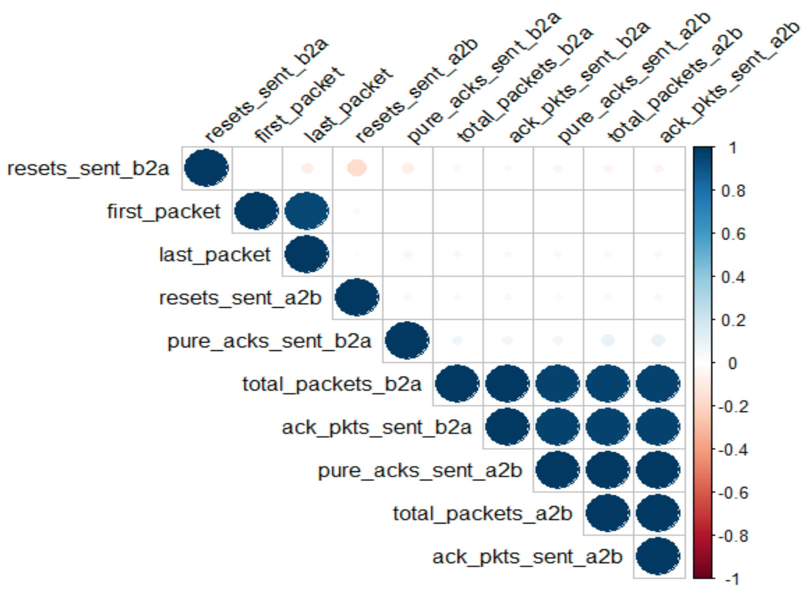


Figure 3.3 Correlation of First 10 Variables

In Feature set1, each of the 129 features contained approximately 0.77% (1/129) of the total variance in the original space. At least 0.77% of the total variance should be explained by the selected principal components. By applying PCA, the contribution information that each principal component makes to the total variance of the data has been gained. Table 3.2 shows the first 13 principal components and their associated eigenvalues, the proportion of variance, and cumulative variance. It is significant to retain a convenient number of components based on the trade-off between simplicity and completeness as a data reduction technique in PCA.

In this work, the first five PCA components have been chosen based on their variance as in Figure 3.5**Error! Reference source not found.** This number of components has been selected based on the results of using the Elbow method. These five principal components represent about 40% of the variation in the data as illustrated in Figure 3.4. Using the first five components will provide a good understanding of the data. Despite that, there is still a need to add more PCA components to cover as much as possible of data based on the trade-off between simplicity and completeness. This is one of the objectives of the work in Chapter 4.

Table 3.2 Components and Their Associated Eigenvalues

Component	**Eigenvalues**	**Variance%**	**Cumulative variance %**
Comp1	18.502	14.34	14.34
Comp2	13.443	10.42	24.77
Comp3	8.175	6.33	31.10
Comp4	7.551	5.86	36.96
Comp5	5.059	3.92	40.88
Comp6	4.253	3.29	44.18
Comp7	4.130	3.20	47.38
Comp8	3.626	2.81	50.19
Comp9	3.126	2.42	52.62
Comp10	2.972	2.30	54.92
Comp11	2.874	2.22	57.15
Comp12	2.587	2.00	59.15
Comp13	2.479	1.92	61.08

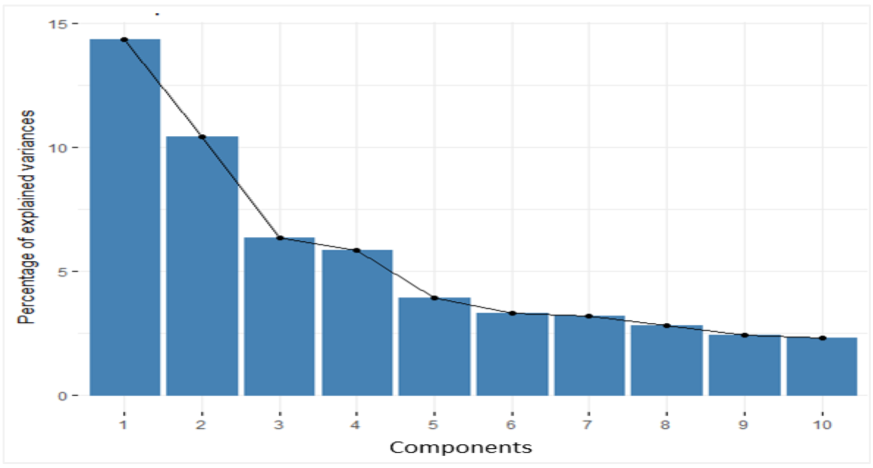


Figure 3.5 Percentages of Variance in Each Principal Component (Elbow Method)

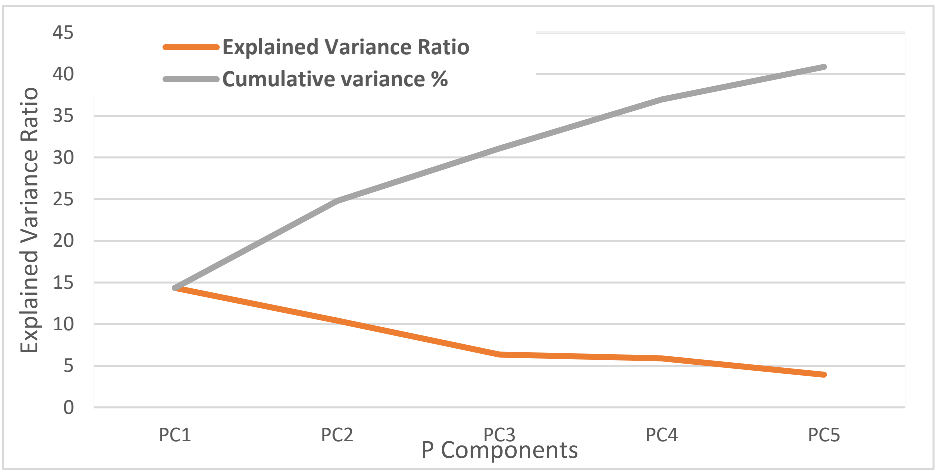


Figure 3.4 The Explained Variance Ratio and Cumulative Variance Percentage of the Five Principal Components

3.4.2 **Feature Selection Implementation**

In this part of the work, feature selection has been achieved by applying the K-means technique using a Feature set1, which contains 129 features. This is accomplished by excluding the driven features for each cluster. Driven features are the set of features that control the distribution of the observations between clusters. These features are extracted from the clusters using the filter approach. The main aim of this step is to diminish the size of high-dimensional data. As a result, the new dataset, which only contains the essential features, is a vector of 52 features. This feature set (Feature set2) has contributed to accelerating and increasing the accuracy of the clustering. Moreover, the PCA technique has been used to reduce the dimensions of this data.

3.4.3 **Clustering Implementation**

Clustering, or cluster analysis, is a task of unsupervised machine learning. Unlike supervised learning, which is predictive model, cluster analysis includes detecting natural grouping in data automatically. This will be achieved by interpreting the input data and discovering natural groups or clusters in space of features. On the other hand, Clustering is the mechanism that causes the instances with a stronger similarity to be gathered to each other than to the remaining instances. In this part, two experiments were conducted to use the K-means algorithm as a clustering technique with two sets of features with and without PCA. The factor that was used to determine the optimization of clustering process is the accuracy.

Accuracy: Each object is included to the closest cluster and then Euclidean distance is used to calculate the distance between the object and the cluster center. Each cluster center will be updated as the mean for objects in each cluster.

The within-sum of squares is:

$$\boldsymbol{WSS} = \sum_{C_i}^{C_n} \left(\sum_{X_i in C_k}^{X_m} (\boldsymbol{X_i} - \boldsymbol{C_k})^2 \right) \qquad 3.1$$

Where X is the data point in each cluster, C is the cluster centroid, k is the number of clusters [191].

The process is iteratively repeated until either it reaches the maximum number of iterations or the change of within-cluster sum of squares in two successive iterations is less than the threshold value.

3.4.3.1 *Analysis of using K-means Clustering Only*

This part of analysis involves employment of K-means clustering algorithm individually without using PCA as a dimensional reduction technique. The algorithm has implemented using the two different sets of features, which are mentioned in Section 3.4.2 to get an optimal clustering. As known, K-means has no ability to determine the number of clusters (k). Therefore, it has applied using two values of (k) based on elbow method (k=3 and k=100) as a lower and a higher limit, respectively.

As shown in Table 3.3, when the value of k is three, the within-cluster sum of the square will be high and the recorded accuracy is considerably low regardless of the number of features, and when comparing the accuracy for the two sets, it turns out that it is considerably low for Feature set1. As the value of k increases to 100, the within-cluster sum of the square value will decrease and the results represent a significant improvement in accuracy, yielding 80.7% for the Feature set1 and 92.9% for Feature set2 even though it consists of a reduced number of features. One interesting finding is when choosing the value of k beyond 100, the accuracy is slightly increased reaching 84.9% and 94.8% for Feature set1 and Feature set2 respectively.

Table 3.3 Accuracy of K-Means with Features Sets

Feature set	**Accuracy of K**						
	3	100	110	120	130	140	150
Features set1	15.2%	80.7%	81.8%	82.8%	83%	83.2%	84.9%
Features set2	43.2%	92.9%	93.6%	94.2%	94.6%	94.7%	94.8%

3.4.3.2 *Analysis of using K-means clustering with PCA*

PCA is a data reduction method, it is significant to retain a suitable number of factors on the basis of keeping a balance between retaining as few as possible factors (simplicity) and explaining most of the variation in the data (completeness). Kaiser's rule recommends only factors with eigenvalues exceeding unity should be retained. Intuitively, this means that any retained factor should compute at least as much variation as any of the original variables [192].

In this step of work, PCA is used to reduce the dimensions of each dataset to five components before applying the clustering algorithm. Using the PCA approach drives faster convergence of the K-Means clustering with few iterations compared to the basic K-Means method. The findings in Table 3.4 show that when applying K-means on the Features set1, with three clusters and five components of PCA, the accuracy is 36.7%. However, when the number of clusters is 100, the accuracy has increased significantly to 97.6% and starts to have a slight increase when the value of k is exceeded 100, and the clusters, which have only one connection have been eliminated. This keeps all resultant clusters in the safe range, which should contain between 5% and 30% of the overall dataset. Whereas, with the Feature set2, the result shows an accuracy rate equal to 60.6%. While the result has changed to the best level with 100 clusters or more, as the accuracy becomes 99%.

Table 3.4 Accuracy of K-Means with PCA

Feature set	**Accuracy of 5 PCs. and K**						
	3	100	110	120	130	140	150
Features set1	36.7%	97.6%	97.8%	97.6%	97.7%	97.8%	98%
Features set2	60.6%	99%	99%	99%	99%	99%	99%

Based on the aforementioned results for both sets of features there is a significant improvement in accuracy when k equals three, in contrast, the accuracy has a slight increase with k equals 100 or more.

According to the findings in the above two experiments (K-means with and without PCA) the accuracy has increased significantly in the two feature datasets. This means applying PCA before clustering, contributes to improving clustering accuracy even though the number of components is few (five PCA components).

3.5 Conclusion

The proposed scheme introduces flow-level characterization based on network performance metrics utilizing unsupervised machine learning techniques. The main idea behind the scheme is finding precise clusters of flows sharing certain characteristics that can be leveraged to improve the flow routing mechanism in a network as a final part of this project. The idea has been achieved through accomplishing experiments based on a group of concepts as follows: (i) trade-off between simplicity and completeness in choosing the suitable number of PCA components and (ii) the reliability of characterization TCP flows based on the accuracy of clustering that basically depends on the choice of (k) values. As a result, the accuracy of the clustering varied according to the use of K-means only and K-means with PCA as clarified in the experimental results section. The experiments were executed utilizing different feature sets to investigate the precision and flexibility of the innovative scheme.

Chapter Four: Traffic Analysis-based Flow Identification

4.1 Introduction

In network performance management, congestion, and unbalanced load are two main problems, which stem from the unfair use of the network resources by specific flows [15]. Some of these flows contain a large amount of data and consume many network resources, which hinders the use of these resources by others that have a small size. Consequently, the differentiation between these flows has an important role to solve network management issues. Characterizing and identifying these flows into Long-lived large flows as elephants versus short-lived small flows as mice can allow optimization of the network performance[17], [193], [194].

Recent studies in network engineering propose new strategies to optimize network performance by identifying and handling mice and elephant flows differently [29], [123], [142], [195]. These studies include the assignment of different flows to different queues, flow distribution across the links, and the creation of a policy of routing as a rule. These methods are achieved by either applying ML approaches, using certain data structures, designing adaptive routing architecture, or proposing a flow-scheduling algorithm. A wide variety of ML methods were used to classify network traffic, using statistical analysis based on conceptual classification [135][25].

The aforementioned studies have used one or two parameters, such as the size of flow and the duration, to distinguish between elephant and mice flows. In addition, they have assigned different threshold values for the identification process of these flows, 100KB to 10MB for total byte count per flow and 10s for duration threshold. Moreover, with the ever increasing Internet traffic, improving the network flow identification is still a challenge and of considerable interest to network operators, and the identification of these types of flows has a significant role in the enhancement of network management. However, the influence of the characterizing of elephant and mice flows on the identification process has remained unclear.

However, no previous research has specifically concentrated on leveraging TCP-based performance information to characterize and identify the flow. The present chapter proposes characterizing flow employing TCP based information and identifying flows based on specific thresholds and features. This has been achieved using unsupervised ML and thresholding techniques. These techniques have been implemented on real data captured and stored in a 2GB pcap file, which will be explained in Section 4.4.1. Hence, the main contributions of this chapter are three-fold:

1. Investigate the impact of a retained number of PCA components on the efficiency and efficacy of clustering.
2. Characterize flows according to TCP-based performance attributes.
3. Based on the above, introduce an innovative mechanism to identify mice and elephant flows.

The remainder of this chapter is organized as follows. Section 4.2 details the proposed flow identification methodology. Section 4.3. contains the experimental set-up. Section 4.4. Presents the results. Results are discussed in Section 4.5, and Section 4.6 concludes the chapter.

4.2 Flow Identification Methodology

Flow identification is a very important process that can be used to improve network management tasks such as scheduling, load balancing, and routing. Flow type determination (elephant or mice) is more challenging because of continual changes in traffic patterns [196]. In this work, a method that addresses the identification problem of elephant and mice flows and characterizes them by leveraging network performance metrics such as packet loss, round trip time (RTT), and throughput has been presented. Figure 4.1 displays the major stages of flow characterization using unsupervised ML and identifies these flows as elephants and mice. The detailed steps of the proposed approach are discussed hereafter.

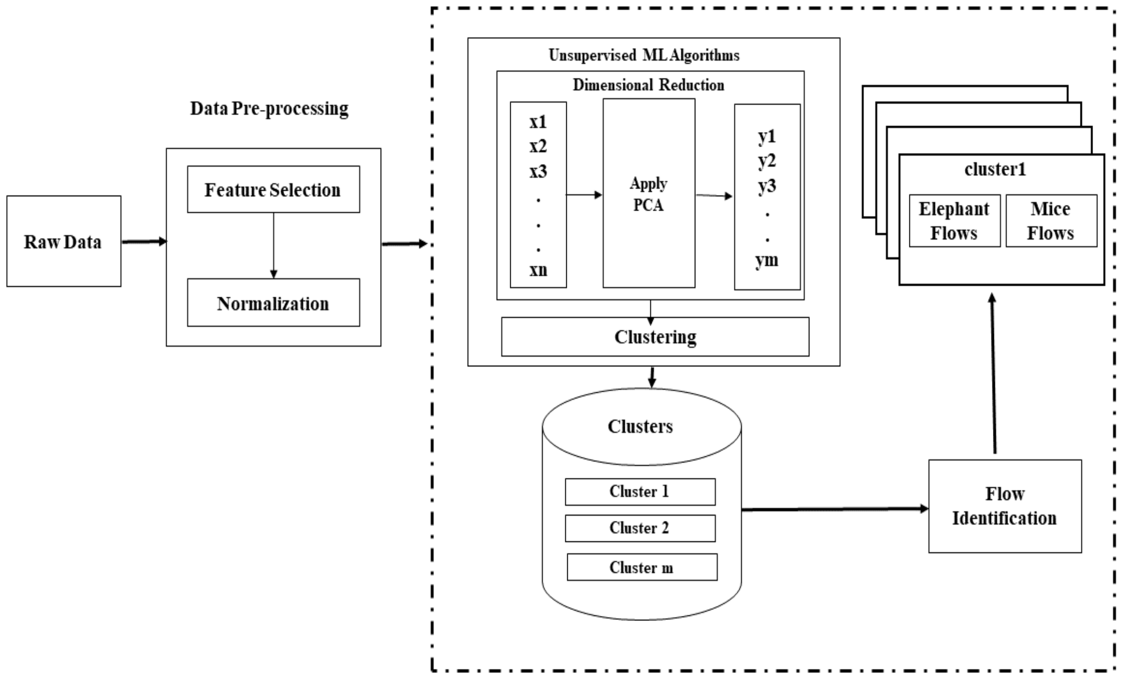


Figure 4.1 Proposed Methodology

4.2.1 **Flow characterization**

The reason behind using traffic characterization is that it is considered the typical solution for data analysis for understanding the behavior of network traffic patterns. The present work intended to characterize traffic behavior using statistics of network performance. After collecting these statistics, flows were grouped into unique clusters by employing unsupervised ML due to the similarity between flows. This is achieved by discovering hidden patterns or groups of data without the need for human intervention.

To simplify the ML step, data preprocessing is applied primarily. In this step, the raw data, which contain missing and non-numeric values, are manipulated to get a dataset of TCP parameters that will be used as input for the next stage. A set of features including 146 comprehensive attributes associated with the network connection provided by the tcptrace analysis tool. Irrelevant and useless records and features are removed as a data-cleaning step. All attributes with character values are converted to numeric values. The normalization step is also applied as a final step of data preprocessing to get a consistent dataset.

Due to the effectiveness of the reduction algorithm PCA with clustering as presented previously in Chapter 3, therefore PCA is utilized in this work to cancel the redundant features from the used dataset and obtain the optimal feature set to form the input of the k-means. To present the impact of the number of PCA components on the accuracy of clustering, different numbers of these components have been chosen to cover different percentages of data in this experiment.

To produce tighter clusters of the traffic flows, based on their performance features, an unsupervised clustering technique, which works with unlabeled data, was used after PCA. The main aim of clustering is to build robust labeling of clusters. The k-means algorithm has run to execute the clustering process, which is not considered the end step of the lifecycle of the proposed system. Therefore, the new data that contain unseen points will be considered as test data that presupposes the model has trained on our data, which will then be considered training data. K-means technique has been implemented using incremental values of k to show how the clustering accuracy changes with the changing of the number of PCA components and the value of (k).

4.2.2 **Elephant and Mice Flows Identification**

A flow is considered an elephant subject to its volume [3], [25]–[29], [33], [197] or depending upon preconfigured thresholds for size and duration [144][3][26][27]. Besides, some approaches were introduced [2], [32]–[34] to differentiate flows between elephants and mice based on the number of packets of a flow. Interestingly, the aforementioned literature explained that the approaches for identifying flows picked one feature (size or the number of packets) or two features (size and duration) to differentiate two types of flow as elephants and mice. This can indicate a motive to develop a novel identification mechanism. In this experiment, mice and elephant flows were identified for each resultant cluster from the clustering process.

The innovative mechanism uses pre-defined threshold values for the parameters of flow: number of packets, flow size, and flow duration. The mechanism identifies the flow as an elephant whenever it surpasses threshold values otherwise it is defined as mice. This work has proved that the flows identification process was efficient as it doesn’t require long processing time and does enhance the accuracy of results. The results show that the type of flow can be identified quickly (less than 0.4ms) and efficiently by the novel mechanism.

4.3 Experimental Set Up

The dataset description, dimensional reduction, and cluster construction will be presented in this section. Firstly, the criteria for choosing the connections, the features that will be used to describe each connection, and the necessary processes of pre-processing that are needed for the used dataset, will be explained. After that, unsupervised machine learning techniques that are used to reduce the dimensionality of the dataset and clustering the connections with respect to network performance metrics will be presented. Finally, flow identification based on pre-defined parameters will be shown.

4.3.1 **The Description and Preprocessing Dataset**

Real-world data have been traced in a university laboratory and captured by tcpdump. Such raw data carries about 2G bytes. The tcptrace tool has been used to analyze and convert this raw traffic to a dataset with two million connections. Complete connections have been selected as a criterion for the entire image of network performance. Therefore, the final figure of the dataset is 1 million complete connections, where each connection is formed by a sequence of TCP packets in two directions. On the other hand, each connection was represented by a vector of 146 extracted feature values. All these features are associated with the network performance such as the bytes number, duration, round trip time, the number of transferred packets, retransmissions' number, throughput, window advertisements, and so on. To handle noisy, incomplete, and inconsistent data, the preprocessing treatment is required for the used dataset.

The main aim of preprocessing is to construct a highly suitable dataset. The process started by excluding all inconsistent or missing values. After that, all categorical such as control attributes (FIN and SYN values) data have been converted to appropriate numerical values. To standardize all the variables of a dataset and to keep the closeness of cases, data normalization was performed as a final step. This step has been achieved for each variable's value by subtracting the mean value of the variable and dividing it by its standard deviation. As a result, the input dataset of the unsupervised ML model will contain 129 features with respect to network performance and 1 million complete bidirectional connections. These features have been elected due to their validity in the previous experiment in Chapter 3.

4.3.2 **Data Reduction and Data Clustering**

In this experiment, each connection in the dataset has been characterized by more than one hundred numeric variables; consequently, PCA has been used to warrant component representation accurately with a minimal number of variables. On the other hand, there are additional two reasons to use this technique. The first one is to handle the high correlation among the original variables. Secondly, the used dataset contains values that are extremely outside the range of expected and unlike the other values. These values are called outliers, which can negatively affect the process of clustering.

Generally, as a rule in PCA, each variable in the dataset will be represented by a component in PCA format where the components explain the full variation in data. As a result, the number of components will be equal to the number of variables in the dataset. Completeness and simplicity are the aims of PCA as a dimensional reduction technique. These aims will be achieved by retaining a suitable number of components based on the trade-off between the aims. To choose a suitable number of components, the “quick.elbow” function has been used in this experiment. Based on the cumulative percentage of components, this function determines the number of components that should be retained. To decide the appropriate number of components that achieve good clustering (clustering with high accuracy), PCA has run with a different number of components.

In clustering step, the k-means clustering method was run based on two criteria, which are the number of clusters (k), and maximum iterations. Using incremental values of k from 10 to 100 in steps of 10 and random initialization of cluster center, k-means was applied. This scenario was executed at first with the original dataset that contains 129 variables and secondly, it was followed using the reduced dataset with a different number of components. Accuracy is the metric that was used for measuring the clustering goodness. Therefore, there was no need to compare the cluster analysis results to the external information (class labels or ground truth). Inter-cluster distance and intra-cluster distance are the factors that have been used to calculate the accuracy of clustering.

4.3.3 **Extracting Flow Type (Elephant and Mice)**

A flow is defined as a sequence of packets that can be identified by a group of characteristics such as Source IP, Source Port, Destination IP, Destination Port, Protocol (TCP, UDP, etc.). Internet traffic is comprised 90% of mice flows, which are small volume, short and transmit a small amount of traffic while elephant flows, which are large, long-lived flows and transmit a large amount of bytes, represent only 10% of all flows. Network congestion is a major performance problem. In general, the temporary congestion is caused by mice, while the elephant flows induce the permanent congestion [198]. Therefore, specifying mice flows and identifying their priority automatically over elephant flows can optimize the forwarding mechanism. This can be achieved by improving the computing of transactions that depend on a small data block. This mechanism can reduce 30% of the completion time of an application [199]. There is flexibility to choose the appropriate threshold for identifying mice and elephants [Cisco's ACI][31][25]. While the aim of the clustering technique in this work is to characterize flows based on the performance metrics of the network, it does not specifically identify the type of generated flows per cluster if it is elephants or mice. Per-type identification consequently requires a separation of flows utilizing an independent technique which is the thresholding method. As a result, flow identification is performed based on threshold values for extracted parameters to retrieve the different flow types contained per cluster. The mechanism of flow type identification follows the algorithm below.

Flow Identification Algorithm

```
1.  PPR: set of rules for predefined parameters
2.  CID: set of predicted clusters (kmeansModel)
3.  FT: a string indicates mice or elephant flow
4.  CF: set of current flows
5.  FT ←Ø
6.  for each cid ∈CID do
7.  for each cf ∈CF do
8.  for each ppr ∈PPR do
9.      if cf.packetCount > 15 and cf.flowDuration > 5 and cf.avgPacketSize >=10 then
10.           FT ← 'elephant'
11.     else
12.           FT ← 'mice'
13. end if
14. end for
15. end for
16. end for
```

4.4 Results

This experiment was conducted using k-means clustering techniques with a new dataset where each connection was described by over one hundred variables and k has been set from 10 to 100. Additionally, the accuracy, which is the clustering internal index, was used for measuring the clustering structure goodness. The calculation of this index depends on two factors. The first is the inter-cluster distance between the observation and cluster center and the intra-cluster distance between cluster centers is the second factor. In general, the results show that the accuracy of clustering is proportional to the increase in k, where it increases substantially from 24.46% to 62.89%. Based on the aforementioned results and because the accuracy is still low. Therefore, the need to reduce the dimension of the dataset is necessary to make the process of clustering more efficient and effective. Further, the variables of the dataset have had a significant correlation. For these reasons, PCA has been used in this experiment to investigate the impact of a number of PCA components on the clustering process. Figure 4.2 shows the correlation of the first ten variables in the dataset. In this figure, the coefficient of correlation has been colored based on the value. Positive correlation has been represented by blue circles, while red circles present negative correlations. In addition, the correlation coefficients are proportional to the color intensity and the size of the circles.

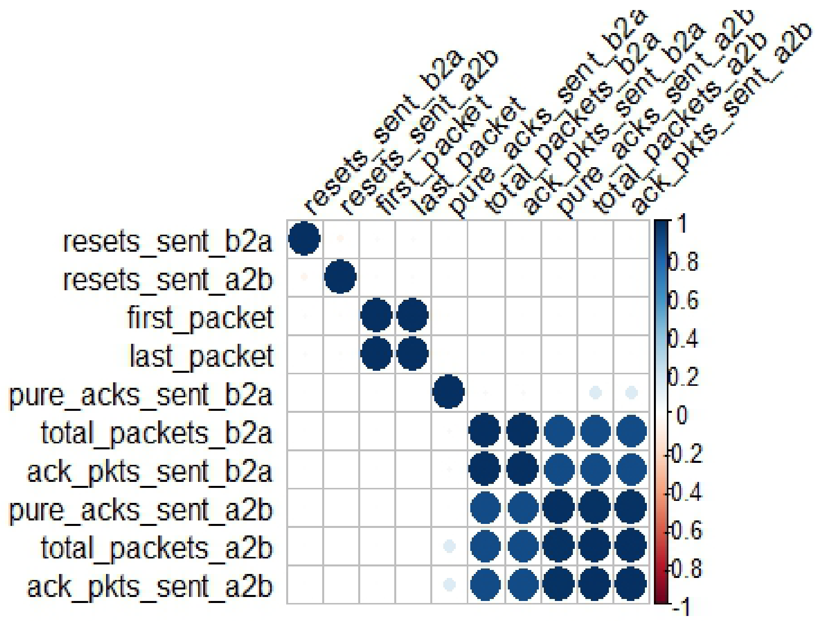


Figure 4.2 Correlation of the First Ten Network Performance Variables

The variance proportion explained by each principal component has been presented in Figure 4.3. The visualized variance of each component can help in determining the number of principal components that are needed to explain the data variation. The first three principal components clarify the most of data variation in the dataset. Since these components exemplify less than 30% of the data, therefore, there is a need to represent more data by increasing the number of components. Also, the first ten PCA components containing the features with the most variance and they covered 40% of the data, but they did not achieve the trade-off between completeness and simplicity, which is the major rule in the PCA technique. Table 4.1 explains associated eigenvalues, the proportion of variance, and the cumulative variance of the first 13 principal components.

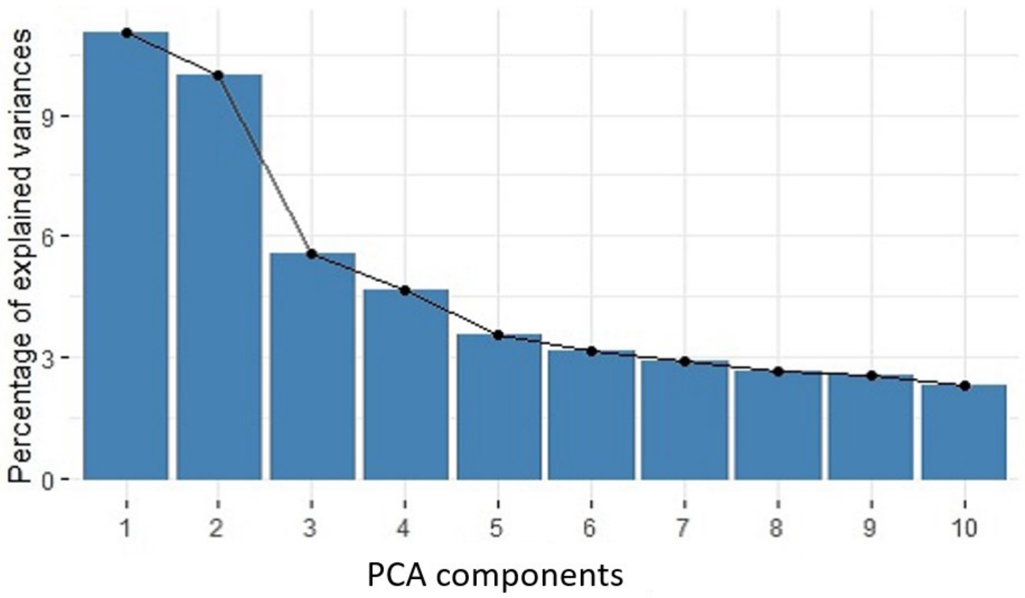


Figure 4.3 Percentages of Variance in Each Principal Component

Table 4.1 The First 13 PCA Components.

Components	Eigenvalues	Variance (%)	Cumulative (%)
Comp1	14.22	11.03	11.03
Comp2	12.84	9.95	20.98
Comp3	7.18	5.57	26.55
Comp4	6.03	4.68	31.23
Comp5	4.57	3.55	34.77
Comp6	4.10	3.17	37.95
Comp7	3.77	2.93	40.87
Comp8	3.41	2.64	43.52
Comp9	3.28	2.54	46.06
Comp10	2.97	2.31	48.36
Comp11	2.94	2.28	50.64
Comp12	2.87	2.22	52.87
Comp13	2.65	2.05	54.92

To achieve the essential rule of the PCA technique, the 13 PCA components that covered 50% of the data were used as inputs to the k-means clustering paradigm. In the clustering process, k has been set from 10 to 100. The findings depicted that the accuracy of clustering increased with the incremental k, where it increases significantly from 43.83% to 90.39%.

In order to comparatively investigate the effect of the chosen number of PCA components upon the accuracy of clustering, different numbers of PCA components (124, 57, 28, 13, 10) were elected taking into account the ratio of data covered by each number i.e., 100%, 90%, 70%, 50%, and 40%, respectively. The 124 components represent most of the features including the features with low variance, which leads to a change in the data distribution among the clusters. Consequently, the accuracy of clustering will be affected. The results show that the accuracy of clustering the reduced dataset is proportional to the increase in k regardless of the number of components. However, at least 10% of the overall accuracy was improved by minimizing the number of components. For example, for 100 clusters, the accuracy was 60.83% with 124 components, whereas it was 90.39% with 13 components.

In this experiment, in general, the results show that the accuracy of clustering both datasets is proportional to the increase in k, where it increases substantially from 24.07% to 91.4% whenever the value of k is increased from 10 to 100. Nevertheless, at least 10% of the overall accuracy was improved with minimizing the number of components when clustering the reduced data. In summary, using PCA before the clustering process and minimizing the number of principal components contributes significantly to increase the accuracy of clustering. In contrast, the accuracy of clustering is inversely proportional to the increase in the number of PCA components, where the accuracy decreases with 124 components, which represent 100% of the data as shown in Figure 4.4.

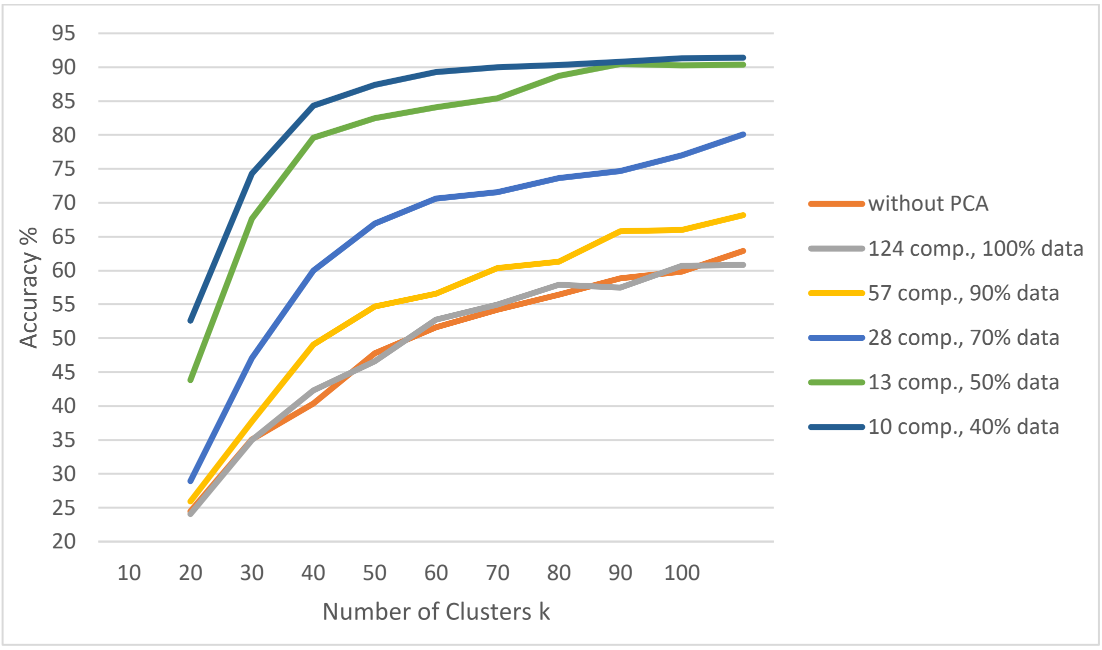


Figure 4.4 Relation between Accuracy and Number of PCA Components with Incremental Number of Clusters

In this part of the work, the aim was to reduce the dimensionality of data and the process of clustering based on 129 performance features. The high accuracy of the clustering process and the balance between simplicity and completeness while retaining the suitable number of PCA components were the criteria that have been used in this work. As shown in Figure 4.4, the results explain that using 13 PCA components, which represent 50% of data and 60 clusters (K=60) leads to getting high accuracy of 85.39%. Therefore, they represent the best choice. The distribution of data points for each cluster is shown in Figure 4.5, where each color is a cluster identity for 60 clusters.

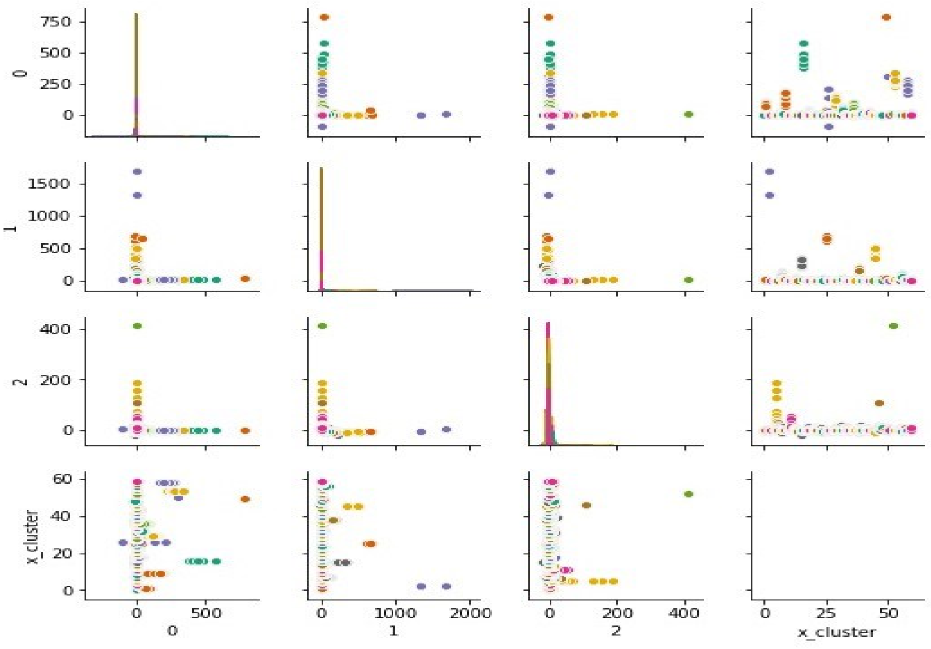


Figure 4.5 Distribution of Points in Clusters

In this context, it is essential to understand the behavior of flows inside each cluster resulting from the clustering process. In addition to the characterization of these flows based on network performance features, their type is needed to be recognized to solve the problem of resource allocation. Therefore, the final part of this experiment includes identifying these flows as mice or elephants based on certain features. The identification mechanism follows the thresholding method in setting values for the selected parameters of flow i.e., number of packets, flow size, and flow duration. The flow is identified as an elephant whenever its size surpasses 10KB, its number of packets exceeds 15 packets, and its duration is more than 5 seconds. In contrast, the flows with values fewer than thresholds for these parameters will be identified as mice. After applying the mechanism to all clusters, the percentages of elephants and mice are different from one cluster to another due to the clusters' size. For example, the percentages of identified mice and elephant flows in one of the 60 clusters were 89.92% and 10.07% respectively as Figure 4.6 depicts.

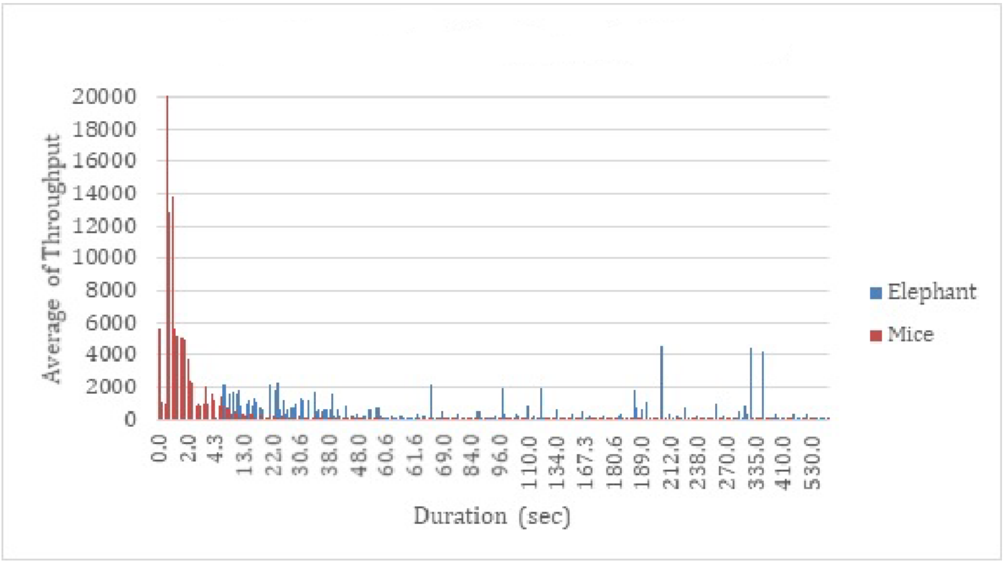


Figure 4.6 Elephant and Mice Identification

According to this, to undertake an evaluation of the flow identification process, the Naïve Bayes classifier [200] has been used. Here, the same ratio for training and testing datasets i.e., 50% of the flows were used to train the classifier and the remaining 50% of flows were used to test it. The Naïve Bayes classifier was applied to all resulting clusters from the clustering process. The empirical results present that elephant and mice flows were identified with an accuracy rate of 92.7% for all clusters. Figure 4.7 illustrate the confusion matrix [90] of flow identification of one of the resulting clusters that contains 179256 flows. the matrix shows four values, which are used to compute the accuracy (F-score [187]) of the classification process. These values will be clarified as follows from top-left to bottom-right:

- True Positives (TP): The values (2) refer to elephant flows that were correctly predicted by the classifier,
- True Negatives (TN): These (77454) mice samples are the negative flows that were correctly predicted by the classifier.
- False Positives (FP): These are (5) negative flows of mice that are labeled incorrectly as positive.
- False Negatives (FN): These are (12167) positive elephant flows, which were mispredicted as negative.

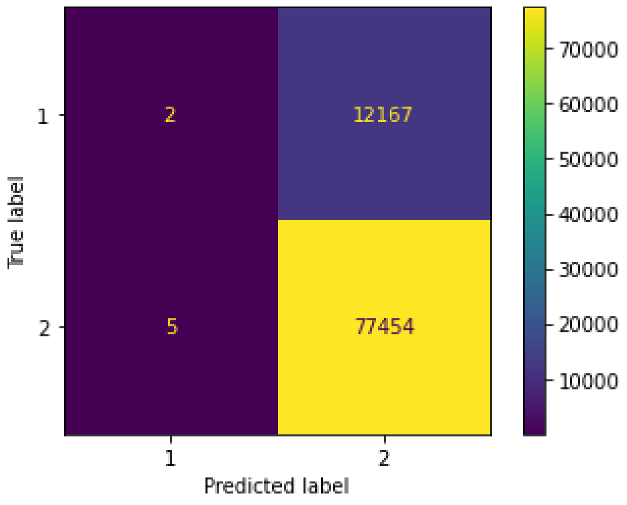


Figure 4.7 The Confusion Matrix

4.5 Discussion

Generally, the experiment results confirm that the proposed mechanism can be used effectively to differentiate the flows using the parameters related to that flow. The clustering conception is leveraged to characterize TCP flows through the performance metrics by using Inter-cluster distance and intra-cluster distance factors to calculate the accuracy of the clustering. Initially, the accuracy was low because of the high dimensionality of the dataset, the correlated features, and the outliers' existence. Hereafter, PCA was employed to get accurate clustering by treating all these problems. At the same time, this approach tested different numbers of PCA components to get the appropriate accuracy with achieving the trade-off between simplicity and completeness. The clustering was applied with numerous numbers of components (124, 57, 28, 13, and 10). The result exhibits that 13 components were the suitable number that achieve the aforementioned two conditions where it covered more than 50% of the data and the accuracy of clustering was 90.39%. The clustering process and PCA showed the two algorithms could collaborate and characterize more than 85% of flows based on the network performance features. These outcomes significantly promote the results of the work in Chapter 3.

Elephant flow identification models either depend on flow-level features such as the total number of packets, count of bytes, duration, etc., or on packet-level features such as RTT, size of a packet, inter-arrival time, and direction of the packet. Each of these two techniques has pros and cons. Flow-level features-based models have proved a high accuracy of elephant flow identification, but they cannot detect elephant flows in their premature stage. In contrast, the models that are based on packet-level features might determine elephant flow in its early stage; however, it needs to define the application type that generates that elephant flow. In addition, the detection accuracy is a major concern in this kind of model.

The empirical results of the innovative flow identification mechanism based on flow-level features models were effective in identifying the flows as elephants and mice. Flow size, number of packets, and duration of flow features have been used in this experiment to distinguish between elephant and mice flows. On the other hand, the threshold that can be chosen for the mentioned features can be fixed or dynamic based on the traffic, or the top number of flows based on the traffic share, however, there is no common guideline to choose the appropriate threshold in different scenarios [201]. Depending upon that the thresholding method revealed that the threshold values (10KB, 15, and 5 seconds respectively) for the parameters that have been chosen in this experiment, were efficient in identifying elephant and mice flows.

To evaluate the flow identification process, the Naïve Bayes classifier has been employed, as it is suitable in use in flow-based traffic classification in relation to the proposed mechanism. The 50% of the flows were used to train the classifier, and the remaining 50% of flows were used to test it. The Naïve Bayes classifier was applied to all each cluster resulting from the clustering process. The accuracy rate of identifying elephant and mice flows for all clusters was 92.7%.4.6

4.6 Conclusion

The introduced platform proposes a working concept to identify a particular flow based on its characteristics that in turn depend on the network performance metrics. The proposed platform is capable to identify elephant and mice flows through pre-decided thresholds. The platform includes differentiation between elephant and mice flows according to their characteristics, and calculation of the own delay of each flow based on its parameters to provide the best path for each type of flow by choosing the least delay path. The objective of the new method is to network performance improvement by managing flow traffic and providing equal use of network resources dynamically as the next chapter proposed.

Chapter Five: SDN routing framework based on flow identification

5.1 Introduction

The idea of software-defined networking (SDN) is introduced to provide the ability to control the pathing of traffic flow across the network. It has been employed to achieve sufficient management of flow and effective utilization of resources in the network. This is attained by developing more programmable control and routing techniques according to a comprehensive view of the network condition and fine-grain control of network traffic and resources [202]. An SDN has several advantages to support TE due to its distinct properties. These properties include control centralization, the separation between forwarding and control planes, and the programmability of network behavior [203][204].

The essential functionality, which affects the performance of the network, is flow routing. An obvious advantage of the routing process is accessing the data as fast as needed. This can contribute to improve the performance of a network [2][3]. In multipath routing, the SDN controller handles determining the best path and substituting the link failure. However, the time consumed for selecting the best path by the SDN controller is still high [23].

SDN is indeed the optimal vehicle to control and prioritize the respective types of traffic according to their needs, as it decouples the network devices in data plane from the traffic and its associated needs in the control plane [115][116]. Specifically, the data plane devices such as router and switches have a packet-forwarding responsibility while the control plane includes rules that are used by the devices of data plane to forward packets. Depending on the above, SDN is characterized by decoupled control and data planes and control plane programmability [117].

On the other hand, the data center is a network of computing and storage resources. It provides the delivery of applications and shared data. Routers, switches, servers, storage systems, firewalls, and controllers of application delivery are the key components of the data center. Recently, the infrastructure of data centers has changed from the traditional structure of physical servers to virtual networks [205], which support applications. Traffic in data centers consists of many latency sensitive flows "mice", which contain only a several packets, and a few of the bandwidth-sensitive 'elephant' flows that comprise more than 80% of the total load [17]. In general, 'mice' flows induce transient congestion, while 'elephant' flows cause constant congestion where the congestion of the network is one of the main inhibitors of its performance [15]. As a result, not all flows use the network resources equally. There are many methods for providing the good network performance and high QoS to flows such as mice flows prioritizing or re-forwarding elephant flows [7], [143], [144], [206]–[209].

This chapter presents an innovative SDN routing framework based on flow type identification to find the best path using a number of developed algorithms. In addition, it takes into account leveraging the network performance features in flow characterization and flow type identification by employing unsupervised ML techniques. The framework has been conducted on different network topologies. The effectiveness of the framework has been evaluated through two experiments and by comparing its performance with that of the Ryu controller according to different factors. The main contributions of this chapter are as follows:

1. Characterizing traffic flows in the data centers using network performance features leveraging Unsupervised ML techniques.
2. Proposing an identification mechanism for distinguishing flows as mice or elephants based on their performance features.
3. Developing a unified architectural flow routing solution that integrates Unsupervised ML with SDN.

4. Developing a route updating-based recursive process for enabling a more efficient calculation of route cost and providing consistency with the real-time constraint.

The rest of this chapter is organized as follows. Section 5.2 highlights the proposed SDN-based flow routing application. The results and analysis of the experiment are presented in Section 5.3 and Section 5.4 presents final Conclusions.

5.2 SDN-based Flow Routing Application

SDN-based flow routing application that optimizes flow routing based on network performance analysis will be presented in this section. This application starts with capturing OpenFlow traffic statistics from the SDN switch. Pre-processing and feature selection is the next step of the proposed mechanism. Then, for dimensional reduction and flow clustering, unsupervised ML techniques were utilized . Particular parameters and thresholds have been pre-defined to identify flows as elephant and mice. Finally, Two topologies of Data Centre Network (DCN) have been used for SDN deployment. The proposed methodology is depicted in **Error! Reference source not f ound.**

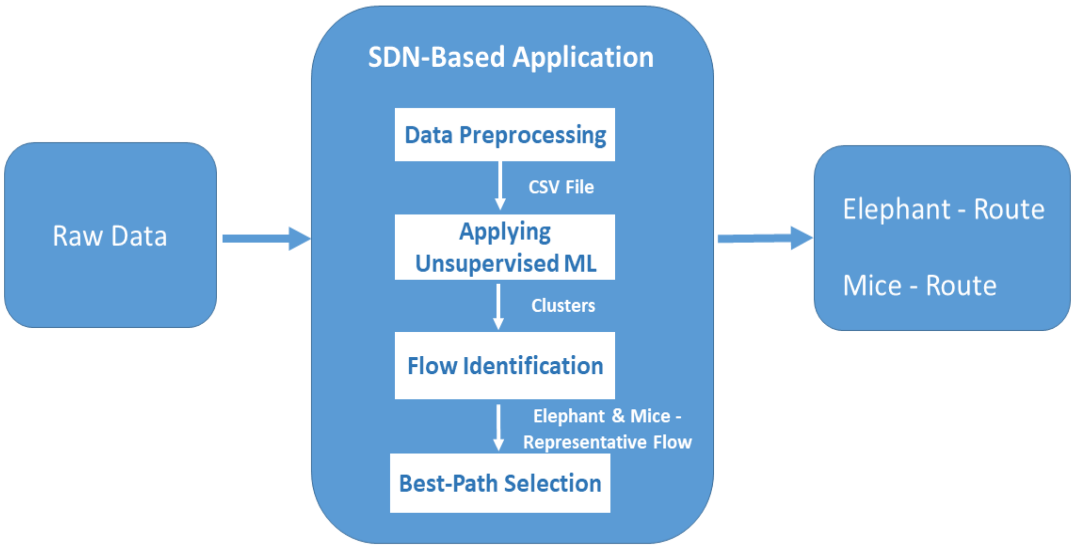


Figure 5.1 Proposed Methodology

5.2.1 **The Proposed Framework**

We advocate that selecting the best route based on flow type can take advantage of the programmability offered by SDN/OpenFlow. To illustrate this, a framework that is competent for identification and routing flows is presented. This framework enables the administrator to use the operation of selecting the best route based on flow type to improve the performance of the network. The operation depends on specifying the appropriate parameters and integrating the unsupervised ML and SDN environment. A conceptual graph of the framework is shown in Figure 5.2. The paradigm contains four intelligible blocks as below:

1) The Data Center (DC) Network topology.
2) SDN controller.
3) External application.
4) Traffic analysis.

To test the proposed application, SDN-based two different DCN topologies have been used. Mininet emulator [210]–[212] was used to implement the proposed paradigm. Figure 5.3 and Figure 5.4 present the two DCN topologies that comprise two servers for each topology and different numbers of switches. In this study, SDN has been deployed utilizing the SDN-Ryu controller [213]. The external application was designed for flow routing optimization that depends on a statistical analysis of network performance. To achieve this, the application includes three stages:

A. Characterizing the flows based on network performance metrics by employing unsupervised ML. this stage starts with using Principal Component Analysis (PCA) as a linear technique to reduce the dimensions of the used dataset. By converting correlated features to uncorrelated features, the dataset will be reduced from high-dimensional space to low-dimensional space. To cluster the flows based on their own features of network performance, K-means was used as a second unsupervised ML technique.
B. Identifying flow type. This stage is the process that is responsible for determining the type of flows (elephant or mice) based on pre-defined parameters and thresholds. The procedure will be applied to each cluster that resulted from the clustering operation.

C. Selecting the best path for each flow according to its type and characteristics. The stage starts with a sampling operation to gain representative flows for each cluster. Then using a developed routing algorithm, the best path selection will be achieved for each representative flow.

The proposed application employs two algorithms, which are the shortest path Dijkstra algorithm [214] and the widest path Dijkstra Algorithm [215] to build the developed routing algorithm. It has been built to find the route that fulfills the conditions related to the types of flow identified before. The developed algorithm was employed to find paths with appropriate bandwidth and latency.

The last part of the proposed framework is the traffic analysis. In this part, the flows with their paths will be stored and visualized. The output of this block is a log file containing the type of flow, cluster-ID, the paths of flow, and the latency for each route. The next section covers the implementation of the proposed framework by running a group of proposed algorithms.

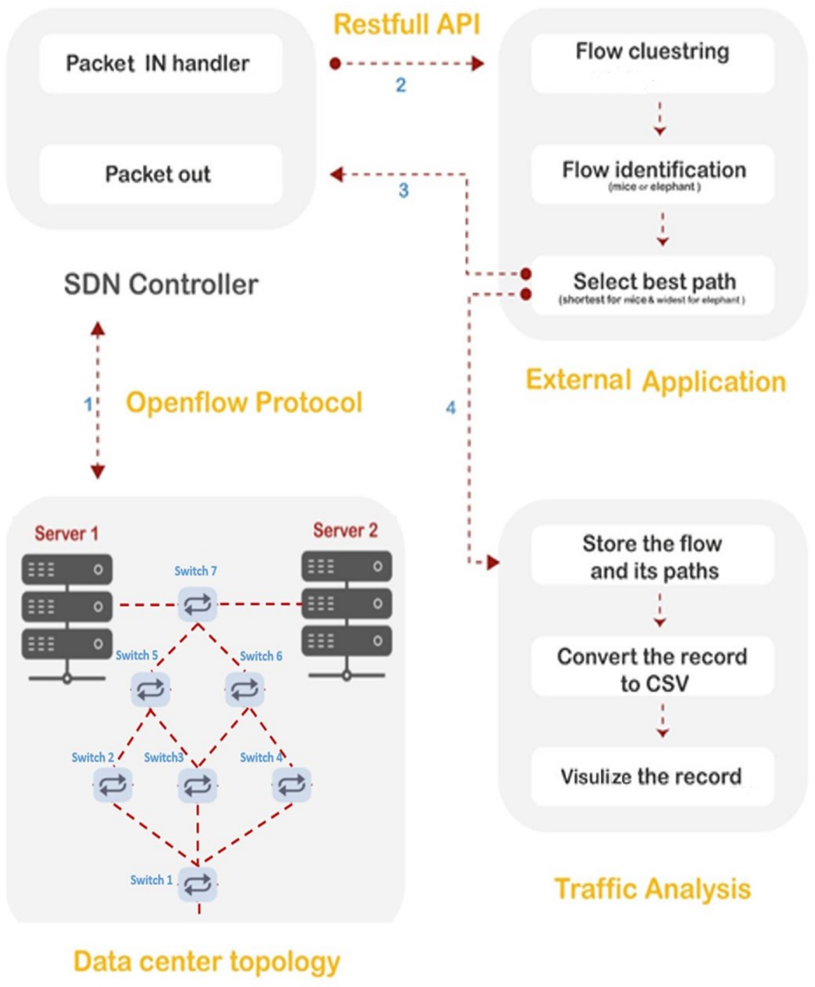


Figure 5.2 The Proposed Framework

5.2.2 **Framework Implementation**

In this section, the implementation of the innovative framework will be clarified. In particular, the main algorithms and their roles in selecting the best path are explained.

5.2.2.1 *Routing rule setting*

The details of the flow routing algorithm are illustrated in algorithm 1. This algorithm aimed to select the best path to route each flow, which is the major procedure of the proposed framework. The procedure starts with extracting datapath, inPort, packet, source, and destination. At first, the existence of the source in the macTable will be checked. Then it will be added to the macTable if it is not there. Thereafter, the algorithm checks whether the destination exists in the macTable or not. In case of the destination is in macTable, a topology discovery will be executed using a Link Layer Discovery Protocol (LLDP). Next, Algorithm 2 FAS (Find all Available paths between two Switches) will be implicitly called by Algorithm 3 FAPBS (Find all Available Paths Between each pair of Switches) to find all available paths between each pair of switches. After that, the bestPath is determined based on clusterID, flowType, and acceptable latency, and in a recursive manner, the costs of paths will be updated. If a destination does not exist in macTable, a broadcast message will be sent.

Algorithm 1 Flow Routing Algorithm

```
1.Every new flow,extract flow information (src, dst, inport, outport, etc.)
2. adjacency ←Ø
3. availableBW  ←Ø {availableBW:available bandwidth}
4. paths  ←Ø
5. clusterID ←Ø
6. flowType ←Ø
7. acceptableLatency  ←Ø
8. bestpath  ←Ø
9. if src not in macTable, then
```

```
10. add src to macTable
11. end if
12. if dst in macTable, then
13. check topology using LLDP to get adjacency and availableBW
14. paths ←FAPBS(adjacency, availableBW)
15. clusterID, flowType, acceptableLatency ← flowIdentification(kmeansModel, flow)
16. bestpath  ←SelectBestPaths(src, dest, acceptableLatency, flowType, paths)
17. paths ←updatedPaths(paths, availableBW)
18. else
19. send broadcast message
20. end if
```

Algorithm 2: Find all Available paths between two Switches (FAS)

```
1.source: source switch
2. destination: destination switch
3. adjacency: adjacency matrix represents the network as a graph
4. paths: set of available paths between source and destination
5. paths ←Ø
6. QueOfPaths ← Queue()
7. QueOfNode ← Queue()
8. QueOfNode.add(source)
9. for each len(QueOfPaths) > 0 do
10. currPath ←QueOfPaths.pop()
11. lastNode ←currQue[-1]
12. if lastNode matches destination then
13. paths.append(currPath)
14. end if
15. for each neighbor in adjacency[lastNode] do
16. newPath ←copy(currPath)
17. newPath.add(neighbor)
18. if !QueOfPaths.contains(newPaths) then
19. QueOfPaths.add(newPath)
20. end if
21. end for
22. end for
23. for each availablePaths in paths do
24. for each path in availablePaths do
25. cost ←Update Cost(path)
26. path.addLast(cost)
27. end for
```

```
28. end for each path in availablePaths do
```

Algorithm 3: Find all Available Paths Between each pair of Switches (FAPBS)

```
1. adjacency: adjacency matrix represents the network as a graph
2. availableBandwidth:2D array contains available bandwidth between each two switches
3. Paths: set of available paths between source and destination
4. Paths ←Ø
5. for each node1 in adjacency do
6. for each node2 in adjacency do
7. if node1 does not match node2 then
8. p ←FAS(node1,node2,adjacency)
9. paths{'node1TOnode2'} ←P
10. end if
11. end for
12. end for
13. Paths ←Update Cost(paths, availableBandwidth)
```

5.2.2.2 *Cluster vector extraction*

The output of the K-means model will be used as input for the flow identification process (Algorithm 4). For each resulted cluster, the flow identification will be implemented. Accordingly, three parameters were extracted for each flow in each cluster. These parameters are flow duration, packet count, and average packet size. In the case of having packet count higher than 15, flow duration higher than 5 seconds, and average packet size higher or equal to 10KB, the flow is predicted as an elephant. Otherwise, the predicted flow is mice. At the end of this process, three variables were obtained. clusterID, which represents the ID of the predicted cluster, flowType that is a string indicating a mice or elephant flow, and the acceptable delay of the flow is represented as a float number called acceptableLatency. Based on the above, the clusterVector will be assigned. This vector represents the features generated from the center of the cluster called representative flow. From each representative flow, the value of the acceptable latency is calculated.

Algorithm 4: Flow Identification

```
1. PPR: set of rules for predefined parameters
```

```
2. CID: set of predicted clusters (kmeansModel)
3. FT: a string indicates mice or elephant flow
4. CF: set of current flows
5. RF: set of representative flows
6. AL: a float number represents the acceptable delay of cf 2 CF
7. FT ←Ø
8. RF ←Ø
9. AL ←Ø
10. for each cid ∈CID do
11. for each cf ∈CF do
12. for each ppr ∈PPR do
13. if cf.packetCount > 15 and cf.flowDuration > 5 and cf.avgPacketSize >=10 then
14. FT ← 'elephant'
15. else
16. FT ← 'mice'
17. AL ←cf.latency
18. end if
19. end for
20. end for
21. RF ←kmeansModel.lefts{cid}
22. end for
```

5.2.2.3 *Latency and bandwidth measurement*

The source, destination, flow, acceptableLatency, paths, and N will be used as the input for algorithm 5 where N determines the number of paths to be selected. This algorithm starts with checking the type of flow. If it was mice, the algorithm will sort the available paths with respect to length and select the shortest path as the best. Otherwise, it was elephant flow and the best N paths will be chosen based on the 1/bandwidth, as shown in Sub-section 5.3.1.6. As a result, for each type of flow, we have the best path based on acceptable delay and bandwidth for the particular source and destination. After all, the particular best path will be installed for each elephant and mice flow. This is done by sending a message containing the information of the selected path to SDN controller to determine the required switches. Finally, an updated version of input paths contains each path associated with the cost of it. The process of updating will be based on the paths, which is the output of FAPBS algorithm and availableBandwidth as a 2D array containing available bandwidth between every two switches. This is summarized in algorithm 6.

Algorithm 5: Select best paths

```
1. source: source switch
2. destination: destination switch
3. acceptableLatency: acceptable delay
4. FT: a string indicates mice or elephant flow
5. paths: set of paths between each two switches
6. N: integer, determine number of paths to be selected
7. RT: routing table
8. N ←Ø
9. cost ←Ø
10. availablePaths ←paths{'sourceTOdestination'}
11. if FT matches 'mice' then
12. path ←availablePaths.sort(key = len, ascending = False){0}
13. Else
14. path ←availablePaths.sort(key = cost,ascending = False){0 N}
15. end if
16. RT ←path /*path installing*/
```

Algorithm 6: Update Cost

```
1. Paths: set of output of FAPBS algorithm
2. key:(src,dst)
3. availableBandwidth: 2D array contains available bandwidth between each two switches
4. updatedPaths: updated version of input paths contains each path associated with the cost
   of that path
5. updatedPaths ←Ø
6. for each key in Paths.keys() do
7. updatedPaths{key}←Ø
8. for each path in Paths{key} do
9. cost ←0
10. for each node1, node2 in path{:-1}, path{1:} do
11. cost ←cost + availableBandwidth{node1}{node2}
12. end for
13. updatedPaths{key}.append({path,cost})
14. end for
15. end for
```

5.3 Results and Analysis of Experiment

The results and analysis of this experiment demonstrate how feasible the proposed approach at calculating the cost of links and selecting the best path in the SDN environment. The experimental design is provided in Sub-section 5.3.1 and the results and evaluation are presented in Sub-section 5.3.2.

5.3.1 **Experimental Design**

The experiments have been conducted using the Mininet emulator. Two DCN topologies. The first topology (No.1) consists of two servers, seven switches, and two hosts, illustrated in Figure 5.3. Whereas the second topology (No.2) includes two servers, 16 switches, which are distributed on five layers, and three hosts as shown in Figure 5.4. The work was carried out by a group of phases as below.

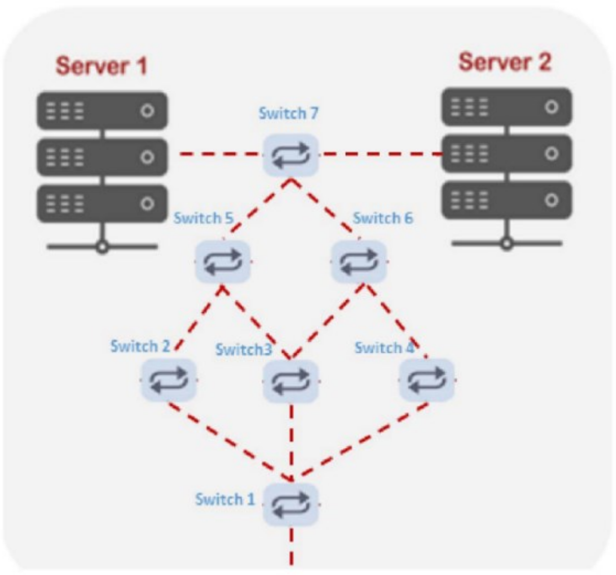


Figure 5.3 Topology No.1 of Network

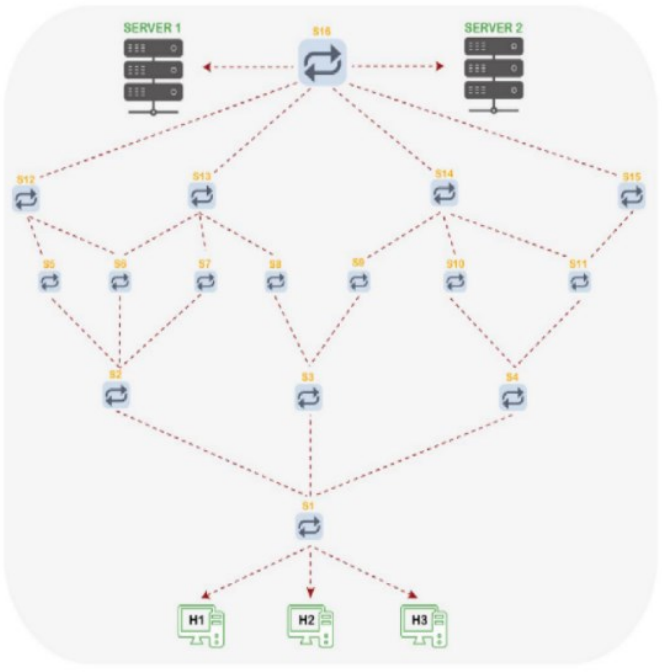


Figure 5.4 Topology No.2 of Network

5.3.1.1 *Monitoring and Data Collection Phases*

For improving the comprehensive performance of the network and traffic flow optimization, which is the aim of this work, traffic monitoring is very vital. SDN presented a dynamic monitoring scheme for network traffic by adopting the concepts of centralized controllability, the scalability of network infrastructure, usable, and programmable. In this experiment, the Mininet emulation paradigm was used for setting up and installing an SDN environment with the parameters of simulation as in Table 5 1. Two scenarios (two topologies with different settings) for the DCN have been emulated for implementing the proposed method. The used topologies are configured for handling the TCP and UDP flows. These flows are generated using a virtual machine created by the VMware workstation in a Linux environment. OpenFlow Wireshark and tcpdump tools have been used with the Mininet simulator to understand the behavior of proposed DCN topologies for analyzing the performance of the networks. In order to perform a flow background load on the network, the work started with generating UDP packets with different packet sizes and rates during 300 seconds of simulation as shown in Table 5.2. Multiple TCP connections (parallel flows) were initiated at different times to raise the throughput and improve the performance between the two hosts. The value of bandwidth for each link was generated randomly between 30 Mbps and 1000 Mbps. Here, for a single stream of TCP, the average throughput is calculated as [216]:

$$\boldsymbol{average\ throughput \approx \frac{MSS}{RTT\sqrt{PL}}\ bytes\ per\ second} \qquad \textbf{5.1}$$

The MSS value represents the maximum segment size and PL represents the rate of packet loss. For multiple parallel streams(X) and if RTT, PL, and MSS are the same in each stream, the aggregate average throughput is the sum of the X average throughput [217]. Figure 5.5 illustrates the behavior of topology No.1, which uses Ryu controller, according to the throughput parameter with the absence of the proposed application.

Table 5.1 the parameters setting for the SDN network simulation

Parameters	Details	Descriptions
Operating System	Mininet 2.2	Default behavior; idle timeout 30s; traffic monitor polling 30s
SDN controller	Ryu controller v3.3 (Equal Cost Multipath routing algorithm(ECMP))	
OpenFlow software	OpenFlow: v1.3 Open vSwitch: v1.3.1	

Table 5.2 UDP packets generated during 300 seconds

Duration (Seconds)	Packet rate(pps)	Data rate (bps)
0-30	250	1024
30-60	500	2097152

60-120	1000	12582912
120-200	1000	14582912
200-260	1000	20971520
260-300	1500	20971520

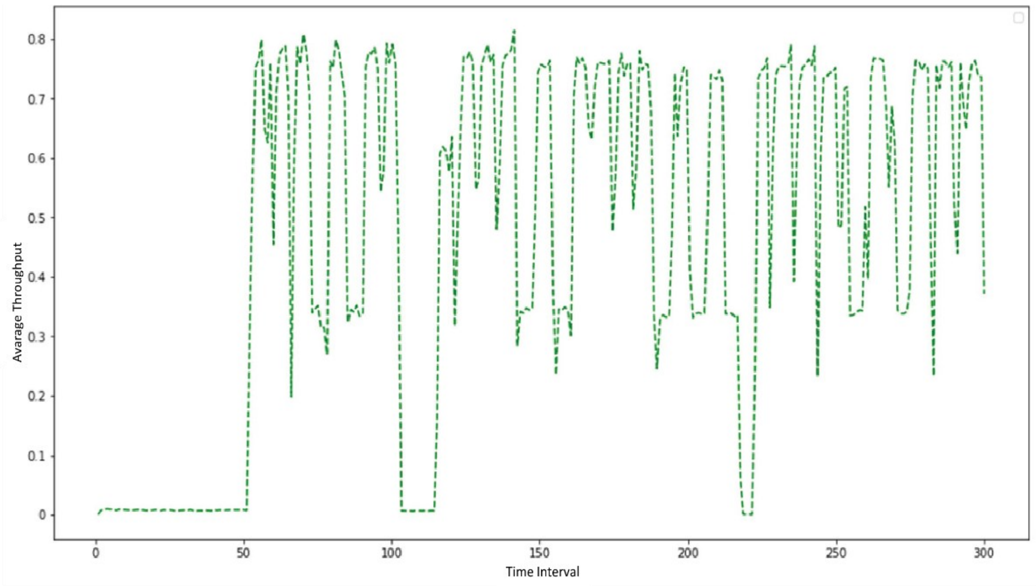


Figure 5.5 Throughput of the Network during 300 Seconds

Wireshark has been run in the background to capture OpenFlow packets, which are TCP packet types. Then the tcpdump tool has been utilized for the loopback interface. The collected data was stored as pcap files for the two proposed DCN topologies. To undertake a comprehensive analysis, the tcptrace tool was employed to produce all complete flows characterized based on performance parameters and store them in a CSV files.

5.3.1.2 *Pre-Processing and Feature metric Phase*

This phase is essential to obtain consistent, integrated, and processable data by machine learning techniques. The data provided by the monitoring tools is of great benefit not only for direct use, but historical data can help academics in understanding the behavior of networks. The tcptrace tool aims to aggregate and analysis the required information across the network to introduce data in a consistent and understandable form. It stores data in a CSV file format. Here, 146 parameter values were produced by the tcptrace tool. These parameters are related to bi-directional complete connections and identify the most significant discriminators of the traffic to determine which discriminators are suitable for traffic classification and how classification accuracy can be improved. In this work, the main data preprocessing steps are similar to those that have been applied in the previous work in Chapter 4. Figure 5.6 explains these steps. 129 parameters resulted after applying data cleaning and data transformation steps. Data reduction is another step of preprocessing. PCA was utilized with 13 components in this experiment. The results proved the validity of the findings of the preprocessing in the previous work in Chapter 4 by achieving the balance between completeness and the simplicity and accuracy of the clustering process.

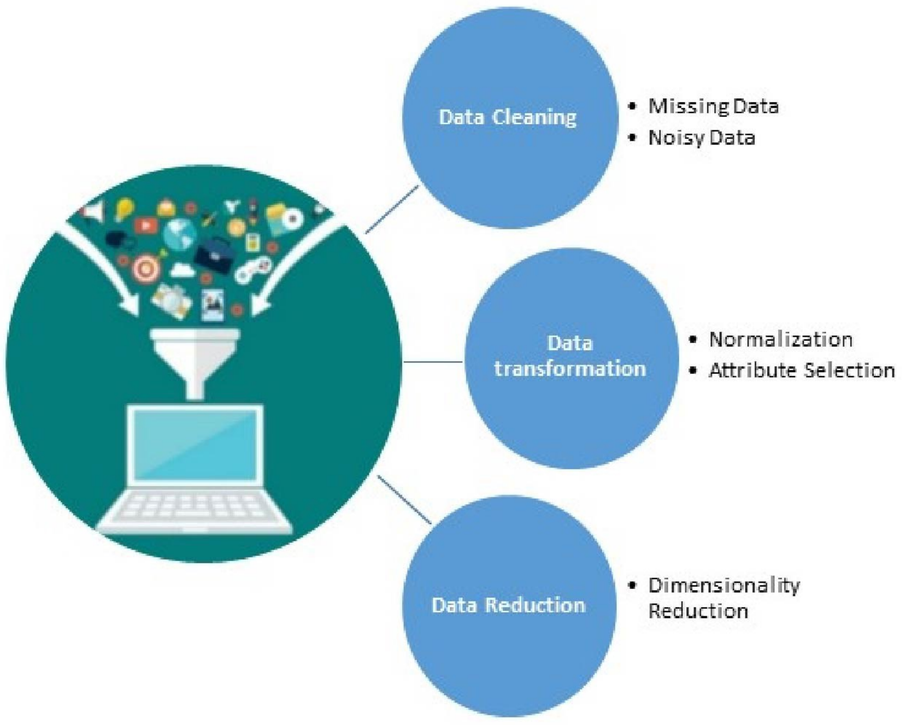


Figure 5.6 The Steps of Preprocessing Phase

5.3.1.3 *Flow statistics-based Clustering Phase*

Recently, ML techniques have overcome some of the heuristic solutions' limitations by enabling new classification methods. Flow-statistics-based classifiers are a popular group of these methods [25], [29], [34], [37], [38], [40], [218], [219]. These classifiers have been adopted because it shows high speed in feature computation and classification. Here, unsupervised ML techniques have been proposed to develop a flow statistics-based characterizing mechanism using the metrics of network performance. One of these techniques is K-means clustering algorithm, which can create clusters of flow without the need for predefining classes. This algorithm has been utilized to understand the differentiation of flows with respect to network performance features. It has been applied to construct three clusters, where the minimum number of clusters has been experienced in this work. The goodness of clustering has been measured by using the accuracy metric. This metric is determined by two factors: inter-cluster distance and intra-cluster distance [197]. As a result, unique samples of flow will be recognized for each cluster.

5.3.1.4 *Flow Type Identification Phase*

One of the serious problems that have affected the quality of service for mice flows is a slow-down transfer caused by elephant flows through a network, which leads to the degradation of network performance. The solution for this problem starts with proposing a novel identification mechanism for distinguishing the flows as elephants and mice by leveraging some of the parameters clarified in the previous Sub-section 5.3.1.4. Each elephant or mice flow has been defined based on static threshold values marked on pre-decided features using the thresholding method. The values of a threshold may be static or dynamic. This depends on the traffic or top-N of flows, where N refers to the top number of traffic as mentioned in Section 4.5. The final step in this phase includes the extraction of a representative flow for each elephant and mice flows in each cluster that results from the clustering process. It is interesting in the step that in some clusters one or two representative flows can be extracted for each elephant or mice flow.

5.3.1.5 *Flow Type-based Path Selection Phase*

Normally, a large number of servers and switches are included in DCN, and each node has multiple flows. Furthermore, the performance of the DCN is a critical aspect; hence, high throughput and sensitivity of packet loss are required to gain high performance of the network. And in accordance with the aforementioned problem in Sub-section 5.3.1.4, accessing the data with reliability and in a simple way is extremely hard in DCN [220][208]. Therefore, the need to manage the traffic in this type of network still exists as clarified in the literature review Sub-section 2.7.3. Many routing algorithms can provide a management solution for elephants and/or mice flow. For example, Equal Cost Multipath Routing (ECMP) [221] can be utilized for routing mice but not elephant flows [150]. In this project, the major aim is to select the best path for each elephant and mice flows. The best path for a flow is the path that achieved the requirements of that flow like low latency for mice flows and high bandwidth for elephant flows. A developed Dijkstra algorithm is proposed to find the route based on the type of flow (elephant or mice flows) by employing algorithms 3, 4, 6, and 7 as shown in Section 5.2.2. Using a developed Dijkstra algorithm, the links that fulfill the conditions will be determined, and the paths that contain appropriate bandwidth and latency will be selected. Dijkstra algorithm was applied to find the shortest path for mice flows whereas, for elephant flows, the Widest-Dijkstra algorithm was used to find multipath to route them and record all the available shortest paths. Because it has an appropriate complexity in real-time problems and it gives deterministic results, the Dijkstra algorithm has been chosen. The proposed approach introduces a new calculation for the cost of a link as compared with the one used in the Dijkstra algorithm. In this part of the work, different topologies based on SDN have been used with improved utilization of the bandwidth and reduced network congestion. The deployed DCN topologies are depicted in Figure 5.3 and Figure 5.4.

5.3.1.6 *Link-Cost Updating phase*

Determining a route for a particular flow through SDN efficiently is still challenging. In order to accomplish this, a lot of information must be obtained by the SDN controller. the information includes getting a comprehensive vision of the network (i.e. network topology discovery and getting link state information), computing optimum paths for the flows regarding the information of the flow and network, and reconfiguring the routing table based on the new forwarding rules in the infrastructure plane. The common routing algorithms depend on three concepts to compute the cost of the link. These concepts are static link-cost (Hop-to-Hop count, distance, link-capacity), dynamic link-cost (available link-capacity, link utilization), and dynamic link-cost with minimizing the interference (available link-capacity, link-utilization, flows count on a link) [222]. The proposed approach presented in this part explains calculating and updating the cost of the link in the SDN framework. The proposed method computes the cost of the link based on the inverse proportional relationship between bandwidth and latency of the link, where latency=1/bandwidth. The cost of the link will be updated recursively. The calculation of the cost of the link based on the Update Cost algorithm in Sub-section 5.2.2 with an example and figures are explained in this regard.

Suppose a host (Host) want to send a set of packets to a server (Server) and there are two different routes between Host and Server as in Figure 5.7. For more explanation, the flowchart in Figure 5.8 and the steps below will be followed to calculate link-cost of both routes:

1. 1st route R1 goes through switches S1- > S2- > S4- > S6- > Server
2. 2nd route R2 goes through switches: S1- > S3- > -S5- > S6- > Server
3. The getAvailablePaths (Host, Server) algorithm will find these two paths (R1, R2) between Host and Server
4. For each link in R1, R2
5. B1 =Available bandwidth of R1 = availableBW(L12) + availableBW(L24) + availableBW(L46)
6. B2 =Available bandwidth of R2 = availableBW(L13) + availableBW(L35) + availableBW(L56)
7. Latency of R1 = D1 = 1/B1; Latency of R2 = D2 = 1/B2.
8. If D1–acceptableLatency > D2–acceptableLatency then bestCost = D2
9. else bestCost = D1
10. Then we get the nearest value to the acceptable Latency
11. If bestCost = D1 then best path is R1 else best path is R2

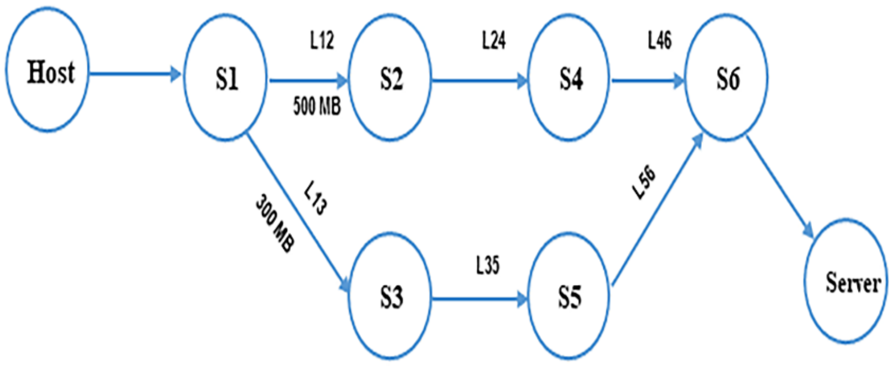


Figure 5.7 Updating Link-Cost

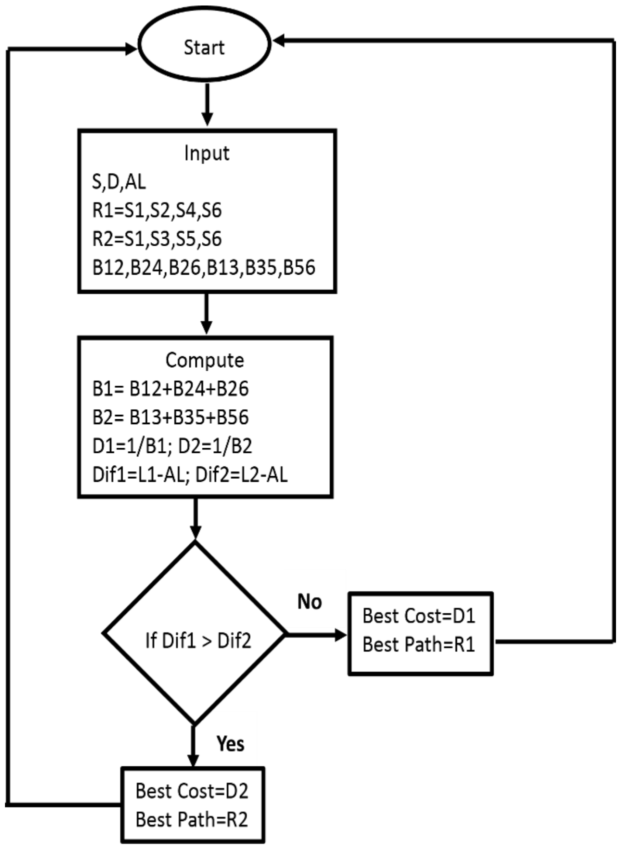


Figure 5.8 Flowchart of Updating Cost of Link

5.3.2 **Results and Evaluation**

The empirical results of this experiment determine the impact of the differentiation between elephant and mice flows using the innovative routing algorithm upon DCN performance. In general, the results of this experiment demonstrate the proposed characterization mechanism can be employed to identify the type of flow to facilitate the election of the path that fulfills the requirement of each of them. This experiment is performed using a number of python programming scripts written and generated on a Linux (Ubuntu 18.04) 64-bit Operating System with Intel Core i7-1165 G7, 2.8 GHz, and 16 GB RAM.

The experiment was conducted using a CSV file for each suggested DCN topology. Each file contains 1 million flow records and 129 features. These features are chosen based on the work in chapter 3. For dimensionality reduction, PCA has been applied to select the most relevant features (13 components out of 129 features) from a whole dataset. These 13 PCA components have been input into the clustering process. A possible explanation for choosing this number of components might be found in Chapter 4. The k-means-based clustering explains that all flows were only obtained in three clusters. Each cluster contains a group of flows with distinct types and characteristics. Another important finding is in every cluster one or two representative flows have been extracted for each elephant and mice type. Finally, the best path for each type of flow will be found based on that representative feature vector. A developed Dijkstra algorithm was utilized for that purpose. A sample of elephant and mice flows with a brief description of the work are provided in Table 5.3.

Table 5.3 Traffic Routing Based On Flow Types

Number of Flows	Flow Type	Port Number	Protocol Service	Packet Rate	Method to Apply
60	Elephant	88	HTTP	1M	Multipath
17	Elephant	443	HTTPS	300K	Multipath
40	Elephant	20	FTP	22K	Multipath
26	Elephant	25	SMTP	12K	Multipath
9	Mice	514	Syslog	632K	Single path
14	Mice	88	Kerberos	220K	Single path
6	Mice	119	NNTP	125K	Single path

Figure 5.9 depicts how the route was elected for each type of flow in all three clusters for network topology No.1. Based on the proposed routing algorithm, the best path has been selected from the available paths on this network. As given in the top left corner of the figure, multipath route has been selected from Host A to the server for elephant-type flows based on their characteristics in cluster 1. The same procedure was followed for clusters 2 and 3. In the same way, the shortest path for all mice flows in three clusters has been selected as represented in the bottom right

corner of the figure. On the other hand, higher priority flows such as real-time traffic have been taken into consideration to be saved in this scenario. The figure clarifies the used links with a green dotted line, whereas the unused link with a red dashed line.

To achieve the aim of implementing the proposed flow routing optimization method and measure its effectiveness in SDN environment, we implement the proposed framework as described in Section 5.2.1. According to this, different experiments were conducted to evaluate the performance of the proposed approach against that of the SDN-Ryu controller. Furthermore, the evaluation will reveal the impact of routing-based traffic management, which depends on the type of flow and link-cost computation, on the network performance.

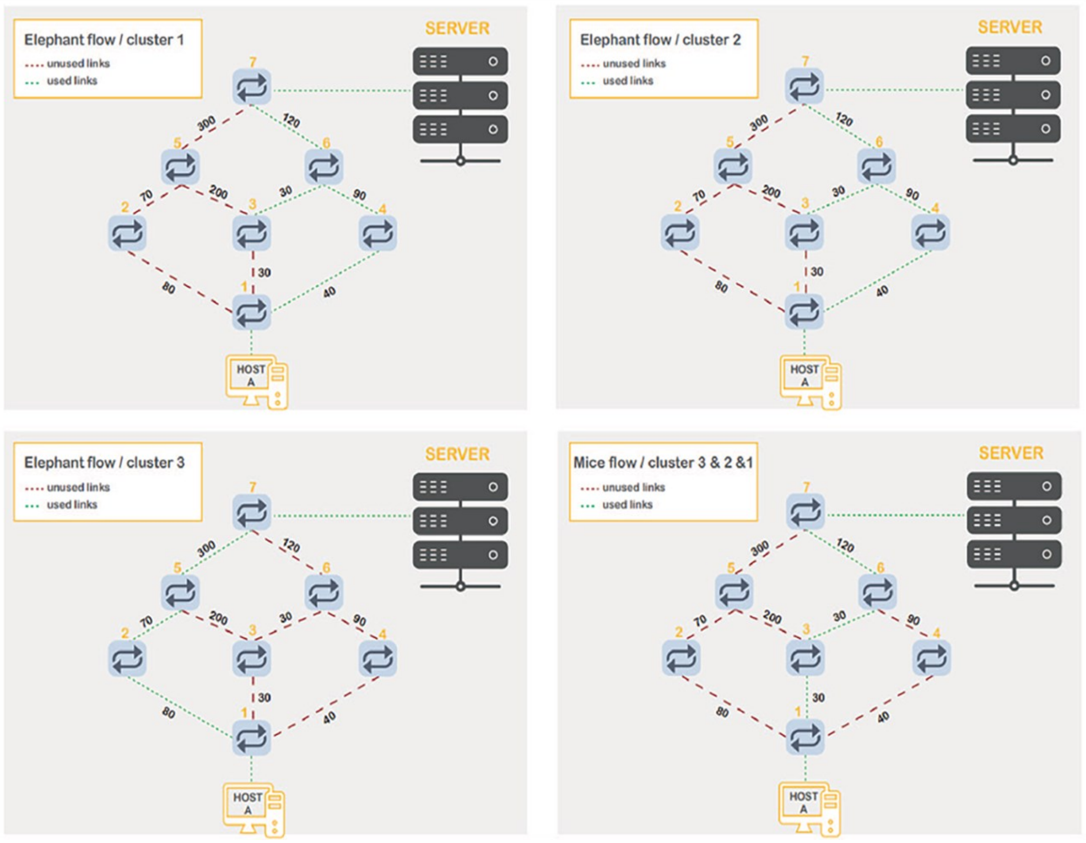


Figure 5.9 An Example of Flow Path's in the Network Using Proposed Approach

The first experiment that carry out to compare the performance of the proposed application with that of Ryu controller using two metrics or parameters throughput and bandwidth usage. As is shown below in Figure 5.10, for all flows the proposed application has provided higher throughput than the Ryu controller has. For example, at flow 10, the throughput of the proposed application was improved with a ratio of 61.5% compared with the throughput of the Ryu controller. In addition, it has been observed that the number of parallel flows in inversely proportional to the throughput of the network. This confirms that increasing the higher the number of parallel flows leads to a decline in the throughput for both the proposed application and the Ryu controller. However, the performance of the proposed method is still better than the Ryu controller. The comparison of throughput was done for the two types of flows.

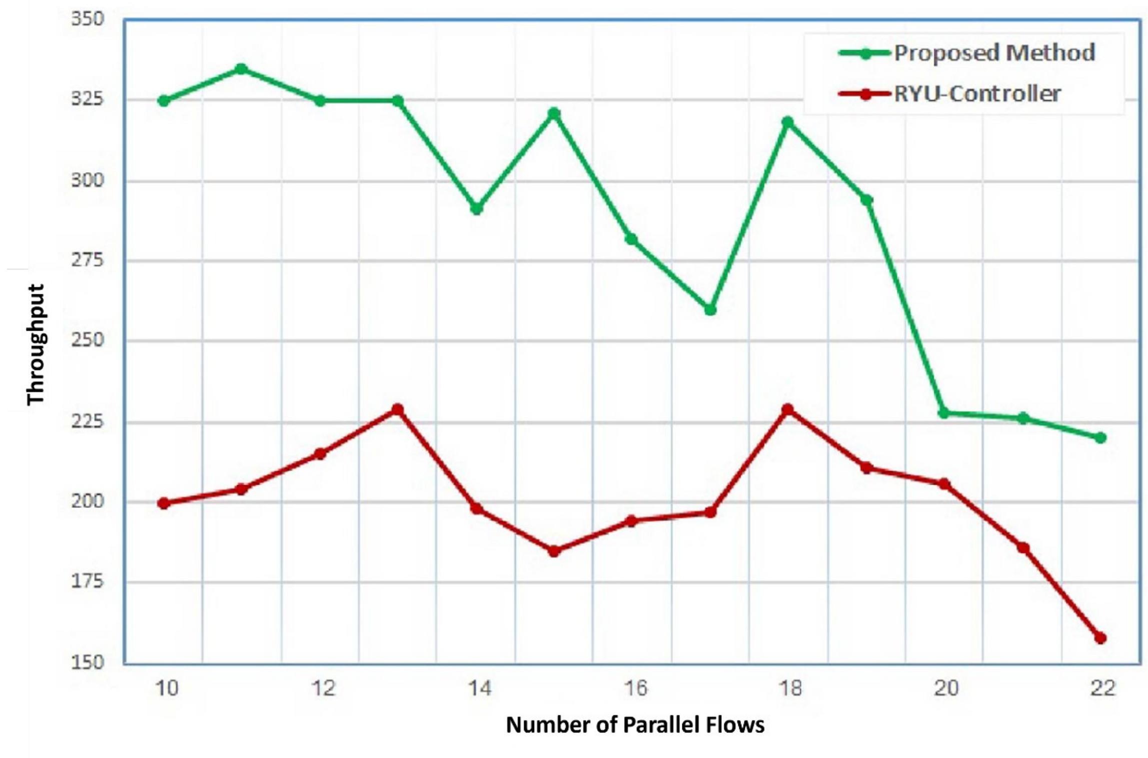


Figure 5.10 Comparison between the Proposed Approach and RYU-Controller with Respect to Throughput for Two Types of Flows

For elephant-type flows, the findings show that the throughput provided by the proposed application is outperforming that of the Ryu controller as depicted in Figure 5.11. For instance, it was found that the throughput of the proposed application is about 88.2% superior to that of the Ryu controller for flow number 22. The measurement of elephant throughput has been executed for intervals of 0-40 seconds and a bandwidth of 100MB.

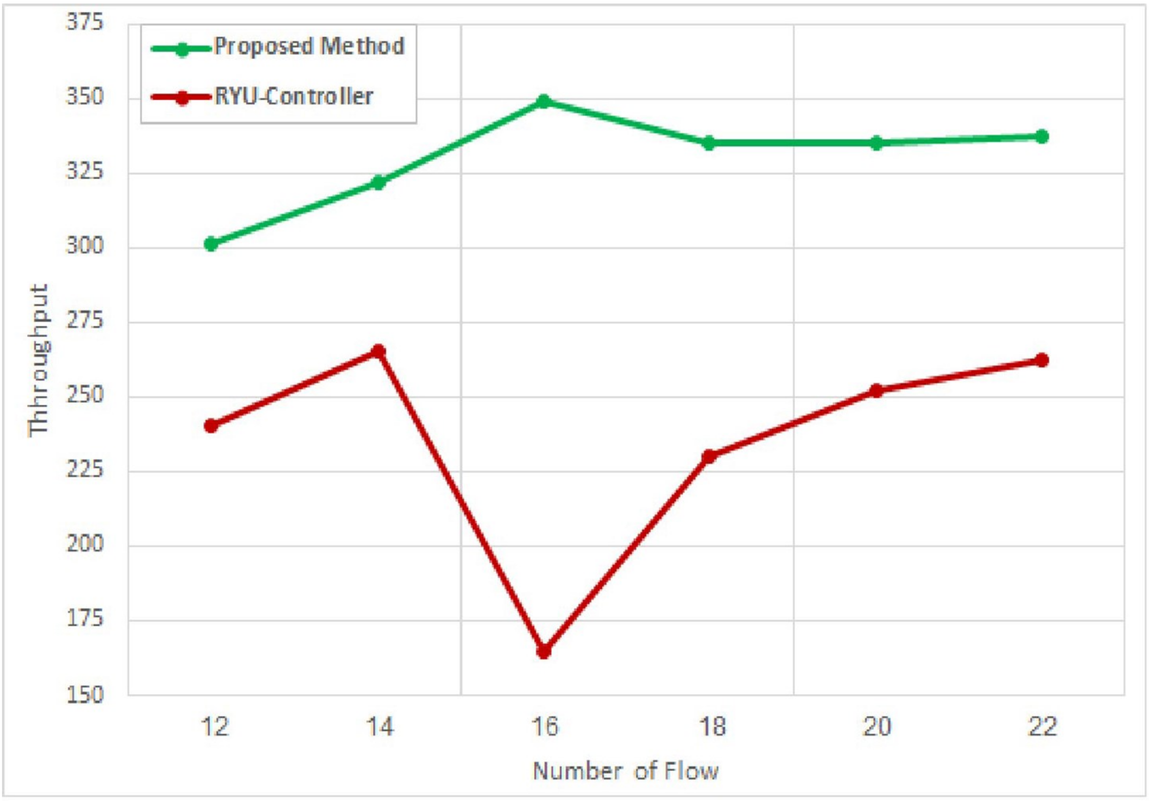


Figure 5.11 Comparison between the Proposed Approach and RYU-Controller with Respect to Throughput for Elephant Flow

Regarding mice-type flows, Figure 5.12 shows that both the proposed application and Ryu controller have provided the same throughput. It is interpreted by the fact that both of them use the shortest path for routing the mice flows. The measurement of mice throughput was run for intervals of 0-60 seconds with a bandwidth of 100 KB.

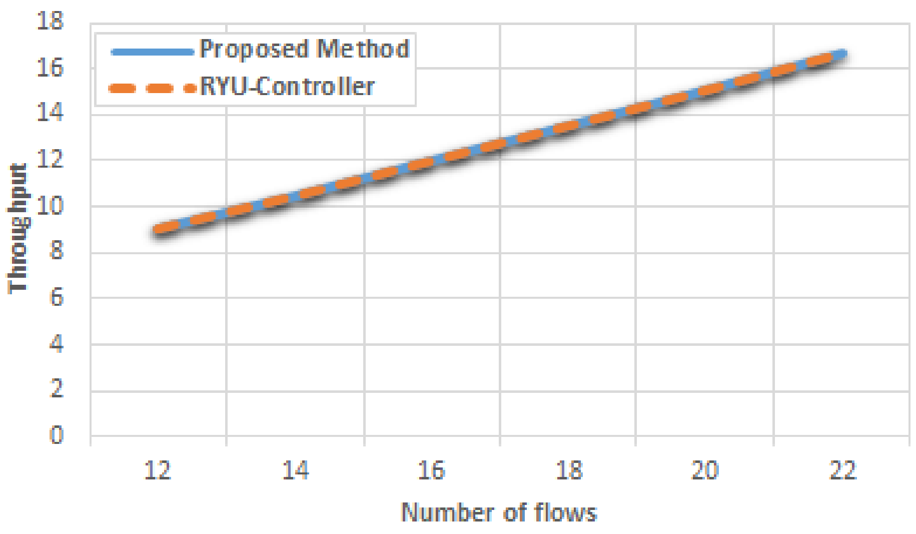


Figure 5.12 Comparison between the Proposed Approach and RYU-Controller with Respect to Throughput for Mice Flow

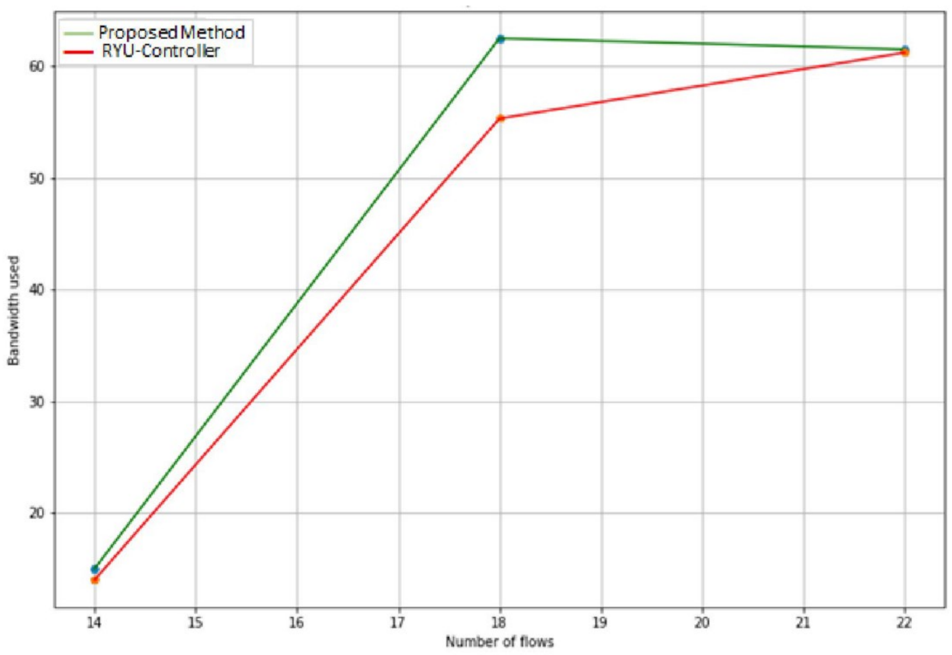
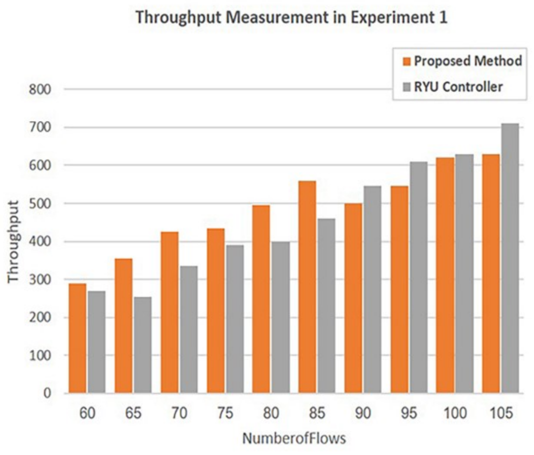


Figure 5.13 Comparison between the Bandwidth Used in the Proposed Approach and RYU-Controller for Two Types of Flows

For bandwidth usage, the experimental findings show that the performance of the proposed application is high in most of the flows compared with the Ryu-controller as illustrated in Figure 5.13. The reasons behind that are (i) using a clustering process and (ii) using a developed Dijkstra algorithm. While the low performance of the proposed method in flows 14 and 22 is because of serving higher priority flows.

For the second topology, the proposed mechanism evaluation was accomplished by running two experiments in order to (i) Compare the performance of the Ryu controller to that of the proposed method. (ii) Compare the performance of the proposed method in both experiments. (ii) Compare the performance of the Ryu controller in both experiments. All the comparisons were based on the same parameters for the first topology. Figure 5.14 and Figure 5.15 depict the throughput and bandwidth usage measurement provided by the proposed mechanism and Ryu controller in the two experiments. It is clear from the charts of experiment 1 that the performance of the proposed method was better than that of the Ryu controller for the majority of flows for both parameters. However, as the number of flows increases, the performance of the proposed method is equal to or slightly less than that of the controller. In experiment 2 as shown in the charts of the aforementioned figures, the throughput, and bandwidth provided by the proposed mechanism were improved compared with those by the Ryu controller. Consequently, the performance of the proposed mechanism enhanced for 70% of flows. Nevertheless, compared to the performance of the Ryu controller, the performance of the proposed application has declined for the rest 30% of flows.

In general, it is observed that the performance of the two methods has improved as the number of flows increase in both experiments.



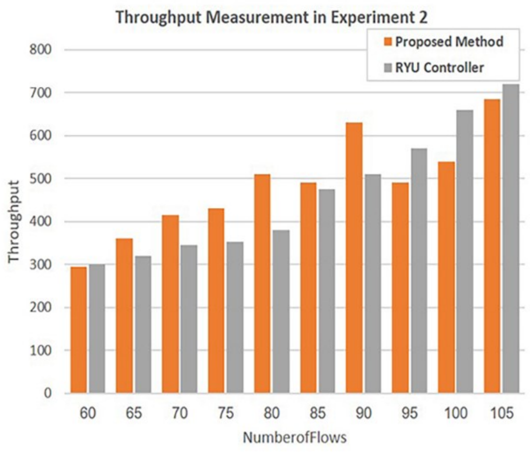


Figure 5.14 Comparison of Throughput between the Proposed Approach and RYU-Controller for Two Types of Flows in Two Experiments

The data transfer rate, which is the third parameter that has been used in these experiments, has been added to support the evaluation of the proposed method. Figure 5.16 shows that the proposed method was more effective in transferring data than the Ryu controller for most of the flows in both experiments. Based on the measurement of the rate of data transfer, the proposed method in the second experiment has the same behavior as in the previous parameters for the majority of flows, where the performance has become better than it is in the first experiment, although it may slightly less than the performance of the controller sometimes.

As a result, the proposed approach was more efficient than the Ryu controller and proved its ability to find the best route according to the type of flow and cost of the link despite the degradation of its performance because of serving the higher priority flows.

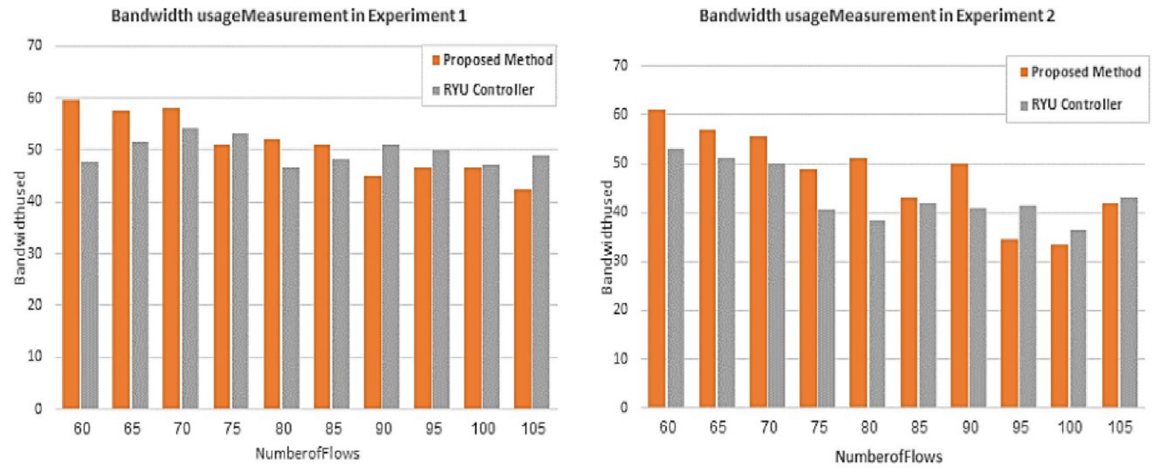


Figure 5.15 Comparison of Bandwidth Usage between the Proposed Approach and RYU Controller for Two Types of Flows in Two Experiments

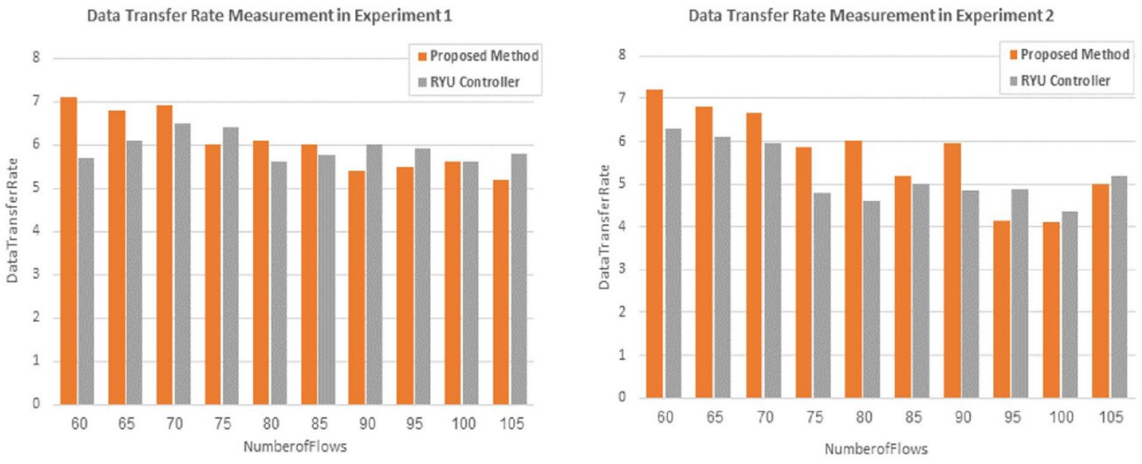


Figure 5.16 Comparison of Data Transfer Rate between the Proposed Approach and RYU- Controller for Two Types of Flows in Two Experiments

For more clarification, Figure 5.17 illustrates the throughput measurement for both the Ryu controller and the proposed method in two experiments. The plot presents descriptive statistics that there is obvious variability through the experiments that examined the throughput (speed) feature. For our method, there is a slight change in the median, first and second quartiles, while there is a significant change in the third and fourth quartiles for the two experiments. Similarly, the throughput of the Ryu controller significantly changes in the first and third quartiles, while it shows stability in the median and second quartile and a slight change in the fourth quartile for two experiments.

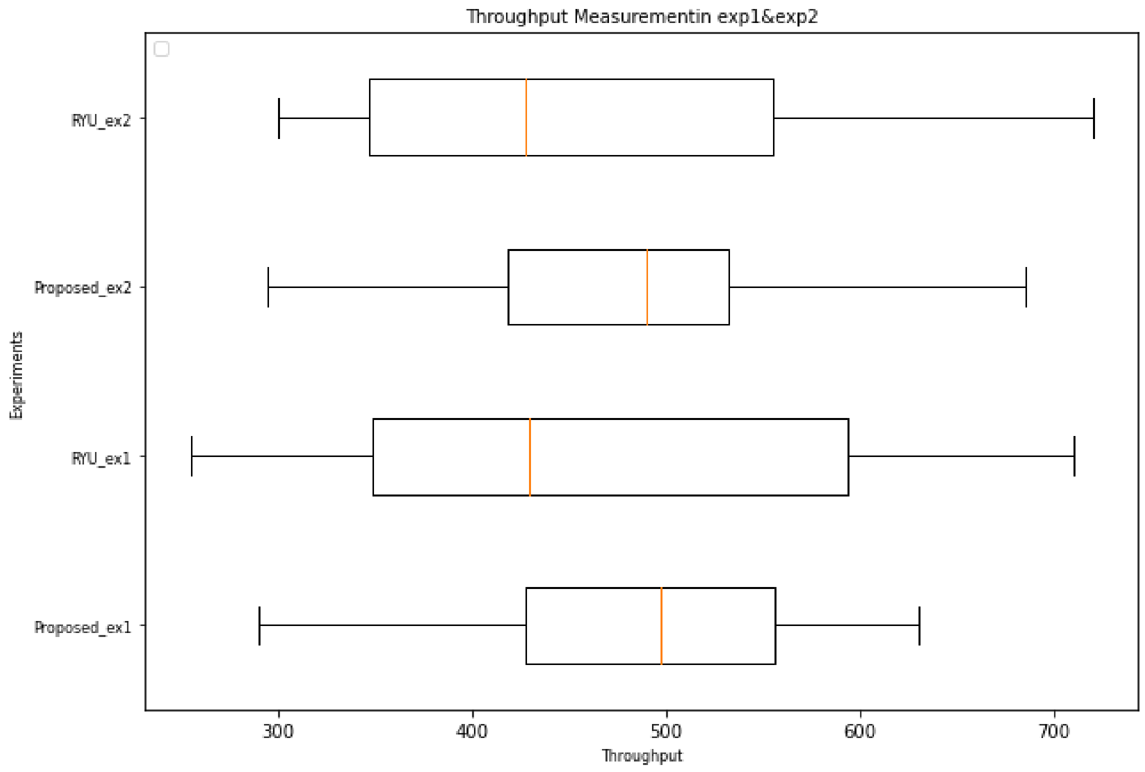


Figure 5.17 Throughput Measurement in Experiment1 and Experiment2

Figure 5.18 presents the measuring of bandwidth usage for the proposed mechanism and the Ryu controller in the two experiments. The plot clarifies clear variation through the experiments that test the usage of bandwidth. For the two methods, there is significant changes in the median, first, second, third, and fourth quartiles in the two experiments. As a result, a clear increase in bandwidth usage through two experiments for both methods.

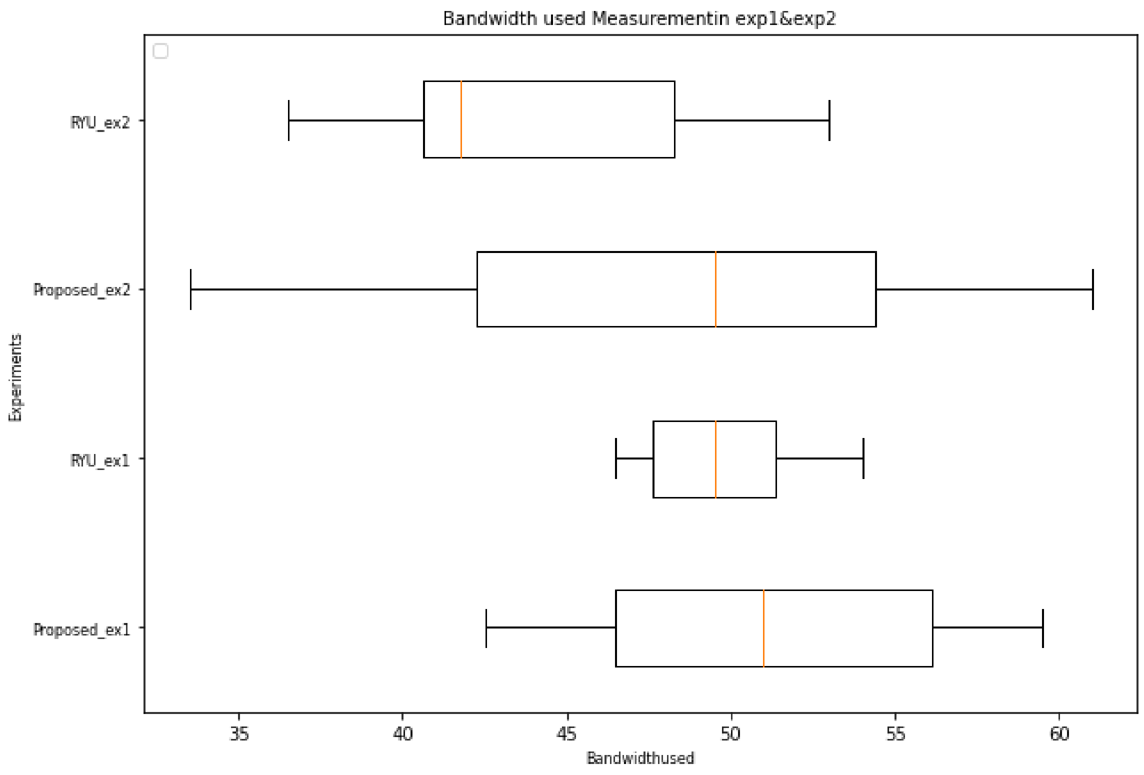


Figure 5.18 Bandwidth Usage Measurement in Experiment1 and Experiment2

Figure 5.19 depicts the third feature, which is the data transfer rate for the two mechanisms in the two experiments. The plot explains significant variation through the experiments for the Ryu controller for the data transfer rate. While our method has a similar median and variance in the ratio of data transfer in the first, second, third, and fourth quartiles in the two experiments.

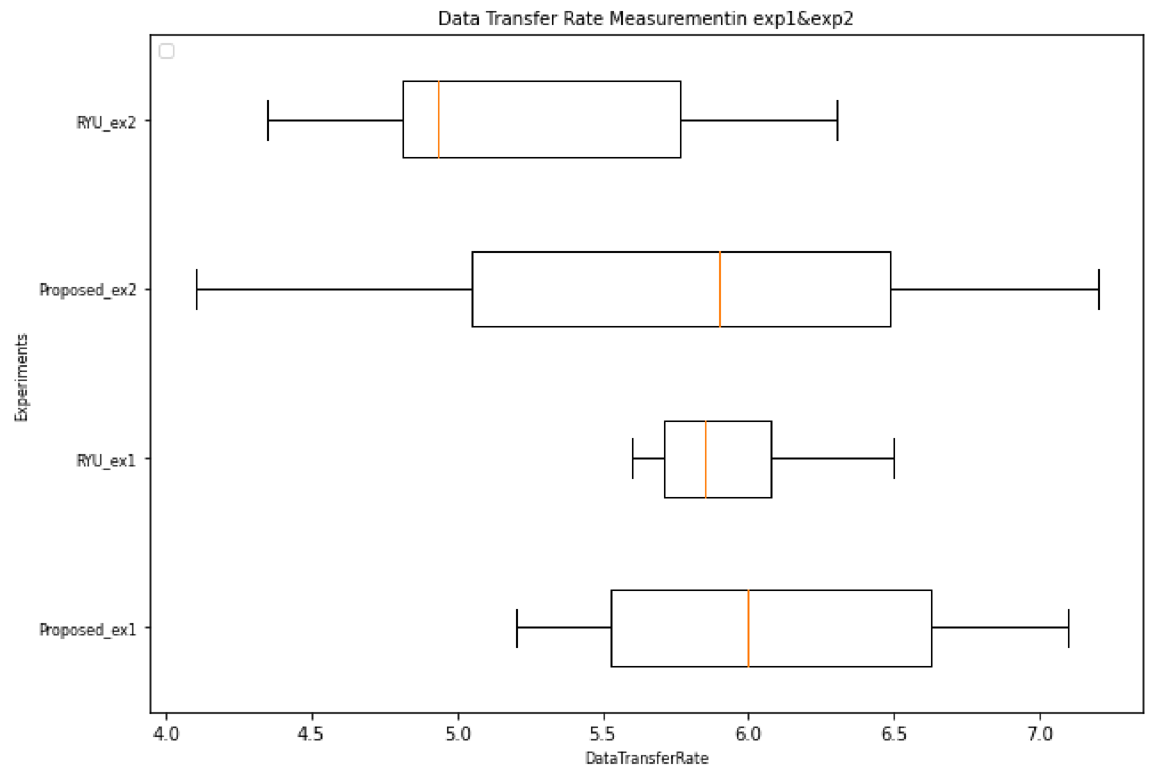


Figure 5.19 Data Transfer Rate Measurement in Experiment1 and Experiment2

5.4 Conclusion

The innovative unified architecture has presented an SDN-based flow routing framework that leverages the concept of flow-level characterization and flow-type identification based on clustering and thresholding techniques that provide a high level of accuracy and flexibility. The flow-level characterization has been executed based on selected metrics of network performance using clustering ideology; consequently, identifying the type of that flow will depend on its own metrics based on pre-defined thresholds. As a result, the best path will be selected for each flow after determining its type and characteristics such as delay and bandwidth. In addition, the Dijkstra algorithm concept-based flow routing technique will be chosen according to the type of flow and cluster-ID.

The most interesting finding was that the presented solution demonstrates an efficient calculation of route cost and consistency with real-time constraints in the SDN environment. This can be seen in the experimental results, which show that 70% of the flows can be routed precisely. Therefore, the dynamic provisions of network resources among different flow types are achieved. For future works, the suggested application sets the ground for designing an automated SDN-based application for routing different types of flow and network topologies to improve bandwidth utilization and reduce congestion across networks.

Chapter Six: Conclusions and Future work

This project presents a novel mechanism to optimize the routing of particular flows by taking advantage of network performance analysis. This Chapter discusses the main contribution to Knowledge and highlights the novelty, research challenges, future work, and the conclusions.

6.1 Contributions to Knowledge

Overall, all the aims that are mentioned in Chapter 1 through this research have been achieved. The core contribution of this project concentrates on performing three experimental studies to investigate the probability of leveraging network performance analysis to characterize and identify elephant and mice flows, consequently, routing them to accomplish the equal utilization of network resources through the design of an application that integrates unsupervised ML and SDN environment. The project establishes the following main contributions.

- For optimizing a network routing mechanism by leveraging its performance analysis, a comprehensive investigation has been established on the topics of network performance metrics, network Traffic analysis methods such as traffic characterization and classification techniques, ML techniques and their applications, Traffic engineering development, and SDN environment architecture and characteristics.
- A baseline of experiment has been conducted to investigate the impact of excluding the driven or control features on the clustering accuracy by using different feature sets. The main aim of this experiment is to gain efficient and accurate characterization of network flows based on network performance features.
- A series of experiments has modeled and performed to investigate the influence of the number of PCA components on the clustering process. The objective of these experiments of gaining a higher accuracy of the clustering

as well as taking into consideration the trade-off between completeness and simplicity.

- Designing an innovative flow characterization model based on utilizing unsupervised ML. This model offers a comprehensive leveraging of network performance metrics to characterize the flows.
- Proposing and evaluating an innovative flow architecture based on pre-defined features and fixed thresholds. This model aims to identify elephant and mice flows precisely using the thresholding method to solve the problem of taking over network resources by elephant flow type in the future.
- A unified architectural solution that integrates unsupervised machine learning techniques and an SDN environment has been proposed. This solution introduces a novel routing mechanism that offers a more balanced and effective network by selecting the best path to route each of the elephant and mice flows based on their requirements

Several papers associated with the research have introduced and published in refereed conferences and journals (Appendix-2). As a result, the project introduced positive contributions to the network management and specifically to the network routing system.

6.2 Research challenges

Although the above research objectives have been achieved, this research work includes some limitations, which are explained hereafter. The main limitations of this research work are described as follows.

1. The flow characterization carried out in the first experiment (**Chapter 3**) primarily relied on flow representation through network performance metrics. An unsupervised ML technique has been used to characterize each flow. An essential aspect of machine learning for training, testing, and validation is the size of the dataset. The aforementioned experiment was

accomplished using a dataset with a small size. This led that the inference of significant relationships among data members being difficult. To solve this limitation, use a dataset with a large size in the second experiment (**Chapter 4**). On the other hand, Outliers are one of the clear problems in data analysis; therefore, detection and manipulation of the outliers are of great effect on Machine Learning because the quality of data is as important as the quality of a prediction or classification model. In the two experiments (**Chapter 3 & Chapter 4**), avoiding outliers' existence completely was not possible, which causes the subsistence of a cluster or more with a size beyond the limits of clusters convergence between 500-4000 connections for each cluster. Using the PCA technique has significantly alleviated this limitation.

2. In PCA, it is significant to retain a suitable number of components based on keeping a balance between simplicity and completeness. However, the choice of clustering accuracy as an aim for the first experiment (**Chapter 3**) led to being restricted to choosing 5 PCA components despite they covered 40% of the data. To avoid this restriction in the second experiment (**Chapter 4**), a different number of PCA components have been experimented with to decide the suitable number of components to achieve an effective clustering process taking into consideration the balance between simplicity and completeness for retaining the appropriate number of principal components as an essential objective of PCA technique.
3. In dynamic networks, one of the essential issues for routing platforms is finding the routing scheme that has the capability to overcome a fault tolerance problem. This problem can be solved by generating more than one path between the source and the destination [223].In addition, the shortest-path algorithm has a superior performance in a dynamic network environment [224]. Our proposed routing mechanism introduces the developed Dijkstra algorithm. The developed algorithm has the ability to

choose between multipath routing and shortest path routing algorithms based on the requirements of the flow type.

4. With increasing the adoption of network traffic encryption in the last years for users' privacy protection. This led to appear of many challenges related to traffic analysis techniques and traffic inspection tools. In this project, a clustering mechanism has been proposed. This mechanism essentially depends on extracting the significant network-related features such as RTT, bytes transmitted, percentage of packet loss, and other network features. Therefore, the proposed framework has the ability to cluster the encrypted traffic [225]. On the other hand, ML classification techniques have the ability to better deal with encrypted traffic [226][227][228].
5. One of the features that have been used in the proposed identification scheme is flow duration. It means all packets should be sent within a certain flow, and the transmission should be completed to determine the duration of the flow. In this case, the transmission of the flow will be delayed for a long period, negatively affecting the performance of the network. For example: If the required time to consider the flow to type elephant is 5 seconds, the transmission will be delayed 5 seconds for each flow, which doesn’t work in real-time. To overcome this problem in real-time, we will assume that all the current packets are from type mice and store the beginning time for each flow, afterword we test the existing duration for every flow, and if it exceeds the specified threshold, it will be elephant type flow.
6. One of the factors that had a clear impact on the performance of the proposed Routing framework is serving higher priority flows such as real-time traffic.

6.3 Future Work Suggestions

1. The idea behind this research supports the area of network management by providing an efficient technique for routing particular flows based on their

pre-defined features and thresholds using the characteristics of the SDN environment. However, to improve or develop the framework of this project further, there are a group of network domains in which future work can be performed. The proposed idea of an unsupervised ML model is compatible with using different feature sets, different types of traffic, and different protocols. For example, the model can be utilized in the network security area to characterize network attacks.

2. In addition, the proposed identification mechanism in this project was based on pre-defined features and static thresholds for distinguishing between elephant and mice flows. Two of these features can be utilized in real-time, so the opportunity to improve this mechanism is required through configuring switches by the controller with the estimated value of the threshold to identify elephant and mice flows at a real-time in a dynamic environment [229] especially since the problem of routing elephant and mice flows is still present.

6.4 Conclusions

Recently, the design goal of the network is network scalability, which means the capability to handle not only immediate demand but also the possibility of traffic growth in the future with little upgrades and costs. Today, traffic engineering is the optimal solution to optimize networks and provide the best services where traffic engineering allows sending traffic over less congested links, regardless of the rule of the shortest path. This can help to mitigate the congestion of the network and to exploit the infrastructure of the network to use it in a better way.

The SDN-based techniques are the best solution to overcome all the challenges of traffic engineering. It provided the possibility of transforming the networks to be flexible and scalable for handling the changes in users' demands. The centralization concept in SDN addresses all the above challenges by calculating the path and specifying the bandwidth for the entire network. Furthermore, the possibility of traffic engineering being a third-party application allows for the creation of a more featured SDN application with a better algorithm.

The novelty of this project is utilizing network performance analysis to characterize network flows through building a clustering model, identifying elephant and mice flows using pre-defined parameters and thresholds to route them based on their requirements by a developed routing algorithm. SDN environment was used to run the proposed framework by creating a consistent unified architecture. The proposed scheme proved to be effective to improve SDN-traffic engineering by routing traffic over less or uncongested links by leveraging the characteristics of the flow. In the end, mitigating the congestion of the network, exploiting the infrastructure of the network to get the most out of it, and achieving the fair use of network resources have been achieved.

References

[1] Y. Wang and Z. Wang, “Explicit routing algorithms for Internet traffic engineering,” in *Proceedings - 8th International Conference on Computer Communications and Networks, ICCCN 1999*, 1999, pp. 582–588, doi: 10.1109/ICCCN.1999.805577.

[2] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, “An overview of routing optimization for internet traffic engineering,” *IEEE Communications Surveys and Tutorials*, vol. 10, no. 1. pp. 36–56, 2008, doi: 10.1109/COMST.2008.4483669.

[3] A. Mendiola, J. Astorga, E. Jacob, M. Higuero, S. Member, and M. Higuero, *A Survey on the Contributions of Software-Defined Networking to Traffic Engineering*, vol. 19, no. 2. 2017.

[4] K. Sawada, D. Kotani, and Y. Okabe, “Network Routing Optimization Based on Machine Learning Using Graph Networks Robust against Topology Change,” in *International Conference on Information Networking*, 2020, vol. 2020-Janua, pp. 608–615, doi: 10.1109/ICOIN48656.2020.9016573.

[5] A. Gupta, “Network Management: Current Trends and Future Perspectives,” *J. Netw. Syst. Manag.*, vol. 14, no. 4, pp. 483–491, Dec. 2006, doi: 10.1007/s10922-006-9044-7.

[6] N. Samaan and A. Karmouch, “Towards autonomic network management: An analysis of current and future research

directions," *IEEE Commun. Surv. Tutorials*, vol. 11, no. 3, pp. 4–21, 2009, doi: 10.1109/SURV.2009.090302.

[7] W. Cui, Y. Yu, and C. Qian, "DiFS: Distributed Flow Scheduling for adaptive switching in FatTree data center networks," *Comput. Networks*, vol. 105, pp. 166–179, 2016, doi: 10.1016/j.comnet.2016.06.003.

[8] S. Verma, Y. Kawamoto, H. Nishiyama, and N. Kato, "A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues," *IEEE Commun. Surv. TUTORIALS*, vol. 19, no. 3, 2017, doi: 10.1109/COMST.2017.2694469.

[9] M. Conti, S. Member, Q. Li, A. Maragno, and R. Spolaor, "The dark side (-channel) of mobile devices: A survey on network traffic analysis," *ieeexplore.ieee.org*, Accessed: Jun. 17, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8371242/.

[10] A. T. b Mahmoud Abbasi a, Amin Shahraki b c, "Deep Learning for Network Traffic Monitoring and Analysis (NTMA): A Survey," *Comput. Commun.*, vol. 170, pp. 19–41, 2021.

[11] A. Shusterman, C. Finkelstein, O. Gruner, Y. S.-C. Networks, and undefined 2021, "Cache-based characterization: A low-infrastructure, distributed alternative to network-based traffic and application characterization," *Elsevier*, Accessed: Feb. 21, 2023. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S1389128621004710.

[12] A. Dainotti, A. Pescapé, and G. Ventre, “A packet-level characterization of network traffic,” *2006 11th Int. Work. Comput. Model. Anal. Des. Commun. Links Networks*, vol. 2006, pp. 38–45, 2006, doi: 10.1109/CAMAD.2006.1649716.

[13] S. Dong *et al.*, “Flow cluster algorithm based on improved K-means method,” *IETE J. Res.*, vol. 59, no. 4, p. 326, 2013, doi: 10.4103/0377-2063.118021.

[14] R. Trestian, G. M. Muntean, and K. Katrinis, “MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow,” in *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management, IM 2013*, 2013, pp. 904–907.

[15] L. C. F. Tang, H. Zhang, L.T. Yang, F. Tang, H. Zhang, L. T. Yang, L. Chen, and L. C. F. Tang, H. Zhang, L.T. Yang, “Elephant Flow Detection and Load-Balanced Routing with Efficient Sampling and Classification,” *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1022–1036, 2021, doi: 10.1109/TCC.2019.2901669.

[16] F. Amezquita-Suarez, F. Estrada-Solano, N. L. S. Da Fonseca, and O. M. C. Rendon, “An Efficient Mice Flow Routing Algorithm for Data Centers Based on Software-Defined Networking,” *IEEE Int. Conf. Commun.*, vol. 2019-May, no. May, 2019, doi: 10.1109/ICC.2019.8761552.

[17] H. Thiri Zaw, A. Htein Maw, H. T. Zaw, A. H. Maw, H. Thiri Zaw, and A. Htein Maw, “Elephant Flow Detection and Delay-Aware Flow Rerouting in Software-Defined Network,” *2017 9th Int. Conf. Inf. Technol. Electr. Eng. ICITEE 2017*, vol. 2018-Janua, pp. 1–6, Jul. 2017, doi: 10.1109/ICITEED.2017.8250487.

[18] K. Boussaoud, M. Ayache, and A. En-Nouaary, “Performance Evaluation of Supervised ML Algorithms for Elephant Flow Detection in SDN; Performance Evaluation of Supervised ML Algorithms for Elephant Flow Detection in SDN,” 2022, doi: 10.1109/ICOA55659.2022.9934652.

[19] A. H. Maw, “Traffic Engineering in Software-Defined Networking (SDN),” vol. 3, no. 5, pp. 1320–1323, 2019.

[20] Y. Lu *et al.*, “SDTCP: Towards Datacenter TCP Congestion Control with SDN for IoT Applications,” doi: 10.3390/s17010109.

[21] R. Mohammadi, S. Akleylek, A. Ghaffari, and A. Shirmarz, “Taxonomy of traffic engineering mechanisms in software-defined networks: a survey,” *Telecommunication Systems*, vol. 81, no. 3. Springer, pp. 475–502, Nov. 01, 2022, doi: 10.1007/s11235-022-00947-6.

[22] K. T. Dinh, S. Kukliński, T. Osiński, and J. Wytrębowicz, “Heuristic traffic engineering for SDN,” *J. Inf. Telecommun.*, vol. 4, no. 3, pp. 251–266, 2020, doi: 10.1080/24751839.2020.1755528.

[23] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: Scaling flow management for high-performance networks,” in *Proceedings of the ACM SIGCOMM 2011 Conference, SIGCOMM’11*, 2011, pp. 254–265, doi: 10.1145/2018436.2018466.

[24] P. Dymora, M. Mazurek, and D. Strzałka, “Computer network traffic analysis with the use of statistical self-similarity factor,” *Ann. UMCS Inform. AI XIII*, vol. 2, pp. 69–81, 2013, doi: 10.2478/v10065-012-0040-0.

[25] F. Pacheco *et al.*, “Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey,” *IEEE Commun. Surv. Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2019, doi: 10.1109/COMST.2018.2883147.

[26] D. Rui, Xu and W., “survey of clustering algorithms,” *IEEE Trans. Neural Netw.*, vol. 16, pp. 269--298, 2009.

[27] P. Berkhin, “A Survey of Clustering Data Mining Techniques,” *Kogan, Jacob; Nicholas, Charles; Teboulle, Marc Group. Multidimens. Data,Springer Press*, pp. 25–72, 2011.

[28] T. Zhang, R. Ramakrishnan, and M. Livny, “BIRCH: An Efficient Data Clustering Method for Very Large Databases,” *SIGMOD Rec. (ACM Spec. Interes. Gr. Manag. Data)*, vol. 25, no. 2, pp. 103–114, 1996, doi: 10.1145/235968.233324.

[29] M. Kiran and A. Chhabra, “Understanding flows in high-speed scientific networks: A Netflow data study,” *Futur. Gener. Comput. Syst.*, vol. 94, pp. 72–79, 2019, doi: 10.1016/j.future.2018.11.006.

[30] B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choudhury, “Design considerations for supporting TCP with per-flow queueing,” *Proc. - IEEE INFOCOM*, vol. 1, pp. 299–306, 1998, doi: 10.1109/infcom.1998.659666.

[31] N. Gude *et al.*, “NOX: Towards an Operating System for Networks,” *ACM SIGCOMM Comput. Commun. Rev. e Submitt. to CCR*, vol. 38, no. 3, pp. 105–110, 2008, Accessed: Sep. 03, 2019. [Online]. Available: http://www.noxrepo.org.

[32] A. J. and R. Dubes, “Algorithms for Clustering Data,” *New Jersey*, 2011.

[33] P. J. F. A. K. Jain, M. N. Murtyand, “Data Clustering: A Review,” *ACM Comput. Surv.*, vol. 31, pp. 264–324, 2012.

[34] R. Ben Basat, G. Einziger, R. Friedman, and Y. Kassner, *Optimal elephant flow detection*. Institute of Electrical and Electronics Engineers Inc., 2017.

[35] J. Alkenani and K. Nassar, “Network Monitoring Measurements for Quality of Service: A Review,” *Iraqi J. Electr. Electron. Eng.*, vol. 18, no. 2, pp. 33–42, 2022, doi: 10.37917/ijeee.18.2.5.

[36] G. Martinovic, B. Petrisevac, and D. Zagar, “Monitoring and

measurement of computer network performance," *Teh. Vjesn.*, vol. 17, no. 3, pp. 317–326, 2010, [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-79951988229&partnerID=40&md5=c1913a7e7a20d4942b5ebed3666e5278.

[37] L. H. Zhiwei Cen, "Measurement and Analysis of Ip," no. March 2003, 2016.

[38] A. D. Orcesi, D. M. Frangopol, and S. Kim, "Network Performance Measurement and Monitoring," *Eng. Struct.*, vol. 1, pp. 1–14, 2009, doi: 10.1016/j.engstruct.2009.11.009.

[39] ITU-T, "General Aspects of Quality of Service and Network Performance, Including ISDNs, ITU. I.350," 1993.

[40] A. Hanemann, A. Liakopoulos, M. Molina, and D. M. Swany, "A study on network performance metrics and their composition," *Campus-Wide Inf. Syst.*, vol. 23, no. 4, pp. 268–282, 2006, doi: 10.1108/10650740610704135.

[41] A. A.Obiniyi, M. B. Soroyewun, and M. M. Abur, "New Innovations in Performance Analysis of Computer Networks: A Review," *Int. J. Appl. Inf. Syst.*, vol. 6, no. 8, pp. 1–10, 2014, doi: 10.5120/ijais14-451085.

[42] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and

comparison," *Perform. Eval.*, vol. 67, no. 6, pp. 451–467, 2010, doi: 10.1016/j.peva.2010.01.001.

[43] S. H. Yoon, J. S. Park, and M. S. Kim, "Behavior signature for fine-grained traffic identification," *Appl. Math. Inf. Sci.*, vol. 9, no. 2, pp. 523–534, 2015, doi: 10.12785/amis/092L27.

[44] M. Shen *et al.*, "Machine Learning-Powered Encrypted Network Traffic Analysis: A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 25, no. 1, pp. 791–824, 2022, doi: 10.1109/COMST.2022.3208196.

[45] N. Alqudah and Q. Yaseen, "Machine Learning for Traffic Analysis: A Review," *Procedia Comput. Sci.*, vol. 170, pp. 911–916, 2020, doi: 10.1016/j.procs.2020.03.111.

[46] O. Aouedi, K. Piamrat, S. Hamma, and J. K. M. Perera, "Network traffic analysis using machine learning: an unsupervised approach to understand and slice your network," *Ann. des Telecommun. Telecommun.*, vol. 77, no. 5–6, pp. 297–309, 2022, doi: 10.1007/s12243-021-00889-1.

[47] H. Luo, C. Liu, and Y. Liang, "A SDN-based Testbed for Underwater Sensor Networks," *ACM Int. Conf. Proceeding Ser.*, no. April, 2019, doi: 10.1145/3321408.3321410.

[48] A. Bulashenko, S. Piltyay, O. Bulashenko, and A. Polishchuk, "New Traffic Model of M2M Technology in 5G Wireless Sensor Networks;

New Traffic Model of M2M Technology in 5G Wireless Sensor Networks," *2020 IEEE 2nd Int. Conf. Adv. Trends Inf. Theory*, 2020, doi: 10.1109/ATIT50783.2020.9349305.

[49] G. A. Michael Kuchnik, Ana Klimovic, Jiri Simsa, Virginia Smith, "Plumber: Diagnosing and Removing Performance Bottlenecks in Machine Learning Data Pipelines," *Proc. Mach. Learn. Syst.*, vol. 4, 2022.

[50] M. H. Fortier PJ, *Computer Systems Performance Evaluation and Prediction*. Butterworth-Heinemann:USA, 2003.

[51] G. Ruth, "Traffic Flow Measurement: Architecture," pp. 1–48, 1999.

[52] A. Us, "Active vs . Passive network monitoring : an i f Navigate :," vol. 474767, pp. 1–5, 2018, [Online]. Available: https://www.irisns.com/active-vs-passive-network-monitoring-an-infographic/.

[53] T. C. Luz, G. A. Nunez, C. B. Margi, and F. L. Verdi, "In-network performance measurements for Software Defined Wireless Sensor Networks; In-network performance measurements for Software Defined Wireless Sensor Networks," *2019 IEEE 16th Int. Conf. Networking, Sens. Control*, 2019.

[54] M. Mainuddin, Z. Duan, Y. Dong, S. Salman, and T. Taami, "IoT Device Identification Based on Network Traffic Characteristics," pp. 6067–6072, 2023, doi: 10.1109/globecom48099.2022.10001639.

[55] A. Montieri, G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapè, "Packet-level prediction of mobile-app traffic using multitask Deep Learning," *Comput. Networks*, vol. 200, p. 108529, Dec. 2021, doi: 10.1016/J.COMNET.2021.108529.

[56] G. Aceto *et al.*, "Characterization and Prediction of Mobile-App Traffic Using Markov Modeling," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 1, 2021, doi: 10.1109/TNSM.2021.3051381.

[57] M. Mainuddin, Z. Duan, and Y. Dong, "Network Traffic Characteristics of IoT Devices in Smart Homes; Network Traffic Characteristics of IoT Devices in Smart Homes," 2021, doi: 10.1109/ICCCN52240.2021.9522168.

[58] R. Roy Chowdhury, S. Aneja, N. Aneja, and P. E. Abas, "Packet-level and IEEE 802.11 MAC frame-level network traffic traces data of the D-Link IoT devices," *Data Br.*, vol. 37, Aug. 2021, doi: 10.1016/J.DIB.2021.107208.

[59] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics (Extended Version)," *Network, IEEE*, vol. 11, no. 6, pp. 1–28, 1997, doi: 10.1109/65.642356.

[60] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *Proc. 2005 ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst. - SIGMETRICS '05*, vol. 33, no. 1, p. 50, 2005, doi: 10.1145/1064212.1064220.

[61] R. Bhattacharjee and G. Santhosh Kumar, “User characterization through network flow analysis; User characterization through network flow analysis,” 2016, doi: 10.1109/ICDSE.2016.7823965.

[62] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z. L. Zhang, “A modular machine learning system for flow-level traffic classification in large networks,” *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, Mar. 2012, doi: 10.1145/2133360.2133364.

[63] M. J. Vargas-Muñoz *et al.*, “Classification of network anomalies in flow level network traffic using Bayesian networks; Classification of network anomalies in flow level network traffic using Bayesian networks,” 2018, doi: 10.1109/CONIELECOMP.2018.8327205.

[64] Z. A. G. H. Shaikh, C. Science, and C. Science, “An Overview of Network Traffic Classification Methods,” *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 3, no. 2, pp. 482–488, 2015.

[65] A. W. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” *Passiv. Act. Netw. Meas.*, vol. 3431, pp. 41–54, 2005, doi: 10.1007/978-3-540-31966-5_4.

[66] A. Madhukar and C. Williamson, “A Longitudinal Study of P2P Traffic Classification,” *14th IEEE Int. Symp. Model. Anal. Simul.*, pp. 179–188, 2006, doi: 10.1109/MASCOTS.2006.6.

[67] S. Sen, O. Spatscheck, and D. Wang, “Accurate, scalable in-network identification of p2p traffic using application signatures,” *Proc. 13th*

Conf. World Wide Web - WWW '04, p. 512, 2004, doi: 10.1145/988672.988742.

[68] O. N. Networking, "of Wide-Area TCP Connections," *Methodology*, vol. 2, no. 4, pp. 316–336, 1994.

[69] C. Dewes and M. Tu, "An A nalysis of I nternet C hat S ystems," pp. 51–64.

[70] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Commun. Surv. Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, 2008, doi: 10.1109/SURV.2008.080406.

[71] R. Kumar and T. Kaur, "Machine Learning based Traffic Classification using Low Level Features and Statistical Analysis," *Int. J. Comput. Appl.*, vol. 108, no. 12, pp. 6–13, 2014, doi: 10.5120/18961-0290.

[72] I. H. Witten, E. Frank, and M. a Hall, *Data Mining: Practical Machine Learning Tools and Techniques (Google eBook)*. 2011.

[73] H. Singh, "Performance Analysis of Unsupervised Machine Learning Techniques for Network Traffic Classification," *2015 Fifth Int. Conf. Adv. Comput. Commun. Technol.*, pp. 401–404, 2015, doi: 10.1109/ACCT.2015.54.

[74] F. Behrad and M. Saniee Abadeh, "An overview of deep learning methods for multimodal medical data mining," *Expert Syst. Appl.*,

vol. 200, no. April, p. 117006, 2022, doi: 10.1016/j.eswa.2022.117006.

[75] S. Moraboena, G. Ketepalli, and P. Ragam, “A deep learning approach to network intrusion detection using deep autoencoder,” *Rev. d’Intelligence Artif.*, vol. 34, no. 4, pp. 457–463, 2020, doi: 10.18280/ria.340410.

[76] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, and B. Yin, “Deep Learning on Traffic Prediction: Methods, Analysis, and Future Directions,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4927–4943, 2022, doi: 10.1109/TITS.2021.3054840.

[77] I. H. Sarker, “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions,” *SN Computer Science*, vol. 2, no. 6. Springer Singapore, 2021, doi: 10.1007/s42979-021-00815-1.

[78] L. Deng, “A tutorial survey of architectures, algorithms, and applications for deep learning,” *APSIPA Trans. Signal Inf. Process.*, vol. 3, 2014, doi: 10.1017/ATSIP.2013.99.

[79] Z. M. Fadlullah *et al.*, “State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow’s Intelligent Network Traffic Control Systems,” *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017, doi: 10.1109/COMST.2017.2707140.

[80] D. P and G. C, “A systematic review on machine learning and deep

learning techniques in cancer survival prediction," *Prog. Biophys. Mol. Biol.*, vol. 174, pp. 62–71, 2022, doi: 10.1016/j.pbiomolbio.2022.07.004.

[81] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," doi: 10.1109/ACCESS.2019.2895334.

[82] G. Aceto, D. Ciuonzo, S. Member, A. Montieri, G. Student Member, and A. Pescapé, "Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 2, p. 445, 2019, doi: 10.1109/TNSM.2019.2899085.

[83] Z. Wang, "The Applications of Deep Learning on Traffic Identification," *Black Hat USA*, 2015, Accessed: May 15, 2023. [Online]. Available: https://paper.bobylive.com/Meeting_Papers/BlackHat/USA-2015/us-15-Wang-The-Applications-Of-Deep-Learning-On-Traffic-Identification-wp.pdf.

[84] T. P. Oliveira, J. S. Barbar, and A. S. Soares, "Multilayer perceptron and stacked autoencoder for Internet traffic prediction," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8707 LNCS, pp. 61–71, 2014, doi: 10.1007/978-3-662-44917-2_6/COVER.

[85] S. Garg *et al.*, “A Hybrid Deep Learning-Based Model for Anomaly Detection in Cloud Datacenter Networks,” *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 3, 2019, doi: 10.1109/TNSM.2019.2927886.

[86] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, “An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection,” doi: 10.1109/ACCESS.2020.2973023.

[87] I. Ullah and Q. H. Mahmoud, “Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks,” doi: 10.1109/ACCESS.2021.3094024.

[88] A. Coates and A. Y. Ng, “Selecting receptive fields in deep networks,” in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 2011, pp. 2528–2536.

[89] S. Gao, H. Pang, P. Gallinari, J. Guo, and N. Kato, “A Novel Embedding Method for Information Diffusion Prediction in Social Network Big Data,” *IEEE Trans. Ind. INFORMATICS*, vol. 13, no. 4, 2017, doi: 10.1109/TII.2017.2684160.

[90] K. L. Dias, M. A. Pongelupe, W. M. Caminhas, and L. de Errico, “An innovative approach for real-time network traffic classification,” *Comput. Networks*, vol. 158, pp. 143–157, Jul. 2019, doi: 10.1016/J.COMNET.2019.04.004.

[91] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing,

“Learning from class-imbalanced data: Review of methods and applications,” *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017, doi: 10.1016/J.ESWA.2016.12.035.

[92] M. F. Umer, M. Sher, and Y. Bi, “Flow-based intrusion detection: Techniques and challenges,” *Comput. Secur.*, vol. 70, pp. 238–254, Sep. 2017, doi: 10.1016/J.COSE.2017.05.009.

[93] J. Zhang, Y. Y. Xiang, Y. Wang, W. Zhou, Y. Y. Xiang, and Y. Guan, “Network traffic classification using correlation information,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 104–117, 2013, doi: 10.1109/TPDS.2012.98.

[94] H. Chen and L. Trajković, “Trunked radio systems: Traffic prediction based on user clusters,” *1st Int. Symp. Wirel. Commun. Syst. 2004, Proc. ISWCS ’04*, pp. 76–80, 2004, doi: 10.1109/iswcs.2004.1407212.

[95] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, “Unsupervised traffic classification using flow statistical properties and IP packet payload,” *J. Comput. Syst. Sci.*, vol. 79, no. 5, pp. 573–585, 2013, doi: 10.1016/j.jcss.2012.11.004.

[96] G. Z. Lin, Y. Xin, X. X. Niu, and H. B. Jiang, “Network traffic classification based on semi-supervised clustering,” *J. China Univ. Posts Telecommun.*, vol. 17, no. SUPPL. 2, pp. 84–88, 2010, doi: 10.1016/S1005-8885(09)60577-X.

[97] M. Zulfadhilah, Y. Prayudi, and I. Riadi, “Cyber Profiling Using Log Analysis And K-Means Clustering,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 7, pp. 430–435, 2016, doi: 10.14569/ijacsa.2016.070759.

[98] S. Rajagopal, “CUSTOMER DATA CLUSTERING USING DATA MINING TECHNIQUE,” *Int. J. Database Manag. Syst. (IJDMS*, vol. 3, no. 4, Dec. 2011, doi: 10.5121/ijdms.2011.3401.

[99] G. Wang, X. Zhang, S. Tang, H. Zheng, and B. Y. Zhao, “Unsupervised Clickstream Clustering for User Behavior Analysis,” *Proc. 2016 CHI Conf. Hum. Factors Comput. Syst. - CHI ’16*, pp. 225–236, 2016, doi: 10.1145/2858036.2858107.

[100] N. Huidrom and N. Bagoria, “Clustering Techniques for the Identification of Web User Session,” *Int. J. Sci. Res. Publ.*, vol. 3, no. 1, pp. 1–4, 2013.

[101] R. Ranjan and G. Sahoo, “A New Clustering Approach for Anomaly Intrusion Detection,” *Int. J. Data Min. Knowl. Manag. Process*, vol. 4, no. 2, p. 10, 2014, doi: 10.5121/ijdkp.2014.4203.

[102] D. Hock, M. Kappes, and B. Ghita, “A pre-clustering method to improve anomaly detection,” *ICETE 2016 - Proc. 13th Int. Jt. Conf. E-bus. Telecommun.*, vol. 4, no. July, pp. 391–396, 2016, doi: 10.5220/0005953003910396.

[103] S.Shraddha, “Intrusion Detection using Artificial Neural Network,” *Adv. Neural Networks, Fuzzy Syst. Artif. Intell. Intrusion*, no. 1, pp.

209–217, 2014.

[104] V. Kumar, H. Chauhan, and D. Panwar, “K-Means Clustering Approach to Analyze NSL-KDD Intrusion Detection Dataset,” *Int. J. Soft Comput. Eng.*, vol. 3, no. 4, pp. 1–4, 2013.

[105] G. Han, J. Jiang, N. Bao, L. Wan, and M. Guizani, “Routing protocols for underwater wireless sensor networks,” *IEEE Commun. Mag.*, vol. 53, no. 11, pp. 72–78, 2015, doi: 10.1109/MCOM.2015.7321974.

[106] L. Gouveia, P. Patrício, and A. De Sousa, “Hop-Constrained Node Survivable Network Design : An Application to MPLS over WDM,” pp. 3–4, 2008, doi: 10.1007/s11067-007-9038-3.

[107] J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin, “SOFTWARE-DEFINED INTERNET OF THINGS FOR SMART URBAN SENSING,” *IEEE Commun. Mag.*, vol. 53(9), no. September, pp. 55–63, 2015.

[108] O.N.F., “Software-defined networking: The new norm for networks,” *ONF White Pap.*, vol. 2, pp. 2–6, 2012, doi: citeulike-article-id:12475417.

[109] J. Wickboldt, W. De Jesus, P. Isolani, C. Both, J. Rochol, and L. Granville, “Software-defined networking: Management requirements and challenges,” *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 278–285, Jan. 2015, doi: 10.1109/MCOM.2015.7010546.

[110] A. Yassine, … H. R.-I. I. &, and undefined 2015, “Software defined

network traffic measurement: Current trends and challenges," *ieeexplore.ieee.org*, 2015, doi: 10.1109/MIM.2015.7066685.

[111] M. R. Parsaei, M. J. Sobouti, S. Raouf, and R. Javidan, "Network Traffic Classification using Machine Learning Techniques over Software Defined Networks," 2017.

[112] A. Al-Jawad, R. Trestian, P. Shah, and O. Gemikonakli, "BaProbSDN: A probabilistic-based QoS routing mechanism for Software Defined Networks," 2015, doi: 10.1109/NETSOFT.2015.7116128.

[113] M. Alkasassbeh, G. Al-Naymat, M. Alauthman, and E. Ednat, "Optimizing Traffic Engineering in Software Defined Networking," no. November, 2018, [Online]. Available: https://www.preprints.org/manuscript/201811.0486/v1.

[114] M. C. De Toro and C. Borrego, "A Software-Defined Networking approach for congestion control in Opportunistic Networking," *Int. Conf. Inf. Netw.*, vol. 2020-Janua, pp. 354–359, Jan. 2020, doi: 10.48550/arxiv.2001.05257.

[115] S. T. V. Pasca, S. S. P. Kodali, and K. Kataoka, "AMPS: Application aware multipath flow routing using machine learning in SDN," *2017 23rd Natl. Conf. Commun. NCC 2017*, 2017, doi: 10.1109/NCC.2017.8077095.

[116] P. Wang, S. C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," in

Proceedings - 2016 IEEE International Conference on Services Computing, SCC 2016, 2016, pp. 760–765, doi: 10.1109/SCC.2016.133.

[117] J. Chen, X. Zheng, and C. Rong, “Survey on software-defined networking,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9106, no. 1, pp. 115–124, 2015, doi: 10.1007/978-3-319-28430-9_9.

[118] M. Cokic and I. Seskar, “Software defined network management for dynamic smart GRID traffic,” *Futur. Gener. Comput. Syst.*, vol. 96, pp. 270–282, Jul. 2019, doi: 10.1016/J.FUTURE.2019.02.022.

[119] A. Stamou, G. Kakkavas, K. Tsitseklis, V. Karyotis, and S. Papavassiliou, “Autonomic Network Management and Cross-Layer Optimization in Software Defined Radio Environments,” doi: 10.3390/fi11020037.

[120] M. Aboubakar, M. Kellil, and P. Roux, “A review of IoT network management: Current status and perspectives,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 7, pp. 4163–4176, Jul. 2022, doi: 10.1016/J.JKSUCI.2021.03.006.

[121] I. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A Roadmap for Traffic Engineering in Software Defined Networks,” *Comput. Networks*, no. June, pp. 1–27, 2014, doi: 10.1016/j.comnet.2014.06.002.

[122] W. Braun and M. Menth, “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices,” *Futur. Internet*, vol. 6, no. 2, pp. 302–336, 2014, doi: 10.3390/fi6020302.

[123] M. Alsaeedi, M. Murtadha Mohamad, and A. A. Al-Roubaiey, “Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey,” doi: 10.1109/ACCESS.2019.2932422.

[124] M. Dash and H. Liu, “Feature selection for classification,” *Intell. Data Anal.*, vol. 1, no. 3, pp. 131–156, Jan. 1997, doi: 10.3233/IDA-1997-1302.

[125] H. Liu and L. Yu, “Toward integrating feature selection algorithms for classification and clustering,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, 2005, doi: 10.1109/TKDE.2005.66.

[126] H. Fröhlich, O. Chapelle, and B. Schölkopf, “Feature Selection for Support Vector Machines by Means of Genetic Algorithms,” *Proc. Int. Conf. Tools with Artif. Intell.*, pp. 142–148, 2003, doi: 10.1109/TAI.2003.1250182.

[127] I. Guyon, A. E.-J. of machine learning research, and undefined 2003, “An introduction to variable and feature selection,” *jmlr.org*, vol. 3, pp. 1157–1182, 2003.

[128] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, “An intelligent algorithm with feature selection and decision rules applied to

anomaly intrusion detection," *Appl. Soft Comput.*, vol. 12, pp. 3285–3290, 2012, doi: 10.1016/j.asoc.2012.05.004.

[129] J. T. de Souza, "Feature Selection with a General Hybrid Algorithm," *Diss. Univ. Ottawa*, no. August, 2004.

[130] A. Hashemi and M. B. Dowlatshahi, "MLCR: A Fast Multi-label Feature Selection Method Based on K-means and L2-norm," 2020, doi: 10.1109/CSICC49403.2020.9050104.

[131] A. F. H. Alharan, H. K. Fatlawi, and N. S. Ali, "A cluster-based feature selection method for image texture classification," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 14, no. 3, p. 1433, Jan. 2019, doi: 10.11591/IJEECS.V14.I3.PP1433-1442.

[132] S. Chormunge and S. Jena, "Correlation based feature selection with clustering for high dimensional data," *J. Electr. Syst. Inf. Technol.*, vol. 5, no. 3, pp. 542–549, Dec. 2018, doi: 10.1016/J.JESIT.2017.06.004.

[133] G. D'Angelo and F. Palmieri, "Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction," *J. Netw. Comput. Appl.*, vol. 173, p. 102890, Jan. 2021, doi: 10.1016/J.JNCA.2020.102890.

[134] M. Usama *et al.*, "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges," *ieeexplore.ieee.org*, 2019, doi: 10.1109/ACCESS.2019.2916648.

[135] I. H. Sarker, “Machine Learning: Algorithms, Real-World Applications and Research Directions,” *SN Comput. Sci.*, vol. 2, no. 3, May 2021, doi: 10.1007/S42979-021-00592-X.

[136] Cisco, “Cisco IOS NetFlow.” https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html (accessed Oct. 01, 2018).

[137] International Telecommunication Union, “Terms and Definitions Related to Quality of Service and Network Performance Including Dependability,” 1994.

[138] F. Pacheco *et al.*, “Towards the deployment of machine learning solutions in network traffic classification: A systematic survey,” *ieeexplore.ieee.org*, vol. 21, no. 2, pp. 1988–2014, 2018, doi: 10.1109/COMST.2018.2883147ï.

[139] M. Lotfollahi *et al.*, “Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning,” *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, Feb. 2017, doi: 10.1007/S00500-019-04030-2.

[140] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, “Anatomy of a large european IXP,” *SIGCOMM’12 - Proc. ACM SIGCOMM 2012 Conf. Appl. Technol. Archit. Protoc. Comput. Commun.*, pp. 163–174, 2012, doi: 10.1145/2342356.2342393.

[141] L. Guo and I. Matta, “The war between mice and elephants,” *Int.*

Conf. Netw. Protoc., pp. 180–188, 2001, doi: 10.1109/ICNP.2001.992898.

[142] A. Chhabra and M. Kiran, “Classifying Elephant and Mice Flows in High-Speed Scientific Networks,” *Prepr. Submitt. to INDIS Sept. 19, 2017*, 2017.

[143] X. Wu and X. Yang, “DARD: Distributed adaptive routing for datacenter networks,” in *Proceedings - International Conference on Distributed Computing Systems*, 2012, pp. 32–41, doi: 10.1109/ICDCS.2012.69.

[144] W. Wang, Y. Sun, K. Zheng, M. A. Kaafar, D. Li, and Z. Li, “Concise Paper: Freeway: Adaptively Isolating the Elephant and Mice Flows on Different Transmission Paths,” 2014, doi: 10.1109/ICNP.2014.59.

[145] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “PolicyCop: An autonomic QoS policy enforcement framework for software defined networks,” 2013. doi: 10.1109/SDN4FNS.2013.6702548.

[146] A. Santos Da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, “ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN,” *Proc. NOMS 2016 - 2016 IEEE/IFIP Netw. Oper. Manag. Symp.*, no. Noms, pp. 27–35, 2016, doi: 10.1109/NOMS.2016.7502793.

[147] R. Gonzalez, C. Soriente, and N. Laoutaris, “User profiling in the

time of HTTPS," *Proc. ACM SIGCOMM Internet Meas. Conf. IMC*, vol. 14-16-Nove, pp. 373–379, 2016, doi: 10.1145/2987443.2987451.

[148] T. Mai, D. Ajwani, and A. Sala, "Profiling user activities with minimal traffic traces," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9114, pp. 116–133, 2015, doi: 10.1007/978-3-319-19890-3_9/COVER/.

[149] T. Bakhshi and B. Ghita, "OpenFlow-enabled user traffic profiling in campus software defined networks," in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2016, pp. 1–8.

[150] R. Thamilselvan, K. T. Selvi, R. R. Rajalaxmi, and E. Gothai, "Multipath Routing of Elephant Flows in Data Centers Based on Software Defined Networking," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 2, pp. 2714–2717, 2019, doi: 10.35940/ijeat.b3258.129219.

[151] S. Tomovic, N. Lekic, I. Radusinovic, and G. Gardasevic, *A new approach to dynamic routing in SDN networks*. IEEE, 2016, pp. 1–6.

[152] L. A. D. Knob, R. P. Esteves, L. Z. Granville, and L. M. R. Tarouco, "Mitigating elephant flows in SDN-based IXP networks," *Proc. - IEEE Symp. Comput. Commun.*, no. Noms, pp. 1352–1359, 2017, doi: 10.1109/ISCC.2017.8024712.

[153] S. Bavugi and V. Sahni, "Detection and Re-routing of elephant flows in a Software Defined Networking to avoid traffic congestion,"

2018, Accessed: Jun. 25, 2022. [Online]. Available: http://norma.ncirl.ie/3303/.

[154] H. T. Zaw and A. H. Maw, “Traffic management with elephant flow detection in software defined networks (SDN),” *Int. J. Electr. Comput. Eng.*, vol. 9, no. 4, pp. 3203–3211, 2019, doi: 10.11591/ijece.v9i4.pp3203-3211.

[155] J. Reed, “What Is Hyper-V Manager and How Does It Work?” https://www.nakivo.com/blog/what-is-hyper-v-manager-and-how-does-it-work/ (accessed Apr. 02, 2019).

[156] M. Hamdan *et al.*, “DPLBAnt: Improved load balancing technique based on detection and rerouting of elephant flows in software-defined networks,” *Comput. Commun.*, vol. 180, pp. 315–327, 2021, doi: 10.1016/j.comcom.2021.10.013.

[157] Q. Fu, E. Sun, E. Sun, K. Meng, M. Li, and Y. Zhang, “Deep Q-Learning for Routing Schemes in SDN-Based Data Center Networks,” *IEEE Access*, vol. 8, pp. 103491–103499, 2020, doi: 10.1109/ACCESS.2020.2995511.

[158] M. Kiran, B. Mohammed, and N. Krishnaswamy, “DeepRoute: Herding Elephant and Mice Flows with Reinforcement Learning,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12081 LNCS, pp. 296–314, 2020, doi: 10.1007/978-3-030-45778-5_20.

[159] C. Zhang, X. Wang, F. Li, and M. Huang, "NNIRSS: neural network-based intelligent routing scheme for SDN," *Neural Comput. Appl.*, vol. 31, no. 10, pp. 6189–6205, Oct. 2019, doi: 10.1007/S00521-018-3427-Z.

[160] D. Kreutz, F. Ramos, … P. V.-P. of the, and U. 2014, "Software-defined networking: A comprehensive survey," *ieeexplore.ieee.org*, 2014, Accessed: Jul. 18, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6994333/.

[161] Z. Bu, B. Zhou, P. Cheng, K. Zhang, and Z. H. Ling, "Encrypted Network Traffic Classification Using Deep and Parallel Network-in-Network Models," *IEEE Access*, vol. 8, pp. 132950–132959, 2020, doi: 10.1109/ACCESS.2020.3010637.

[162] L. Yang, S. Fu, X. Zhang, S. Guo, Y. Wang, and C. Yang, "FlowSpectrum: a concrete characterization scheme of network traffic behavior for anomaly detection," *World Wide Web*, vol. 25, no. 5, pp. 2139–2161, 2022, doi: 10.1007/s11280-022-01057-8.

[163] Draper-Gil, G., Lashkari, A.H., Mamun, M.S.I. , Ghorbani, A.A., "Characterization of Encrypted and VPN Traffic using Time-related Features," *Proc. 2nd Int. Conf. Inf. Syst. Secur. Priv.*, pp. 407–414, 2016.

[164] M. Di Mauro and M. Longo, "Revealing encrypted WebRTC traffic via machine learning tools; Revealing encrypted WebRTC traffic via machine learning tools," 2015.

[165] D. Bebortta, S. , Senapati, “Empirical characterization of network traffic for reliable communication in IoT devices,” *Secur. cyber-physical Syst. Found. Appl.*, pp. 67–90, 2021.

[166] “TCPDUMP&LiBPCAP.” http://www.tcpdump.org/ (accessed Oct. 01, 2018).

[167] Shawn Ostermann, “tcptrace.” http://www.tcptrace.org/ (accessed Oct. 01, 2018).

[168] S. Ayesha, M. K. Hanif, and R. Talib, “Overview and comparative study of dimensionality reduction techniques for high dimensional data,” *Inf. Fusion*, vol. 59, pp. 44–58, 2020, doi: 10.1016/j.inffus.2020.01.005.

[169] D. Q. Zeebaree, H. Haron, A. Mohsin Abdulazeez, and S. R. M. Zeebaree, “Combination of K-means clustering with Genetic Algorithm: A review,” *Int. J. Appl. Eng. Res.*, vol. 12, pp. 14238–14245, 2017, Accessed: May 16, 2023. [Online]. Available: http://www.ripublication.com.

[170] S. Khare and M. Totaro, “Big Data in IoT,” in *2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019*, 2019, pp. 1–7, doi: 10.1109/ICCCNT45670.2019.8944495.

[171] M. A. Mohammed *et al.*, “Neural network and multi-fractal dimension features for breast cancer classification from ultrasound

images R," *Comput. Electr. Eng.*, vol. 70, pp. 871–882, 2018, doi: 10.1016/j.compeleceng.2018.01.033.

[172] M. Li, H. Wang, L. Yang, Y. Liang, Z. Shang, and H. Wan, "Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction," *Expert Syst. Appl.*, vol. 150, p. 113277, 2020, doi: 10.1016/j.eswa.2020.113277.

[173] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar Joseph, "A Review of Dimensionality Reduction Techniques for Efficient Computation," *Procedia Comput. Sci.*, vol. 165, pp. 104–111, 2019, doi: 10.1016/J.PROCS.2020.01.079.

[174] W. Wang, W.-G. Shen, Y.-X. Sun, B. Chen, and R. Zhu, "Dimensionality reduction via adjusting data distribution density," 2018.

[175] J. Stuckman, J. Walden, and R. Scandariato, "The Effect of Dimensionality Reduction on Software Vulnerability Prediction Models," *IEEE Trans. Reliab.*, vol. 66, no. 1, 2017, doi: 10.1109/TR.2016.2630503.

[176] Q. Fournier and D. Aloise, "Empirical comparison between autoencoders and traditional dimensionality reduction methods," in *Proceedings - IEEE 2nd International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2019*, 2019, pp. 211–214, doi: 10.1109/AIKE.2019.00044.

[177] D. Hai Hoang and H. Duong Nguyen, “A PCA-based method for IoT network traffic anomaly detection; A PCA-based method for IoT network traffic anomaly detection,” 2018, doi: 10.23919/ICACT.2018.8323766.

[178] K. Keerthi Vasan and B. Surendiran, “Dimensionality reduction using Principal Component Analysis for network intrusion detection,” *Perspect. Sci.*, vol. 8, pp. 510–512, Sep. 2016, doi: 10.1016/J.PISC.2016.05.010.

[179] J. Almotiri, K. Elleithy, and A. Elleithy, “Comparison of Autoencoder and Principal Component Analysis Followed by Neural Network for E-Learning Using Handwritten Recognition,” 2017, doi: 10.1109/LISAT.2017.8001963.

[180] Z. Wang, D. Han, M. Li, H. Liu, and M. Cui, “The abnormal traffic detection scheme based on PCA and SSH,” *Conn. Sci.*, vol. 2022, no. 1, pp. 1201–1220, 2022, doi: 10.1080/09540091.2022.2051434.

[181] V. L. Chetana, S. S. Kolisetty, and K. Amogh, “A Short Survey of Dimensionality Reduction Techniques,” in *Recent Advances in Computer Based Systems, Processes and Applications*, 2020, pp. 3–14.

[182] J. Josse and F. Husson, “Selecting the number of components in principal component analysis using cross-validation approximations,” *Comput. Stat. Data Anal.*, vol. 56, no. 6, pp. 1869–1879, Jun. 2012, doi: 10.1016/J.CSDA.2011.11.012.

[183] P. Pudil and J. Novovičová, “Novel Methods for Feature Subset Selection with Respect to Problem Knowledge,” *Featur. Extr. Constr. Sel.*, pp. 101–116, 1998, doi: 10.1007/978-1-4615-5725-8_7.

[184] L. Yu, H. L.-P. of the 20th international conference on, and undefined 2003, “Feature selection for high-dimensional data: A fast correlation-based filter solution,” *aaai.org*.

[185] W. M. Hartmann, “Dimension reduction vs. variable selection,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3732 LNCS, pp. 931–938, 2006, doi: 10.1007/11558958_113.

[186] M. Z. F. Audah, T. S. Chin, Y. Zulfadzli, C. K. Lee, K. Rizaluddin, and K. Audah, M.Z.F.; Chin, T.S.; Zulfadzli, Y.; Lee, C.K.; Rizaluddin, “Towards Efficient and Scalable Machine Learning-Based QoS Traffic Classification in Software-Defined Network,” *Mob. Web Intell. Inf. Syst. Springer Cham, Switz.*, pp. 217–229, 2019.

[187] G. S. A.A. Afuwape, Y. Xu, J.H. Anajemba, A. A. Afuwape, Y. Xu, J. H. Anajemba, and G. Srivastava, “Performance evaluation of secured network traffic classification using a machine learning approach,” *Comput. Stand. Interfaces,78*, p. 103545, 2021, doi: 10.1016/j.csi.2021.103545.

[188] G. Dong, Y. Jin, S. Wang, W. Li, … Z. T.-2019 20th A.-P., and undefined 2019, “Db-kmeans: an intrusion detection algorithm

based on dbscan and k-means," *ieeexplore.ieee.org*, Accessed: Apr. 26, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8892910/.

[189] L. Zhang, S. Deng, and S. Li, "Analysis of power consumer behavior based on the complementation of K-means and DBSCAN," *2017 IEEE Conf. Energy Internet Energy Syst. Integr. EI2 2017 - Proc.*, vol. 2018-January, pp. 1–5, Jun. 2017, doi: 10.1109/EI2.2017.8245490.

[190] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained K-means Clustering with Background Knowledge," in *International Conference on Machine Learning ICML*, 2001, vol. pages, pp. 577–584, Accessed: Jul. 23, 2022. [Online]. Available: http://www.cs.cornell.edu/home/wkiri/cop-kmeans/.

[191] A. Nair, "Beginner's Guide To K-Means Clustering," 2019. https://analyticsindiamag.com/beginners-guide-to-k-means-clustering/ (accessed Jul. 26, 2022).

[192] L. Bai, X. Cheng, J. Liang, H. Shen, and Y. Guo, "Fast density clustering strategies based on the k-means algorithm," *Pattern Recognit.*, vol. 71, pp. 375–386, Nov. 2017, doi: 10.1016/j.patcog.2017.06.023.

[193] M. Vinicius Brito da Silva, J. Adilson Marques, L. Paschoal Gaspary, and L. Zambenedetti Granville, "Identifying elephant flows using dynamic thresholds in programmable IXP networks," *J. Internet Serv. Appl. Silva al. J. Internet Serv. Appl.*, vol. 11, p. 10, 2020, doi:

10.1186/s13174-020-00131-6.

[194] W. Wang, Y. Sun, K. Salamatian, Z. Li, and Z. Li, “Adaptive Path Isolation for Elephant and Mice Flows by Exploiting Path Diversity in Datacenters,” *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 1, 2016, doi: 10.1109/TNSM.2016.2517087.

[195] M. Hamdan *et al.*, “Flow-Aware Elephant Flow Detection for Software-Defined Networks,” *IEEE Access*, vol. 8, pp. 72585–72597, 2020, doi: 10.1109/ACCESS.2020.2987977.

[196] H. Yahyaoui, S. Aidi, and M. F. Zhaní, “On Using Flow Classification to Optimize Traffic Routing in SDN Networks; On Using Flow Classification to Optimize Traffic Routing in SDN Networks,” 2020.

[197] M. Al-Saadi, A. Khan, V. Kelefouras, D. J. Walker, and B. Al-Saadi, “Unsupervised Machine Learning-Based Elephant and Mice Flow Identification,” *Lect. Notes Networks Syst.*, vol. 284, pp. 357–370, 2021, doi: 10.1007/978-3-030-80126-7_27.

[198] W. Cheng, F. Ren, W. Jiang, K. Qian, T. Zhang, and R. Shu, “Isolating Mice and Elephant in Data Centers,” 2016, Accessed: May 16, 2023. [Online]. Available: http://arxiv.org/abs/1605.07732.

[199] P. K. Mondal, L. P. Aguirre Sanchez, E. Benedetto, Y. Shen, and M. Guo, “A dynamic network traffic classifier using supervised ML for a Docker-based SDN network,” *Conn. Sci.*, vol. 33, no. 3, pp. 693–718, 2021, doi: 10.1080/09540091.2020.1870437.

[200] D. Berrar, “Bayes’ theorem and naive bayes classifier,” *Encycl. Bioinforma. Comput. Biol. ABC Bioinforma.*, vol. 1–3, no. January 2018, pp. 403–412, 2018, doi: 10.1016/B978-0-12-809633-8.20473-1.

[201] L. X. Liao, H.-C. C. Chao, and M.-Y. Y. Chen, “Intelligently modeling, detecting, and scheduling elephant flows in software defined energy cloud: A survey,” *J. Parallel Distrib. Comput.*, vol. 146, pp. 64–78, 2020, doi: 10.1016/j.jpdc.2020.07.008.

[202] S. K. S. Keshari, V. Kansal, S. Kumar, S. K.-W. P. Communications, undefined 2021, and S. Kumar, “A Systematic Review of Quality of Services (QoS) in Software Defined Networking (SDN),” *Wirel. Pers. Commun.*, vol. 116, no. 3, pp. 2593–2614, Feb. 2021, doi: 10.1007/s11277-020-07812-2.

[203] M. Khattar Awad, A. Almutairi, M. Hassan Hafez Ahmed, A. F. Almutairi, I. Ahmad, and M. Khattar Awad mohamad, “Machine learning-based multipath routing for software defined networks,” *Springer*, vol. 29, no. 2, p. 18, Apr. 2021, doi: 10.1007/s10922-020-09583-4.

[204] Y. S. Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, “A survey of networking applications applying the software defined networking concept based on machine learning,” *IEEE Access*, pp. 95397–95417, 2019.

[205] A. Nasir, T. Alyas, M. Asif, and M. N. Akhtar, “Reliability

Management Framework and Recommender System for Hyper-converged Infrastructured Data Centers,” 2020.

[206] M. Apostolaki, E. Zürich, L. Vanbever, and M. Ghobadi, “FAB: Toward Flow-aware Buffer Sharing on Programmable Switches,” 2019, doi: 10.1145/3375235.3375237.

[207] L. Wang *et al.*, “Scheduling with machine-learning-based flow detection for packet-switched optical data center networks,” *J. Opt. Commun. Netw.*, vol. 10, no. 4, pp. 365–375, 2018, doi: 10.1364/JOCN.10.000365.

[208] A. Alghadhban and B. Shihada, “FLight: A fast and lightweight elephant-flow detection mechanism,” in *Proceedings - International Conference on Distributed Computing Systems*, 2018, vol. 2018-July, pp. 1537–1538, doi: 10.1109/ICDCS.2018.00161.

[209] E. T. B. Hong and C. Y. Wey, “An optimized flow management mechanism in OpenFlow network,” in *International Conference on Information Networking*, 2017, pp. 143–147, doi: 10.1109/ICOIN.2017.7899493.

[210] “Mininet.” http://mininet.org/.

[211] K. Kaur, J. Singh, N. G.-I. conference On, and U. 2014, “Mininet as software defined networking testing platform,” *researchgate.net*, 2014, Accessed: Jun. 25, 2022. [Online]. Available: https://www.researchgate.net/profile/Sundara-Kumar-

V/post/how_to_implement_optical_CDMA_using_optisystem_and_matlab_all_together/attachment/5e770593cfe4a7809f8a2b62/AS%3A871773948096512%401584858514455/download/ProceedingsICCCS2014.pdf#page=156.

[212] R. Leão *et al.*, “Using mininet for emulation and prototyping software-defined networks,” *ieeexplore.ieee.org*, 2014, doi: 10.1109/ColComCon.2014.6860404.

[213] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, “In VINI veritas: Realistic and controlled network experimentation,” in *Computer Communication Review*, 2006, vol. 36, no. 4, pp. 3–14, doi: 10.1145/1151659.1159916.

[214] M. Iqbal, K. Zhang, S. Iqbal, I. T.-I. J. C. S. Eng, and undefined 2018, “A fast and reliable Dijkstra algorithm for online shortest path,” *pdfs.semanticscholar.org*, vol. 5, p. 12, 2018, Accessed: Jun. 25, 2022. [Online]. Available: https://pdfs.semanticscholar.org/b745/15006908078bc2efc27844f85a351740b148.pdf.

[215] J. Yi and B. Parrein, “Multipath Extension for the Optimized Link State Routing Protocol Version 2 (OLSRv2),” *No. rfc8218*, 2017, Accessed: Jun. 17, 2023. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8218.

[216] O. T. Mathis M, Semke J, Mahdavi J, “The macroscopic behavior of the TCP congestion avoidance algorithm,” *ACM SIGCOMM Comput.*

Commun. Rev., vol. 27, no. 3, pp. 67–82, 1997.

[217] Thomas J. Hacker; Brian D. Athey, "The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy WideArea Network," *InProceedings 16th Int. Parallel Distrib. Process. Symp. 2002 Apr 15 IEEE.*, pp. 10-pp, 2002.

[218] F. Techniques and I. P. P. Selection, "02/12/12 RFC 5475 - Sampling and Filtering Techniques f or IP Packet Selection," pp. 1–46, 2018.

[219] X. Jiang, S. Liu, S. Naama, F. Bronzino, P. Schmitt, and N. Feamster, "AC-DC: Adaptive Ensemble Classification for Network Traffic Identification," *arxiv.org*, 2023, Accessed: May 16, 2023. [Online]. Available: http://arxiv.org/abs/2302.11718.

[220] B. Nougnanke, Y. Labit, M. Bruyere, U. Aivodji, and S. Ferlin, "ML-based Performance Modeling in SDN-enabled Data Center Networks," *IEEE Trans. Netw. Serv. Manag.*, pp. 1–15, 2022, doi: 10.1109/TNSM.2022.3197789.

[221] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 779–792, 2017, doi: 10.1109/TNET.2016.2614247.

[222] E. Akin and T. Korkmaz, "Comparison of Routing Algorithms with Static and Dynamic Link Cost in Software Defined Networking (SDN)," *IEEE Access*, vol. 7, pp. 148629–148644, 2019, doi:

10.1109/ACCESS.2019.2946707.

[223] X. Feng and H. Chengde, “A Fault-Tolerant Routing Scheme in Dynamic Networks,” *J. Comput. Sci. Technol*, vol. 16, no. 4, 2001.

[224] X. You *et al.*, “Toward Packet Routing With Fully Distributed Multiagent Deep Reinforcement Learning,” *SYSTEMS*, vol. 52, no. 2, p. 855, 2022, doi: 10.1109/TSMC.2020.3012832.

[225] E. Papadogiannaki and S. Ioannidis, “A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures,” *ACM Comput. Surv.*, vol. 54, no. 6, 2021, doi: 10.1145/3457904.

[226] P. Wang, X. Chen, F. Ye, and Z. Sun, “A Survey of Techniques for Mobile Service Encrypted Traffic Classification Using Deep Learning,” *IEEE Access*, vol. 7, pp. 54024–54033, 2019, doi: 10.1109/ACCESS.2019.2912896.

[227] A. Gupta, “Vpn-nonvpn traffic classification using deep reinforced naive bayes and fuzzy k-means clustering,” *Proc. - 2021 IEEE 41st Int. Conf. Distrib. Comput. Syst. Work. ICDCSW 2021*, pp. 1–6, 2021, doi: 10.1109/ICDCSW53096.2021.00008.

[228] M. Shen, J. Zhang, S. Chen, Y. Liu, and L. Zhu, “Machine learning classification on traffic of secondary encryption,” *2019 IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc.*, pp. 1–6, 2019, doi: 10.1109/GLOBECOM38437.2019.9013272.

[229] Z. Liu, D. Gao, Y. Liu, H. Zhang, and C. H. Foh, “An adaptive approach for elephant flow detection with the rapidly changing traffic in data center network,” *Int. J. Netw. Manag.*, vol. 27, no. 6, pp. 1–13, Nov. 2017, doi: 10.1002/nem.1987.

Appendix-1 Table of Features

1. First _packet	2. last_packet	3. total_packets_a2b
4. total_packets_b2a	5. resets_sent_a2b	6. resets_sent_b2a
7. ack_pkts_sent_a2b	8. ack_pkts_sent_b2a	9. pure_acks_sent_a2b
10.pure_acks_sent_b2a	11.sack_pkts_sent_a2b	12.sack_pkts_sent_b2a
13.dsack_pkts_sent_a2b	14.dsack_pkts_sent_b2a	15.max_sack_blks.ack_a2b
16.max_sack_blks.ack_b2a	17.unique_bytes_sent_a2b	18.unique_bytes_sent_b2a
19.actual_data_pkts_a2b	20.actual_data_pkts_b2a	21.actual_data_bytes_a2b
22.actual_data_bytes_b2a	23.rexmt_data_pkts_a2b	24.rexmt_data_pkts_b2a
25.rexmt_data_bytes_a2b	26.rexmt_data_bytes_b2a	27.outoforder_pkts_a2b
28.outoforder_pkts_b2a	29.pushed_data_pkts_a2b	30.pushed_data_pkts_b2a
31.adv_wind_scale_a2b	32.adv_wind_scale_b2a	33.sacks_sent_a2b
34.sacks_sent_b2a	35.mss_requested_a2b	36.mss_requested_b2a
37.max_segm_size_a2b	38.max_segm_size_b2a	39.min_segm_size_a2b
40.min_segm_size_b2a	41.avg_segm_size_a2b	42.avg_segm_size_b2a
43.max_win_adv_a2b	44.max_win_adv_b2a	45.min_win_adv_a2b
46.min_win_adv_b2a	47.zero_win_adv_a2b	48.zero_win_adv_b2a
49.avg_win_adv_a2b	50.avg_win_adv_b2a	51.max_owin_a2b
52.max_owin_b2a	53.min_non.zero_owin_b2a	54.avg_owin_a2b
55.avg_owin_b2a	56.wavg_owin_a2b	57.wavg_owin_b2a
58.initial_window_bytes_a2b	59.initial_window_bytes_b2a	60.initial_window_pkts_a2b
61.initial_window_pkts_b2a	62.ttl_stream_length_a2b	63.ttl_stream_length_b2a
64.missed_data_a2b	65.missed_data_b2a	66.data_xmit_time_a2b
67.data_xmit_time_b2a	68.idletime_max_a2b	69.idletime_max_b2a
70.throughput_a2b	71.throughput_b2a	72.RTT_samples_a2b
73.RTT_samples_b2a	74.RTT_min_a2b	75.RTT_min_b2a
76.RTT_max_a2b	77.RTT_max_b2a	78.RTT_avg_a2b
79.RTT_avg_b2a	80.RTT_stdev_a2b	81.RTT_stdev_b2a
82.RTT_from_3WHS_a2b	83.RTT_from_3WHS_b2a	84.RTT_full_sz_smpls_a2b
85.RTT_full_sz_smpls_b2a	86.RTT_full_sz_min_a2b	87.RTT_full_sz_min_b2a
88.RTT_full_sz_max_a2b	89.RTT_full_sz_max_b2a	90.RTT_full_sz_avg_a2b
91.RTT_full_sz_avg_b2a	92.RTT.full_sz_stdev_a2b	93.RTT_full_sz_stdev_b2a
94.post.loss_acks_a2b	95.post.loss_acks_b2a	96.ambiguous_acks_a2b
97.ambiguous_acks_b2a	98.RTT_min_.last._a2b	99.RTT_min_.last._b2a
100. RTT_max_.last._a2b	101. RTT_max_.last._b2a	102. RTT_avg_.last._a2b
103. RTT_avg_.last._b2a	104. RTT_sdv_.last._a2b	105. RTT_sdv_.last._b2a
106. segs_cum_acked_a2b	107. segs_cum_acked_b2a	108. duplicate_acks_a2b
109. duplicate_acks_b2a	110. triple_dupacks_a2b	111. triple_dupacks_b2a
112. max_._retrans_a2b	113. max_._retrans_b2a	114. min_retr_time_a2b
115. min_retr_time_b2a	116. max_retr_time_a2b	117. max_retr_time_b2a
118. avg_retr_time_a2b	119. avg_retr_time_b2a	120. sdv_retr_time_a2b
121. sdv_retr_time_b2a	122. SYN_pkts_sent_a2b	123. FIN_pkts_sent_a2b
124. SYN_pkts_sent_b2a	125. FIN_pkts_sent_b2a	126. req_1323_ws_a2b
127. req_1323_ts_a2b	128. req_1323_ws_b2a	129. req_1323_ts_b2a

Appendix-2 List of publications

[1] Muna Al-Saadi, Bogdan V Ghita, Stavros Shiaeles, Panagiotis Sarigiannidis: A novel approach for performance-based clustering and management of network traffic flows, IWCMC, 978-1-5386-7747-6/19/$31.00 ©2019 IEEE.

[2] M. Al-Saadi, A. Khan, V. Kelefouras, D. J. Walker, and B. Al-Saadi: Unsupervised Machine Learning-Based Elephant and Mice Flow Identification, Lect. Notes Networks Syst., vol. 284, pp. 357–370, 2021, doi: 10.1007/978-3-030-80126-7_27.

[3] M. Al-Saadi, A. Khan, V. Kelefouras, D. J. Walker, and B. Al-Saadi: SDN-Based Routing Framework for Elephant and Mice Flows Using Unsupervised Machine Learning, Network, 3(1), pp.218-238, 2023.