UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JEANDRO DE MESQUITA BEZERRA

Traffic Engineering in Data Center Networks: Prediction and Scheduling via Randomized Rounding for Elephant Flows

Recife

2020

JEANDRO DE MESQUITA BEZERRA

Traffic Engineering in Data Center Networks: Prediction and Scheduling via Randomized Rounding for Elephant Flows

Work presented to the Graduate Program in Computer Science of the Centro de Informática of the Universidade Federal de Pernambuco in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Concentration Area: Computer Networks.

Advisor: Prof. Dr. Divanilson Rodrigo de Sousa Campelo.

Co-Advisor: Prof. Dr. Críston Pereira de Souza.

Recife

2020

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

B574t Bezerra, Jeandro de Mesquita

Traffic engineering in data center networks: prediction and scheduling via randomized rounding for elephant flows / Jeandro de Mesquita Bezerra. – 2020.
105 f.: il., fig., tab.

Orientador: Divanilson Rodrigo de Sousa Campelo.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências.

1. Redes de computadores. 2. Avaliação de desempenho. I. Campelo, Divanilson Rodrigo de Sousa (orientador). II. Título.

004.6 CDD (23. ed.) UFPE - CCEN 2021 – 42

Jeandro de Mesquita Bezerra

"Traffic Engineering in Data Center Networks: Prediction and Scheduling via Randomized Rounding for Elephant Flows"

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Aprovado em: 25/11/2020.

I dedicate this work to God, to my parents Afonso and Custódia *in memoriam*, to my sisters Janaina and Ana Joecília, to my wife Karine and my son Jorge Benício.

ACKNOWLEDGEMENTS

To God and all spiritual energies for leadership and comfort in every moment that I needed. To Custódia (In memoriam) and Afonso, my parents, my thanks for everything! They have always dedicated their lives to provide us the conditions so that we could achieve our goals. I also thank my sisters Janaina and Joecília. To my wife, Karine Linhares, for all attention, affection, understanding, support, and love received over these years. To my son Jorge Benicio, for making me a more fulfilled and happy person.

To my family and friends who have always been with me and encouraged me to face challenges.

To colleagues who collaborated with this research and lived in Recife. To the former student and friend, Janael, with his example of an inspiring life.

To Divanilson Campelo (advisor) and Críston Souza (co-advisor), for the opportunity, supervision, chances, and support that with their expertise and experience, provided to me the opportunity to work on this project.

To all lecturers of Centro de Informatica (CIN) that, in one way or another, contributed to my academic and personal growth.

To FUNCAP (Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico) for their financial support.

ABSTRACT

Applications and services hosted in large Data Centers account for most of the increase in Internet traffic. Data Center Networks (DCNs) are often designed with a fat-tree topology, allowing multiple paths between any two servers. The most widely adopted solution for flow routing in DCNs is equal-cost multipath (ECMP), which can cause link performance degradation due to the possible occurrence of hash collisions in the presence of flows with many gigabytes of data, called elephants. Such collisions can result in packet discard, which generates packet retransmission, causes additional latency, and further degrades link performance. This thesis proposes a hybrid prediction model by combining aspects of the FARIMA and the Recurrent Neural Network (FARIMA-RNN) models to predict elephant flows on a short-term basis. Besides, we implement an SDN solution based on a randomized rounding heuristic, named RDRH, to schedule elephant flows in DCNs. We employ a linear programming formulation that provides in polynomial time lower bounds for balancing elephant flows. A methodology based on the rank of the prediction accuracy metrics is applied to compare the hybrid model's performance with the ARIMA, GARCH, RBF, MLP, and LSTM models. Results show that the FARIMA-RNN model presents lower error rates than the other predictors. Furthermore, we evaluate our proposed heuristic performance on an emulated network with Mininet. The experiments show that the RDRH solution presents a performance gain compared to the ECMP and Hedera solutions in the round-trip delay and loss metrics in two of the four evaluated scenarios.

Keywords: Data center networks. Elephant Flows. Prediction. Performance Evaluation. Randomized Rounding.

RESUMO

Aplicações e serviços hospedados em grandes data centers ocasionam um aumento no volume de tráfego da Internet. Redes de data center (DCNs) são frequentemente projetadas com a topologia fat-tree que permitem múltiplos caminhos entre quaisquer dois servidores. A solução mais adotada para roteamento de fluxos em DCNs é o ECMP (equal-cost multipath), que pode causar degradação do desempenho no enlace devido à possibilidade de ocorrência de colisões de hash na presença de fluxos com muitos gigabytes, chamados de elefante. Tais colisões podem ocasionar descarte de pacotes que geram retransmissões causando atrasos adicionais e degradam o desempenho do enlace. Esta tese propõe um modelo híbrido de predição combinando aspectos dos modelos FARIMA e de Redes Neurais Recorrentes, chamado de (FARIMA-RNN), para prever fluxos elefante em curto período. Além disso, implementamos uma solução SDN baseada em uma heurística de arredondamento probabilístico, denominada de RDRH, para escalonar fluxos elefante em DCNs. Uma metodologia baseada no rank da métrica de acurácia de previsão é aplicada para comparar o desempenho do modelo híbrido com os modelos ARIMA, GARCH, RBF, MLP e LSTM. Os resultados mostram que o modelo FARIMA-RNN apresenta taxas de erro menores que os demais modelos. A heurística proposta foi avaliada em uma DCN emulada com o Mininet. Os experimentos mostram que a solução RDRH apresenta ganho de desempenho comparado com as soluções ECMP e Hedera nas métricas atraso de ida e volta e perda em dois dos quatro cenários avaliados.

Palavras-chaves: Redes de data center. Fluxos elefante. Predição. Avaliação de desempenho. Arredondamento probabilístico.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ANN	Artificial Neural Networks
API	Application Programming Interface
AR	Autoregressive
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	Autoregressive Integrated Moving Average
ATM	Asynchronous Transfer Mode
BIC	Bayesian Information Criterion
DCN	*Data Center Networks*
ECMP	Equal-cost multi-path
FARIMA	Fractional ARIMA
FNN	False Nearest Neighbors
FSSM	Flow-Switch Stable Matching
FTR	Fat-Tree Routing
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GPH	Geweke and Porter-Hudak
i.i.d.	independent and identically distributed
IP	Internet Protocol
JE_RProp	Jordan and Elman Resilient Backpropagation
LP	Linear Programming
LRD	Long Range Dependence
LSTM	Long Short-Term Memory
LWR	Locally Weighted Regression
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error

ML	Maximum Likelihood
MLP	Multi-Layer Perceptron
MMFR	Modified Multicommodity Flow Relaxation
MPLS	Multiprotocol Label Switching
NOS	Network Operating System
QoS	Quality of Service
RBF	Radial Basic Function
RDRH	Randomized Rounding Heuristic
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Networks
RPPS	Cloud Resource Prediction and Provisioning Scheme
RProp	Resilient Backpropagation
SDN	Software-Defined Networking
SNNS	Stuttgart Neural Network Simulator
SSTD	Skewed Student's t-distribution
SW	Shapiro-Wilk
TCP	Transmission Control Protocol
TE	Traffic Engineering
TOR	Top of Rack
VNEP	Virtual Network Embedding Problem
WNN	White Neural Network
WSRT	Wilcoxon Signed-Rank tests

LIST OF SYMBOLS

θ	theta
σ	sigma
μ	mi
ϕ	phi
φ	varphi
Ψ	psi
Γ	gamma

CONTENTS

1 INTRODUCTION

In this chapter, we describe the context, problem and motivation of this thesis. Next, we present its objectives, and research questions. Besides, we describe the research methodology and published works related to this thesis. Finally, the thesis structure is defined.

1.1 CONTEXT

Data centers provide vital infrastructures such as clusters, storage, and networking for hosting various applications that benefit from the availability of such resources. Currently, cloud computing services, present mainly in large data centers, account for a significant part of Internet traffic. Because of the dynamic characteristics and unpredictability of the traffic generated by these services, the traffic growth poses challenges to intra- and inter- *Data Center Networks* (DCN). Traffic patterns in DCNs reflect two aspects: bulk transfers with many gigabytes of data, called elephants (e.g., backups, data migration), and low traffic with flows of a few kilobytes, called mice (e.g., web searches, social networking, etc.). The work (ANDERSON; CROVELLA; DIOT, 2004) describes the prevalence of long-tailed distributions in network workloads known as the "elephant and mice" phenomenon. In the literature, results contrast the data center traffic characteristics when hosting different applications (BENSON et al., 2010; ROY et al., 2015).

The demand for solutions that improve and implement Traffic Engineering (TE) in data centers has increased and enabled the dissemination and consolidation of Software-Defined Networking (SDN). By decoupling the control plane and the data plane, SDN allows greater flexibility and adds new features to the network. It has permitted the implementation of use cases to enhance the network's management through traffic control techniques. Proposed architectures such as Hedera (AL-FARES et al., 2010), B4 (JAIN et al., 2013), Presto (HE et al., 2015) have been landmarks to how new networks are designed and managed. Today, large enterprises are adopting SDN in their DCNs due to ease of management and improved network performance. The authors of (DAI et al., 2017) describe the improvements SDN can bring to the DCN environment, and highlight the growth of studies in this area. The authors argue that their work is the first survey with the integration of data center with SDN and addresses works in optimization for resource allocation.

Flow forwarding in a DCN topology, such as fat-tree (MOLLAH et al., 2018), can be modeled as multicommodity flow approaches. The application of simple heuristics that do not consider flow size leads to load imbalance. Proposals that consider multicommodity approaches and the scheduling flow problem in SDN scenarios appear in (BENSON et al., 2011; BRANDT; FOERSTER; WATTENHOFER, 2017). Heuristics based on random rounding techniques are effective in problems of operational research (BARAHONA; CHUDAK, 2005) and computer networks (MANGILI et al., 2014; XU et al., 2017). The randomized rounding provides a solution for an unsplittable multicommodity flow problem, e.g., a solution for an integer programming formulation P with binary variables. Probabilistic approximation, based on randomized rounding technique, is the primary mechanism in approximating unsplittable flow problems (BIALON, 2017). This technique uses linear relaxation L of P, and an optimal solution S of L. Therefore, in S the binary variables of P have values between 0 and 1 that are used as probabilities in a draw sequence to produce an integer solution for P.

The optimization of energy consumption and load balancing schemes are two examples of data center applications to which prediction and multicommodity models can be applied. They use the forecasting measures to aid in the decision-making process (YOON; KAMAL; ZHU, 2017; BASTAM; SABAEI; YOUSEFPOUR, 2018). Short-term (e.g., one minute ahead) and long-term time interval predictions can help in the development of decision support systems. Besides, predicting the size of flows in a DCN can help revise the routing and scheduling required to improve TE (ALVAREZ-HORCAJO et al., 2019).

The length of packets is used as a metric for classification and identification of traffic patterns (CHAO; LIN; CHEN, 2019; ROY et al., 2015). This metric allows the development of TE mechanisms that perform the scheduling (AL-FARES et al., 2010), classification (CHAO; LIN; CHEN, 2019), and forecasting (LI et al., 2016) of elephant flows. TE mechanisms use time series analysis and forecasting to application development. The length of the packets varying over time is a time series that can be used to identify and predict elephant flow traffic. Besides, flow aggregation is one of the techniques that make it possible to optimize resource allocation in a network (KAMIYAMA et al., 2014).

1.2 PROBLEM AND MOTIVATION

The fat-tree topology allows the diversity of paths and the ability to achieve full-bisection bandwidth in DCNs (AL-FARES; LOUKISSAS; VAHDAT, 2008). This diversity not only facilitates the distribution of flows over all paths of equal length, such as Equal-cost multi-path (ECMP), but also supports the handling of failures (THALER; HOPPS, 2000; ZHANG; CUI; ZHANG, 2017). The problem of selecting routes for flow demands in fat-trees can be modeled as a problem close to multicommodity flow, which is an NP-Hard problem (ZHANG; CUI; ZHANG, 2017). Multicommodity flow approaches can be used to model problems of transmission between pairs in the fat-tree topology (MOLLAH et al., 2018). The application of simple heuristics that do not consider flow size leads to load imbalance. For instance, ECMP forwards flows based on the hash value, causing load imbalance in the links because it does not consider the size of the flows for forwarding.

The large volume of traffic generated in data centers brings up performance and Quality of Service (QoS) problems due to elephant flows (HAMDAN et al., 2020; JURKIEWICZ, 2020; XIE et al., 2019). Although mice and elephants phenomenon emerged two decades ago (ANDERSON; CROVELLA; DIOT, 2004; GUO; MATTA, 2001), this research challenge is still in full development, mainly in data centers (ABDELMONIEM; BENSAOU, 2015; WANG et al., 2016, 2016; CHAO; LIN; CHEN, 2019; FU et al., 2020). A concentration of elephant flows in DCNs can lead to load imbalance that affects other flows that are trying to utilize the shared link. The imbalance can lead to packet discards, thus causing the retransmission of the discarded packets. In turn, retransmission causes additional latency and degradation of the link performance. Elephant flows utilize more bandwidth than other flows, what causes load imbalance because some paths can be congested while others are underutilized.

ECMP is the most broadly used hash-based algorithm to optimize multipath switched networks (ZHANG et al., 2018; TSO; PEZAROS, 2013; CHIESA; KINDLER; SCHAPIRA, 2017). To balance traffic flows between a pair of source and destination nodes, ECMP uses several equal-cost paths between them. In most implementations, ECMP relies on a link-state routing protocol to calculate diverse equal-cost shortest paths. This can lead to a load imbalance because the ECMP routing maps them to the same path based on the number of outgoing interfaces, i.e., links associated with the shortest path at the node level along the route, not at the source-destination path level (MEDHI; RAMASAMY, 2017). The path selection is based on the hash of specific fields of the IP packet, such as the source and destination IP addresses, transport port numbers, and protocol. For instance, by applying the function $x \mod n$, where x is the least significant octet of the IP address and n the number of interfaces (HOPPS et al., 2000). A switch locally selects the next hop across multiple paths of equal cost, calculating a hash value based on the $x \mod n$ function return. The concentration of elephant flows can lead to the occurrence of hash collisions that degrade over all network utilization due to router flooding or switch buffers.

In the literature of computing networking, few research studies tackled the multicommodity approach with the randomized rounding heuristic. A feature of randomized algorithms is their structural simplicity (GONZALEZ, 2007). The randomized rounding technique makes efficient approximations and integrates well with Linear Programming (LP). (MANGILI et al., 2014; KLOPFENSTEIN, 2005) showed the use of randomized rounding in computer networking. The work in (MANGILI et al., 2014) proposed a randomized rounding heuristic that scales up to realistic topologies in Information-Centric Networks. (KLOPFENSTEIN, 2005) presented a randomized rounding heuristic to reroute tunnels in Multiprotocol Label Switching (MPLS) networks.

Traffic load imbalance causes an impact on the capital and operational costs in DCNs (GOMEZ et al., 2019; Saber et al., 2020). Therefore, obtaining the forecast of elephant flows allows planning and better provisioning of links, avoiding underutilization or overload. Classical statistical prediction models such as Autoregressive (AR) and Autoregressive Integrated Moving Average (ARIMA) are extensively used for forecasting. In cases in which the traffic carries Long Range Dependence (LRD) properties, these models present poor performance (KATRIS; DASKALAKI, 2015). Fractional ARIMA (FARIMA) has been proposed to overcome ARIMA's limitations because it is most appropriate to describe the existence of LRD (SHENG; CHEN, 2011). In the presence of nonlinear time series, Artificial Neural Networks (ANN) can be applied to traffic forecasting in computer networks (SABBEH et al., 2016; HONG; WANG; HSU, 2013). Recurrent Neural Networks (RNN) are ANN models for processing sequential data. RNN models work well with time series analysis and can also process sequences of variable length (GOODFELLOW; BENGIO; COURVILLE, 2016). Hybrid models can be defined as a proposal in which several models are merged to form a new enhanced model to better forecast results (RAMOS-PEREZ; ALONSO-GONZALEZ; NUNEZ-VELAZQUEZ, 2019; RATHER; AGARWAL; SASTRY, 2015). Artificial intelligence techniques combined with traditional statistics models result in a hybrid model.

It is difficult to obtain current traffic traces from production data centers for analysis because of industrial privacy and confidentiality policy. There is an initiative from the Altoona, Pennsylvania Facebook released a set of real traces of its DCN (ZENG; SHAO, 2016). Trace documentation is available in a private Facebook group called Facebook Network Analytics

Data Sharing[1]. This benefited the research community and enabled the development of this work. Besides, most traditional data centers do not have the infrastructure to support flow-level monitoring and scheduling, thus relying on an accurate traffic prediction to perform short-term/long-term traffic scheduling to aid with load balancing (LI et al., 2016). For instance, if we can predict short-term elephant flow traffic, we can reroute mice flows to non-congested links. Long-term prediction is beyond the scope of this thesis.

Systematic literature reviews about TE with SDN were conducted by (AKYILDIZ et al., 2014; AKYILDIZ et al., 2016; DAI et al., 2017). These reviews pointed to many types of contributions, such as load balancing mechanisms and prediction of traffic patterns in DCNs. The integration of SDN with DCNs enables the development of innovations that make it possible to address the challenges raised in this section. Therefore, we noticed a demand for a prediction model and load balancing mechanisms in DCNs.

1.3 OBJECTIVES

The challenges related to the scheduling and forecasting of elephant flows in DCNs, mentioned above, motivated the investigation of how such processes can be implemented and improved.

The main objective of this thesis is to minimize the imbalance in the data flow in DCNs. This thesis investigates real data traces of a Facebook DCN, considering the packet length as a time series metric for identifying and determining the predictability of elephant flow traffic. Also, this thesis presents the development of an elephant flow scheduling prototype for DCNs based on the randomized rounding heuristic and SDN. To achieve this goal, we defined the following specific objectives:

1. Processing and time series analysis of real traces from the Altoona, Pennsylvania Facebook Data Center to identify scenarios where elephant flow prediction can be made. The packet length is used as a time series metric for identifying and determining the predictability of elephant flow traffic.

2. The proposal of a hybrid prediction model based on FARIMA and RNN models, which is denoted by FARIMA-RNN.

[1] <https://www.facebook.com/groups/1144031739005495>

3. A performance evaluation of the FARIMA-RNN model and its comparison with existing prediction models.

4. The development of a flow scheduling for DCNs based on SDN that uses an approximative algorithm, called Randomized Rounding Heuristic (RDRH), for load balancing.

1.4 RESEARCH QUESTIONS

As described in the previous section, we identified two research challenges to explore with elephant flows in DCNs: short-term prediction and load balancing in the fat-tree. Thus, the following research questions (RQs) are raised in this work:

- RQ1: Which procedures are required to identify elephant flows in real data traces?
- RQ2: How to design a hybrid model for elephant flows prediction in DCNs?
- RQ3: How does the proposed hybrid prediction model compare with the related state-of-the-art models?
- RQ4: How to implement a heuristic for more efficient routing of elephant flows, and how does it compare with related state-of-the-art solutions?
- RQ5: What is the effect of applying the proposed heuristic when it is used with different traffic patterns?
- RQ6: What is the effect of optimizing elephant flow only? Does this improve network performance?

1.5 RESEARCH METHODOLOGY

This thesis was based on a methodological approach suggested in (WAZLAWICK, 2009). The steps of the methodological process adopted in this thesis are:

- Problem identification: data centers host many applications and concentrate a large mass of data. The network infrastructure to support the demand for elephant flow requires load balancing mechanisms that offer scalability. Real-time applications require a low delay. This requirement can be compromised due to the fact that elephant flows cause bottlenecks in routers or switches in DCNs;

- The elaboration of a strategy for elephant flows forwarding in DCNs: we propose a randomized rounding heuristic since the finding of paths for elephant flows in fat-tree topology is very challenging in terms of computational time, due to its NP-Hardness (ZHANG; CUI; ZHANG, 2017). Therefore, there is a clear necessity of formulating heuristics that can efficiently find an approximate solution for elephant flows forwarding;
- The development and implementation of a prototype for scheduling elephant flows in an emulated DCN. We opted for popular network emulation, Mininet, due to its cost, flexibility, and reproducibility of experiments for new services and applications. Besides, it integrates many SDN elements, e.g., controller;
- The development and implementation of a hybrid elephant flow forecasting model using real DCN traces;
- Performance evaluation: planning and execution of a set of experiments to evaluate the FARIMA-RNN and RDRH prototype;
- Analysis and presentation of results to obtain conclusions from the results.

1.6 PUBLISHED WORKS

1.6.1 Related Publications

In this subsection, we list the papers generated by this thesis. Subsection 1.6.2 presents co-authored publications without a direct relation to this work.

- Journal paper:
  - **Performance evaluation of elephant flow predictors in data center networking.**
    Future Generation Computer Systems, s, v. 102, p. 952 – 964, Jan, 2020.
    J. M. Bezerra, A. J. Pinheiro, C. Souza, D. R. Campelo.
    https://doi.org/10.1016/j.future.2019.09.031.
    Highest percentile Scopus (%) - 96th, "Novo Qualis":A1.
    The results of this publication are presented as parts of Chapters 2, 3, and 5;
- Book chapter:

  - **Engenharia de Tráfego em Redes Definidas por Software.**
    Livro de Mini-cursos do SBRC 2016, SBC, 2016.
    J. M. Bezerra, A. J. Pinheiro, M. S. Bonfim, J. A. Suruagy Monteiro, D. R. Campelo.
    http://www.sbrc2016.ufba.br/minicurso/minicurso-3.
    The results of this publication are presented as part of Chapter 2;

- Paper to be submitted:

  - **A randomized rounding approach for scheduling elephant flows in data center networks.**
    To be defined.
    J. M. Bezerra, A. J. Pinheiro, C. Souza, D. R. Campelo.
    Chapters 2, and 4;

1.6.2 Co-authored papers

- Journal paper:

  - **Adaptive packet padding approach for smart home networks: a trade-off between privacy and performance.**
    IEEE Internet of Things Journal, accepted for publication 2020.
    A. J. Pinheiro, P. Freitas de Araujo Filho, J. M. Bezerra, D. R. Campelo.
    https://doi.org/10.1016/j.comcom.2019.05.012.
    Highest percentile Scopus (%) - 99th, “Novo Qualis”:A1;
  - **Identifying IoT devices and events based on packet length from encrypted traffic**
    Computer Communications, v.144, p.8 – 17, 2019.
    A. J. Pinheiro, J. M. Bezerra, C. A. P. Burgardt, D. R. Campelo.
    https://doi.org/10.1016/j.comcom.2019.05.012.
    Highest percentile Scopus (%) - 85th, “Novo Qualis”:A2;

- Conference paper:

- **Packet padding for improving privacy in consumer IoT.**
  IEEE Symposium on Computers and Communications (ISCC) 2018, pp. 1-5, Jun 2018.
  A. J. Pinheiro, J. M. Bezerra, D. R. Campelo.
  https://doi.org/10.1109/ISCC.2018.8538744.
  "Novo Qualis":A3;

- **An OpenFlow-based elastic solution for cloud-CDN video streaming service.**
  IEEE GLOBECOM 2015, pp. 1-7, Dec 2015.
  P. A. L. Rego, M. S. Bonfim, M. Ortiz, J. M. Bezerra, D. R. Campelo, J. Neuman de Souza.
  "Novo Qualis":A1;

1.7 THESIS STRUCTURE

The remainder of this thesis is structured as follows:

- **Chapter 2** provides the background of DCNs, as well as the statistical and pattern recognition prediction methods. It also briefly presents examples of multicommodity flow models and randomized rounding heuristic.

- **Chapter 3** describes the proposed traffic engineering solution and the hybrid prediction model. Finally, it discusses related works;

- **Chapter 4** describes the randomized rounding heuristic for traffic engineering. It also presents implementation details of RDRH experiments and the results of the performance evaluation. It highlights the main shortcomings of the proposals;

- **Chapter 5** analyzes the performance of predictors and the research methodology to identify elephant flow, and data patterns in data centers. Besides, it analyzes the descriptive and inferential statistics through hypothesis tests and explains how the prediction tests were done.

- **Chapter 6** discusses the results obtained in this thesis and makes some final considerations on the development of this work, presenting the main findings and answers to the research questions. Finally, it indicates how future works could improve our approach.

2 BACKGROUND

This chapter defines the central background concepts that are used throughout this thesis. Initially, we present details of the DCNs, including topology, and features of the Facebook DCN. This chapter also includes SDN and traffic engineering paradigms. Then, the time series forecasting models that will be evaluated are described. The prediction models are based on statistical methods and pattern recognition models to forecast elephant flows. In Section 2.3, an example of multicommodity network flow and two examples of randomized rounding heuristic are presented.

2.1 DATA CENTER NETWORKS

In this section, we focus on DCNs, mainly the details of the fat-tree topology. We also describe details of the Facebook DCN since it provides part of the data used in this work. Subsection 2.1.1 presents SDN paradigms and its integration with DCNs. Finally, in Subsection 2.1.2, we describe the traffic engineering in DCNs.

DCNs are a specially designed network with many interconnected nodes that act as servers with a large processing and storage capacities. Real-time applications demand considerable traffic volume and high performance in application response time, and TE approaches make it possible to meet these requirements.

The main demands of a DCN are scalability and flexibility with ease of management. Technology giants like Microsoft, Google, Amazon, and Facebook are working on a better scalability of data centers. It is possible to maintain scalability and make changes with flexibility without requiring a significant re-organization of the data center, such as addressing to the topology (MEDHI; RAMASAMY, 2017).

Fat-tree topologies are multirooted trees with multiple paths between host pairs. A fat-tree topology is typically associated with the number of pods it has, which are small network units down from the core switches that implement an infrastructure to allow performance scalability. A 4-pod fat-tree is illustrated in Figure 1. They are numbered from left to right as Pod-0 to Pod-$(k-1)$ (MEDHI; RAMASAMY, 2017). For each k-port switch, there are k pods formed by two layers: lower pod switches (edge switches), and the upper pod switches (aggregation switches). The k pods are interconnected by $(k/2)^2$ core switches. Edge switches manage

$(k/2)$ hosts, providing connectivity among end hosts and aggregation switches. Core switches are on top and provide inter-pod communication. The leaf nodes of this hierarchy represent hosts. The advantages of fat-tree are cost, high scalability, and multipath forwarding.

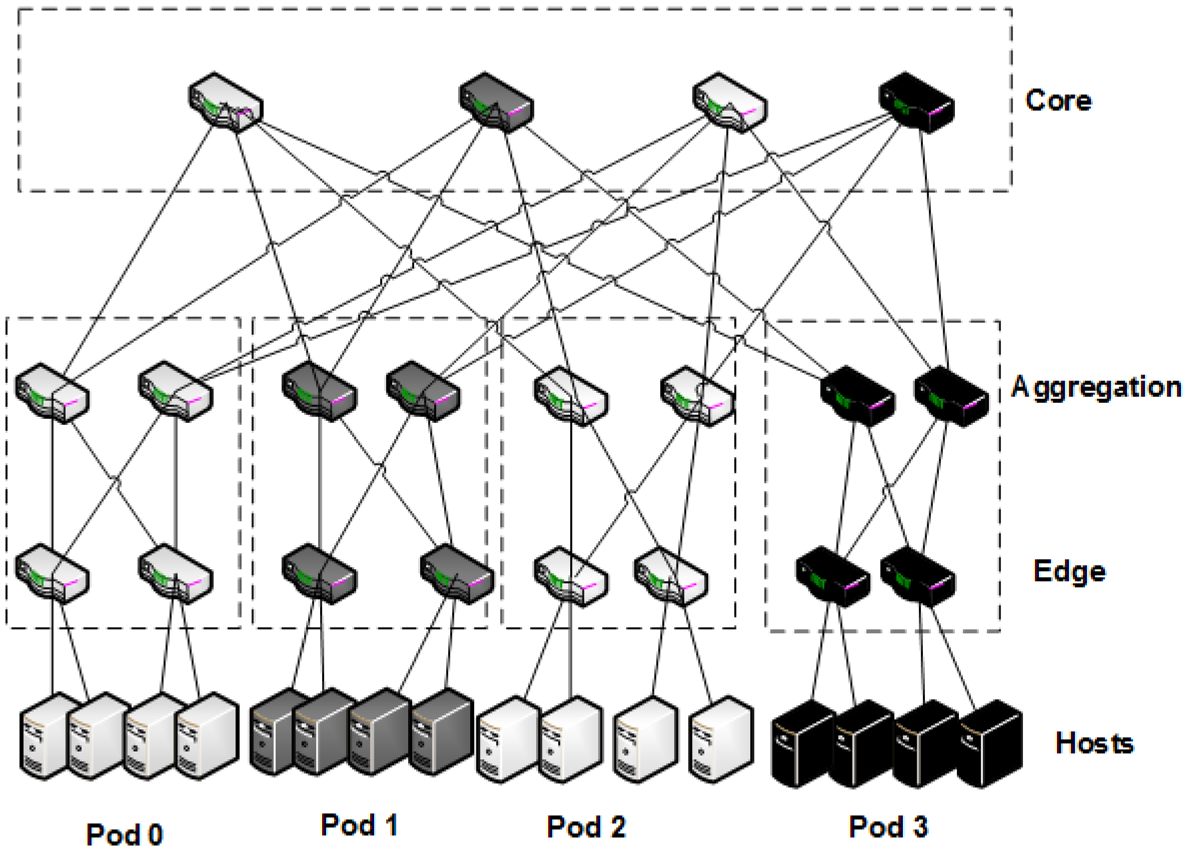


Figure 1 – Fat Tree topology. Adapted from (VAHDAT et al., 2010)

Facebook developed a fat-tree model to increase the bisection bandwidth and performance scalability to support the high demand for applications running in its data center. The bisection bandwidth refers to the bandwidth available between two topology partitions and can be defined as the maximum capacity between any two servers (VARMA, 2015). The internal data center traffic, called machine-to-machine traffic, has experienced an exponential growth (ANDREYEV, 2014). This factor particularly challenges the infrastructure design to support rapid growth, scalability, and planning for new large data center fabrics. Large production networks such as those of Facebook (ANDREYEV, 2014) and Google (SINGH et al., 2015) use 3-tier or more complex Clos topologies (WANG et al., 2019).

Figure 2 shows a pod, which consists of a set of four fabric switches, with each Top of Rack (TOR) having 4 x 40 Gbps uplinks. Moreover, the pod has 48 server racks, the same configuration for all pods. Each Facebook data center contains multiple logical clusters. The work (ANDREYEV, 2014) highlights that the use of clusters was initiated to overcome network

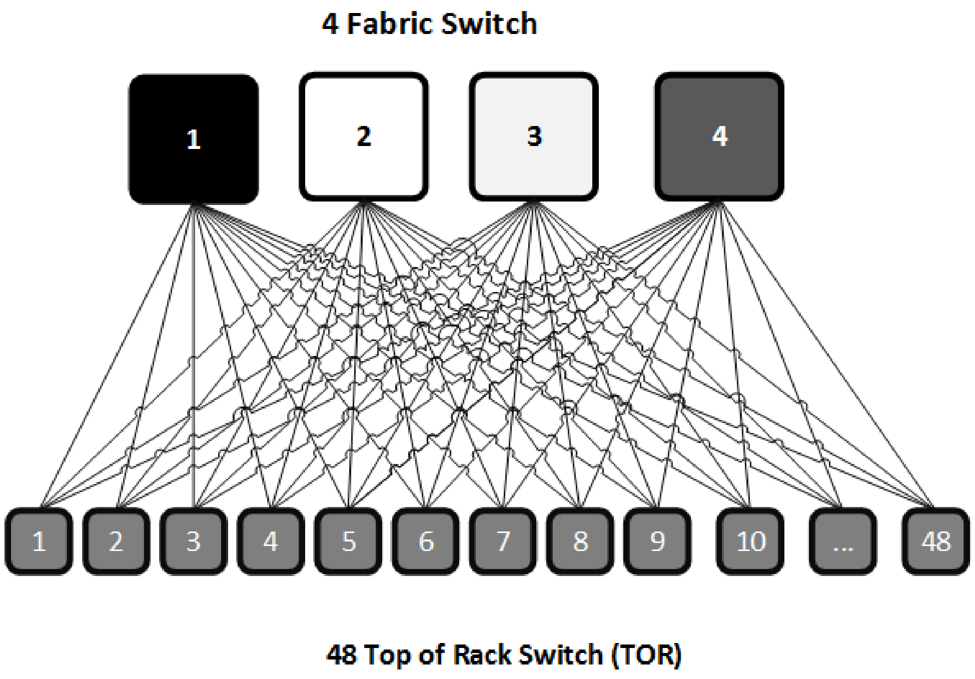


Figure 2 – Sample pod. Adapted from (ANDREYEV, 2014)

limitations, such as port density in switches. Facebook's DCN fabric utilizes virtual clusters, which correspond to the database, web, and Hadoop applications.

(XIA et al., 2017) defines topologies as switch centric, those in which the switches are capable of building the network and transmitting data. In addition to the fat-tree, the other switch centric topologies are: VL2 (GREENBERG et al., 2011), Aspen tree (WALRAED-SULLIVAN; VAHDAT; MARZULLO, 2013), and Jellyfish (SINGLA et al., 2012).

VL2 also uses commodity switches to form a three-layer tree-based topology. The difference with fat-tree is that the links between core switches and aggregation switches generate a complete bipartite graph, and each edge switch is connected to two aggregation switches.

Aspen tree is a variation of traditional fat-tree by disconnecting links at a given level and repurpose them as redundant links for added fault tolerance at the same level. This action generates a tradeoff in scalability, i.e., reduced host count (XIA et al., 2017).

Jellyfish builds a degree-bounded random regular graph at the edge layer. In particular, parts of the edge switches' ports are used to connect servers, and all the remaining ports are used to form a random graph of the switches. The advantage over the fat-tree is that a server can reach more servers in a few hops.

2.1.1 Data Center Networks with Software Defined Networking

The main characteristic of SDN is the separation of control and data planes in the network (FEAMSTER; REXFORD; ZEGURA, 2013). In SDN networks, the control plane, represented by an entity called controller, can decide how to handle traffic. The controller can run on commercial servers, separate from the network equipment. The data plane, embedded in the network devices, can route the data according to established rules. These rules are created and managed by the controller, which performs direct control over the data devices through an Application Programming Interface (API) (KREUTZ et al., 2014; KREUTZ et al., 2015).

Software-Defined Networking facilitates network resource allocation in DCNs. The integration of these paradigms enables and eases network application development. The traditional network is constrained to provide dynamic and complex network rules, large-scale parallel processing algorithms, and a standard and open interface for network devices. To meet these DCN requirements, the research community has significantly employed the SDN paradigm to develop new traffic engineering applications (DAI et al., 2017; XIA et al., 2017). With the SDN controller's APIs, the DCN operator and tenants can program flow tables of hardware and software switches for flexible network services with lower cost and higher efficiency, implementing functions such as creating and configuring private networks and QoS.

A layered structure represents the SDN architecture, as illustrated in Figure 3 depicts the three layers with two planes and the APIs interfaced among them. The data plane contains physical network components, which produce the data path. The control plane has a Network Operating System (NOS) also known as a controller, which executes rules on data plane devices. These rules and policies are designed in the Application of SDN architecture. Communication among these planes is set by using well-defined APIs. The Northbound API is the link for the communication between the application layer and the control plane. This API provides a guide for developing network applications, including traffic engineering and QoS (ZHANG et al., 2020; WANG et al., 2020). The Southbound API is capable of communicating the data plane with the control plane. The path diversity of fat-tree DCNs allows the potential use of SDN in optimizing network traffic (DAI et al., 2017).

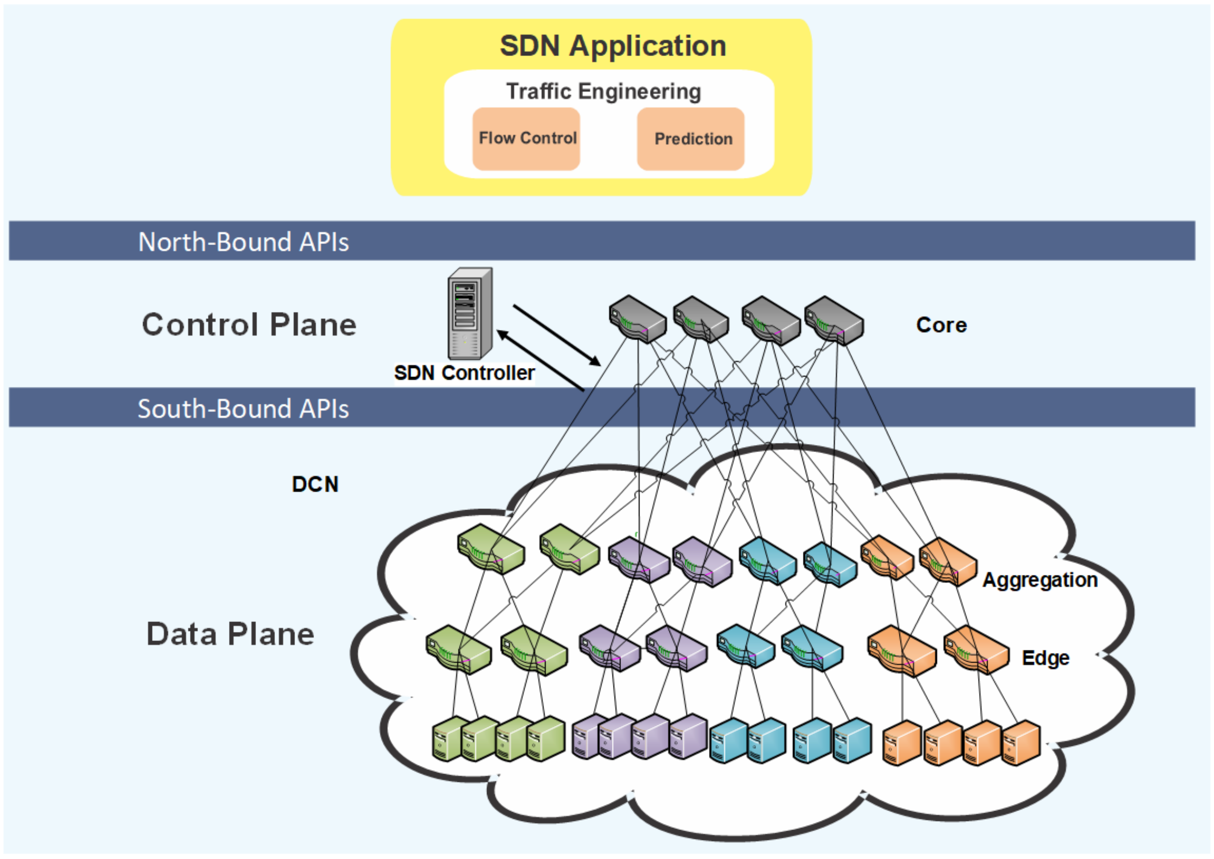


Figure 3 – Data center with SDN architecture.

2.1.2 Traffic Engineering in DCNs

In this subsection, we discuss traffic engineering for intra-domain DCN. Traffic engineering aims to optimize an operational network to meet performance requirements, yet network resources are well employed. The main aim of Internet Protocol (IP) TE is to determine link metrics (weights) optimally so that the network's primary goal such as reducing congestion or delay is met through efficient traffic flows (MEDHI; RAMASAMY, 2017). Traffic engineering was developed for all legacy network technologies since Asynchronous Transfer Mode (ATM) to current paradigms.

SDN enables the development of applications in various approaches of TE, such as flow management and traffic analysis (AKYILDIZ et al., 2014). We highlight issues such as performance optimization, prediction, and scheduling of the transmitted data (AKYILDIZ et al., 2016; CUI; YU; YAN, 2016). These issues are research challenges in cloud computing, which demands large-scale DCNs. Besides, flow scheduling in the fat-tree topologies is challenging due to the NP-hard complexity of managing elephant flow load. In Figure 3, we illustrate traffic engineering applications, such as flow control and traffic prediction, with SDN for DCN. The elements of the DCN and SDN architectures are the target of the TE applications. Details of the proposed TE architecture are in the next chapter, in Section 3.1.

Figure 4 illustrates two scopes to TE with flow management, traffic analysis, and prediction. These scopes were used as a background for this research. Flow management is essential to address the elephant flows problems investigated in this thesis. In the SDN paradigm, the centralized management function in the controller brings a series of challenges. The increase in traffic volume increases the complexity of managing flows and causes loss of network performance. This is a tradeoff because SDN makes management more flexible by developing applications like the previous ones, but they generate load on the SDN controller. Another element of DCNs that has limitations is the switch, as it has limited memory resources. Therefore the increase in the number of flow entries in the switches causes memory consumption problems. These drawbacks are relevant to TE in SDN because they direct the development of research and applications to flow management. This thesis focus is on the areas of traffic prediction and link load balancing in DCNs.

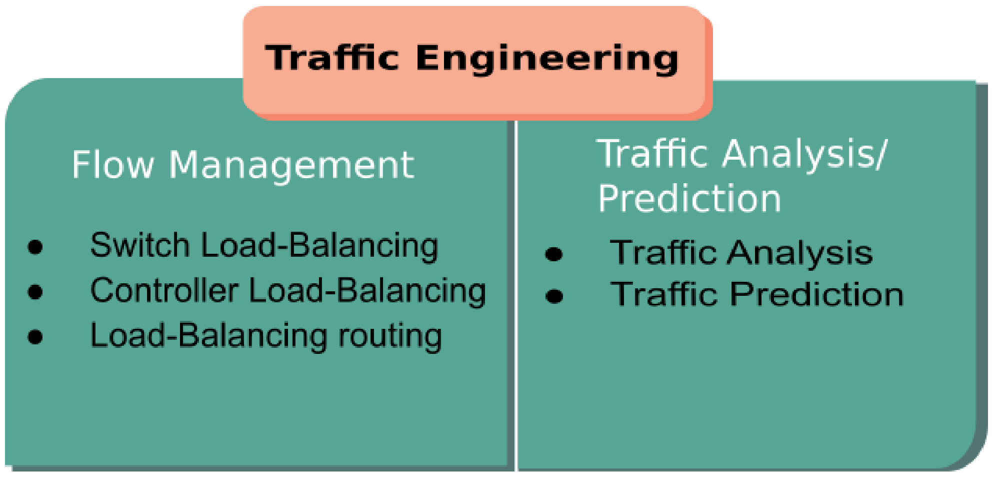


Figure 4 – The scope of traffic engineering areas selected in this thesis.

Traffic analysis is related to monitoring the complexity of SDN environments operation. Traffic pattern analysis and monitoring allow, for instance, the detection of elephant flows and the prediction of congestion. This analysis also makes it possible to predict resource allocation and detect network invariants, that is, bugs in the software that cause failures in the network's operation.

2.2 TIME SERIES PREDICTION WITH STATISTICAL METHODS

The applicability of time series analysis and prediction methods covers many areas of computer networking. The definition of time series is a sequence of observations on a variable of interest, e.g., flow size. The values of the series in pairs can be represented by y_t and y_{t+k}, which are separated in the same time interval k called lag (MONTGOMERY; KULAHCI, 2015). The quantitative forecasting techniques make regular use of historical data and a prediction model (WEI, 2006). The model searches for patterns (e.g., randomness, trends, and seasonality) in the data series, then analyzes the past and present to identify future patterns. The selection criteria for the models described in the following sections were based on the patterns of the analyzed data. These models are used in the literature of traffic forecasting on computer networks (KATRIS; DASKALAKI, 2015; OTOSHI et al., 2015; TONG; WANG; YU, 2009; ANAND; SCOGLIO; NATARAJAN, 2008). The models included in this thesis are in the class of regression and general time series.

2.2.1 Non-seasonal ARIMA model

The combination of differencing with autoregressive models and a moving average model generates a non-seasonal ARIMA model (HYNDMAN; ATHANASOPOULOS, 2017). ARIMA models propose to describe the autocorrelation in the data, that is, the measurement of the linear relationship between the lagged values of a time series. The model is described as

$$Z'_t = c + \phi_1 Z'_{t-1} + \cdots + \phi_p Z'_{t-p} + \theta_1 a_{t-1} + \cdots + \theta_q a_{t-q} + a_t, \tag{2.1}$$

where Z'_t is the differencing series, c is a constant, ϕ is the parameter of the autoregressive models, θ is the parameter of the moving average models, and a_t is white noise. Eq. (2.1) can be written in backshift notation (B has the effect of shifting the data back dth-order period) as

$$\phi_p(B)(1-B)^d Z_t = \theta_0 + \theta_q(B) a_t, \tag{2.2}$$

where the stationary Autoregressive (AR) operator $\phi_p(B) = (1 - \phi_1 B - \cdots - \phi_p B^p)$ and the Moving Average (MA) $\theta_q(B) = (1 - \theta_1 B + \cdots + \theta_q B^q)$ share no common factors (WEI, 2006). This description is known as the ARIMA (p, d, q) model, where p is the order of the autoregressive part, d is the degree of the first differencing, and q is the order of the average moving part.

2.2.2 GARCH model

The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model is used for series modeling, mainly in cases where it is heteroscedastic, that is, the variance is serially correlated. Non-stationary series have periods of volatility during which there is an apparent change in variance (WEI, 2006).

According to Wei (WEI, 2006), the GARCH model is an extension of the Autoregressive Conditional Heteroskedasticity (ARCH) model. ARCH considers that the error's conditional variance is related not only to the square of the errors, but also to the past conditional variance. The generalization of the error is calculated as

$$n_t = \sigma_t \epsilon_t, \tag{2.3}$$

where ϵ_t are independent and identically distributed (i.i.d.) random variables with mean 0 and variance 1; they are independent of past realizations of n_{t-i}. The following equation describes the conditional variance:

$$\sigma_t^2 = \theta_0 + \phi_1 \sigma_{t-1}^2 + \cdots + \phi_p \sigma_{t-p}^2 + \theta_1 n_{t-1}^2 + \cdots + \theta_q n_{t-q}^2, \tag{2.4}$$

where ϕ_i and θ_j $(i = 0, 1, \ldots, p; j = 1, \ldots, q)$ are parameters of the model(s). To assure $\sigma_t^2 > 0$, assume that $\theta_0 > 0$ and that ϕ_i and θ_j are non-negative (WEI, 2006). The model for the error term described in (2.3) with the property (2.4) is known as GARCH (p, q).

2.2.3 FARIMA model

In the previous description of the ARIMA model, d is an integer. A Fractional ARIMA (FARIMA) model is an extension of the ARIMA (p, d, q), allowing the fractional parameter d to take real values (KATRIS; DASKALAKI, 2018; WEI, 2006). It is formulated as

$$\Phi_p(B)(1 - B)^d Z_t = \Theta_q(B) a_t, \tag{2.5}$$

for some $-0.5 < d < 0.5$, where $\Phi_p(B)\Theta_q(B) \neq 0$ for $|B| \leq 1$ and a_t is a white noise process with zero mean and constant variance σ_a^2 to be stationary (WEI, 2006). Considering, without loss of generality, the case

$$(1-B)^d Z_t = a_t. \tag{2.6}$$

if Eq. (2.6) is stationary, then it is expressed as

$$Z_t = (1-B)^{-d} a_t = \sum_{j=0}^{\infty} \Psi_j a_{t-j}, \tag{2.7}$$

i.e,

$$(1-B)^{-d} = \sum_{j=0}^{\infty} \Psi_j B^j, \tag{2.8}$$

such that Ψ_j is square-summable. From Taylor series expansion, the general binomial formula is defined as (WEI, 2006)

$$(1-B)^{-d} = \sum_{j=0}^{\infty} \binom{-d}{j} (-B)^j = \sum_{j=0}^{\infty} \Psi_j B^j, \tag{2.9}$$

where

$$\Psi_j = (-1)^j \binom{-d}{j} = (-1)^j \frac{(-d)(-d-1)\dots(-d-j+1)}{j!} = \frac{\Gamma(j+d)}{\Gamma(j+1)\Gamma(d)}$$

and $\Gamma(\cdot)$ is the gamma function.

2.2.4 Time Series Prediction with Pattern Recognition

Pattern recognition is an area of artificial intelligence that encompasses algorithms which focus is learning through training. Among the categories of pattern recognition is that of Artificial Neural Networks (ANNs).

ANNs have multiple applications across diverse areas of knowledge; they are widely used for nonlinear processes with unknown functional form (GOOIJER, 2017). There are other architectures in the literature, but in this work, because these models work with non-linearity in the data, the Multi-Layer Perceptron (MLP), Radial Basic Function (RBF) (VIEIRA; COSTA; GONÇALVES, 2013; KATRIS; DASKALAKI, 2015), and Recurrent Neural Networks (RNNs) (GOODFELLOW; BENGIO; COURVILLE, 2016; WANG; ZHANG; ZHANG, 2012; TONG; WANG; YU, 2009) are used as benchmarks.

Neural networks can be classified into dynamic and static networks. Static or feed-forward networks do not possess delay or feedback elements. On the contrary, the output of dynamic networks depends on the current and previous inputs or outputs of the network. It has recurrent, i.e. feedback connections, which means that the current output is a function of outputs at previous times (ARDALANI-FARSA; ZOLFAGHARI, 2010; DEMUTH et al., 2014).

2.2.5 Feed-Forward Neural Networks

One of the models most widely used are the feed-forward networks or MLP, which aim to approximate some function f^*. For example, the predictor $y = f^*(\mathbf{x})$ maps an input x to a predicted value $\hat{y}$. A feed-forward network defines the mapping $\mathbf{y} = f(\mathbf{x}; \theta)$ and learns the value of the parameters θ, resulting in the approximation of the best function (GOODFELLOW; BENGIO; COURVILLE, 2016). The training set specifies directly what the output layer must do at each point x; it must produce a value that is close to y. The learning algorithm controls how to use those layers to generate the desired output.

The design decision parameters are: the input variables, the number of hidden layers, and the number of nodes for each layer of each ANN used by Katris and Daskalaki (KATRIS; DASKALAKI, 2015). Each ANN includes one input layer, one hidden layer, and one output node. A Sigmoid is a mathematical function which has a characteristic S-shaped curve. In (2.10), it is used to activate a function at a hidden layer:

$$S(t) = \frac{1}{1 + e^{-t}}. \tag{2.10}$$

For training, a gradient descent algorithm optimizes the input and weights of the network. The gradient is computed using the back-propagation technique, in which the cost information flows backward through the network (GOODFELLOW; BENGIO; COURVILLE, 2016).

Another commonly applied ANN feed-forward network is the RBF (KATRIS; DASKALAKI, 2015). The design used in this work consists of one input layer, one hidden layer with nonlinear activation function that for this work is the Gaussian basis function in Eq. (2.11), and one output node:

$$\phi(x, c, \sigma) = \exp(-\|x - c\|^2 / 2\sigma^2), \tag{2.11}$$

where the center $c \in R^m$ is the mean and σ the width of the function. Finally, a linear transformation is performed going from the hidden layer to the output. Eq. (2.12) presents the output of the RBF network:

$$\hat{y} = \sum_{i=1}^{N} w_i \phi(x, c, \sigma) + w_0, \tag{2.12}$$

where w_0 is a bias term and $\mathbf{w_i}$ is the weight between the hidden and output layers, whereas N is the number of hidden neurons. One of the advantages of the RBF network is the flexibility for function approximation. It has centers adequately distributed across the range of network inputs (DEMUTH et al., 2014).

2.2.6 Recurrent Neural Networks

RNNs are a class of neural networks for processing time series data that can scale to much longer sequences than would be practical for networks without sequence-based specialization (GOODFELLOW; BENGIO; COURVILLE, 2016). Because RNNs have memory, they can be trained to learn sequential or time-varying patterns (DEMUTH et al., 2014). RNN consists of a hidden layer h and an extra output layer y which both operate on a variable-length sequence $\mathbf{x} = (x_1 \dots x_T)$. At each time step t, the hidden layer h_t is updated by

$$H_t = \phi(\mathbf{b_h} + x\mathbf{w_x} + h_{t-1}\mathbf{w_h}) \tag{2.13}$$

$$\hat{Y} = \mathbf{b_o} + h_t\mathbf{w_o}, \tag{2.14}$$

where the parameters are the bias vectors $\mathbf{b_h}$ and $\mathbf{b_o}$ along with the weight matrices $\mathbf{w_x}$, $\mathbf{w_h}$ and $\mathbf{w_o}$ (GOODFELLOW; BENGIO; COURVILLE, 2016). ϕ is a non-linear activation function.

The main difference between Eqs. (2.13), (2.14), and the classical ANN previously described is that the hidden layer of information from the previous period, $h_{t-1}w_h$, is added. An advantage of recurrent networks is the simultaneous sharing of parameters across different parts of a model.

Figure 5 shows an example of RNN composed of four layers: the input layer, the recurrent layer, the hidden layer, and the output layer. The number of nodes of the recurrent layer is always the same as that of the hidden layer. The dashed lines represent the input that recurrent nodes receive from hidden nodes. Only one neuron is used in the output layer.

One of the disadvantages of RNNs is that they should theoretically maintain at time t information about inputs seen many times before. In practice, such long-term dependencies are

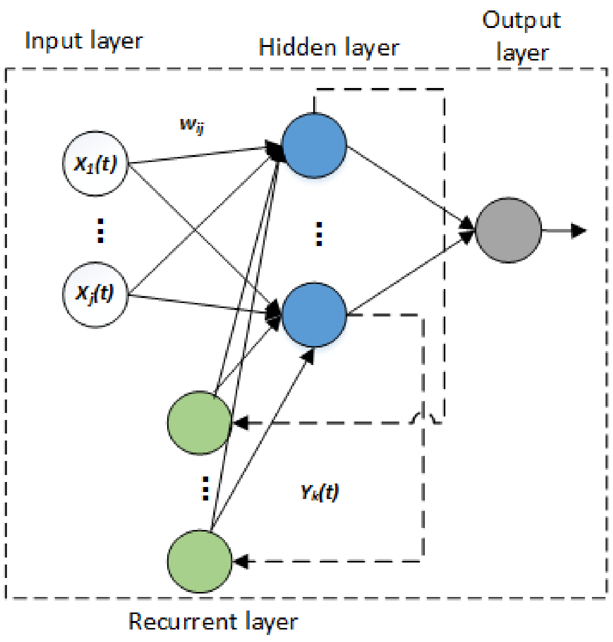


Figure 5 – An RNN whose recurrence is the feedback connection from the output to the hidden layer.

challenging to learn (CHOLLET; ALLAIRE, 2018). Long Short-Term Memory (LSTM) networks are designed to solve this problem (HOCHREITER; SCHMIDHUBER, 1997a).

An LSTM RNN is composed of units called memory blocks. Each memory block contains memory cells with self-connections saving the temporal state of the network, in addition to appropriate multiplicative units called gates to control the flow of information. Each memory block contains an input gate to control the flow of input activations into the memory cell, an output gate to control the output flow of cell activations into the rest of the network and a forget gate. An LSTM network gets an input sequence $x = (x_1, \dots, x_T)$ to an output sequence $y = (y_1, \dots, y_T)$ by computing the network unit activations using the following equations iteratively from $t = 1$ to T (SAK; SENIOR; BEAUFAYS, 2014):

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \tag{2.15}$$

$$f_t = \sigma(W_{fx}x_t + W_{mf}m_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.16}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \tag{2.17}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \tag{2.18}$$

$$m_t = o_t \odot h_{(c_t)} \tag{2.19}$$

$$y_t = \varphi(W_{ym}m_t + b_y) \tag{2.20}$$

where $i, f, \mathbf{c}$, and o are, respectively, the input gate, forget gate, cell activation vectors, and output gate. The output activation vector is $\mathbf{m}$, and $\odot$ is the element-wise product of the vectors. The cell input and cell output activation functions are g and h, respectively; in general both are $\tanh$. φ is the network output activation function. The terms $\mathbf{b}$ denote bias vectors and the terms $\mathbf{W}$ denote weight matrices. σ is the logistic sigmoid function.

2.2.7 Hybrid FARIMA-MLP Model

In 2015, Katris and Daskalaki (KATRIS; DASKALAKI, 2015) presented a hybrid time series forecasting model using both FARIMA and ANN models. The proposed approach first fits a FARIMA model to the data and then creates a neural network with the residuals.

The FARIMA model is fit to each data set following the same procedure as described in (KATRIS; DASKALAKI, 2015):

1. All the samples are normalized. The sample mean $\bar{x}$ of the traffic trace is subtracted from each observation to convert it to a zero-mean data series.

2. The order (p, q) of the model is specified using the lowest Bayesian Information Criterion (BIC). In this work, the order of autoregressive and the moving average components' order will be less than or equal to 4 $(0 \leq p \leq 4, 0 \leq q \leq 4)$.

3. Having set the order of the model, and under the assumption that the stationary, fractionally integrated series follows a Student's t-distribution (KATRIS; DASKALAKI, 2015), the rest of the parameters (d, φ_i, θ_j) are estimated using the Geweke and Porter-Hudak (GPH) estimator for d and a Maximum Likelihood (ML) methodology for the others.

The time series of the residuals from a fitted FARIMA are used to construct the ANN. See 5.2 for a description of the ANN creation procedure. In each step, the two models' forecasts are added together in a single prediction for one period ahead.

2.3 MULTICOMMODITY NETWORK FLOW AND RANDOMIZED ROUNDING

2.3.1 Multicommodity Network Flow

Multicommodity flow is essential to address traffic engineering problems because this approach allows us to design new mechanisms to forward flows in the best way. The fat-trees can be modeled as a graph. We adapt the problem of selecting routes for data traffic to a problem that approximates the multicommodity flow. In this modeling, the switches are vertices, the links are the edges, and the network flows are the elephant flows.

Figure 6 illustrates examples of graphs in two flow routing situations. The circles are the vertices, and the lines connecting them are the edges with the capacities. In these graphs, there are two flows: the red flow, originating in d, the destination in g and size 2, and the blue one, originating in a, the destination in g and size 7. Two useful ways of allocating these flows are presented in figures 6a and 6b. The solution in Figure 6a shows the edge that connects the vertex e and the vertex g is saturated, as it has capacity 9 and is being used by two flows, size 2 and 7.

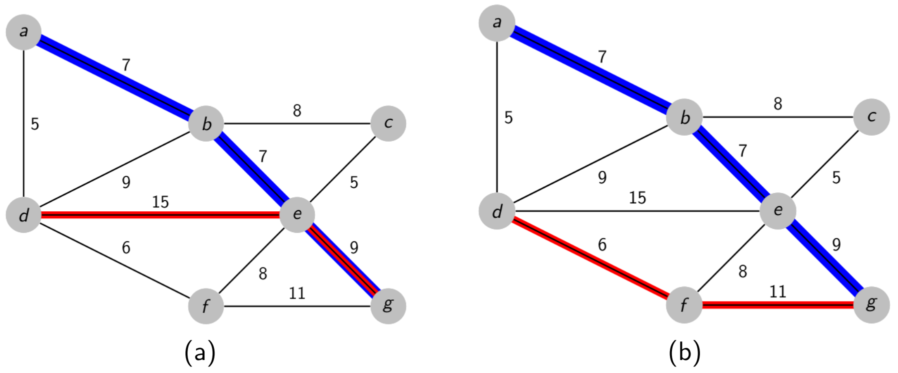


Figure 6 – Multicommodity flow example

2.3.2 Fat-Tree Routing Problem

We adapt the problem of selecting routes for data traffic to a problem that approximates the multicommodity flow. In this modeling, the switches are vertices, the links are the edges, and the network flows are the elephant flows.

The problem of routing flows in fat-tree topologies, called Fat-Tree Routing (FTR), consists in: given a directed graph $G = (V, E)$, where the edges $(u, v) \in E$ have a capacity $c(u, v) \geq 0$.

For each demand i we have a tuple (s_i, t_i, d_i) indicating that d_i units of flow must be sent from the node s_i to the node t_i. ℓ indicates the number of demands, and $f_i(u, v)$ is the flow of demand i passing through the link.

1. **Link Capacity**: For each edge (u, v), the sum of flows passing through (u, v) must not exceed its capacity $c(u, v)$.

$$\sum_{i=1}^{\ell} f_i(u,v)d_i \leq c(u,v), \forall (u,v) \in E \tag{2.21}$$

2. **Flow conservation in transition vertices**: For each vertex, except s_i and t_i, the total flow entering the node is equal to the total flow leaving the node.

$$\sum_{w \in V} f_i(w,u) = \sum_{w \in V} f_i(u,w), \forall i \in \{1, 2, \ldots, \ell\} \tag{2.22}$$

3. **Flow conservation in the vertices of origin**: For the start node s_i of each demand i, the total flow leaving this node is one unit more than the total flow entering, since this node is producing the flow of demand i.

$$\sum_{w \in V} f_i(s_i,w) - \sum_{w \in V} f_i(w,s_i) = 1, \forall i \in \{1, 2, \ldots, \ell\} \tag{2.23}$$

4. **Flow conservation at destination vertices:** For the end node t_i of each demand i, the total flow entering this node is one unit more than the total flow leaving, since this node is consuming the flow of demand i.

$$\sum_{w \in V} f_i(w,t_i) - \sum_{w \in V} f_i(t_i,w) = 1, \forall i \in \{1, 2, \ldots, \ell\} \tag{2.24}$$

For clarity, Table 1 summarizes the notation used throughout this chapter.

The objective function, described in the next subsection, seeks to balance the links' residual capacity. Let the use of a link be the sum of the flows that pass through it, and the residue the difference between its capacity and use. Thus, the goal is to balance the residues of the links by maximizing the smallest residue.

2.3.3 Randomized Rounding

A feature of randomized algorithms is their structural simplicity (GONZALEZ, 2007). The randomized rounding method makes efficient approximations and integrates well with LP.

Parameters	**Explanation**
E	Set of links
V	Set of switches
s_i	Source switch of demand i
t_i	Destination switch of demand i
d_i	Demand volume of demand i
$c(u,v)$	Capacity of link (u,v)
Decision variables	
$f_i(u,v)$	Flow of demand i passing through the link (u,v)

Table 1 – Summary of the notation used in RDRH.

(WILLIAMSON; SHMOYS, 2011) describe an example of the randomized rounding that uses the set cover problem.

The set cover problem is: given a set of elements $E = \{e_1, e_2, \ldots, e_n\}$ and subsets $S_1, S_2, \ldots, S_m \subseteq E$ with weights $w_1, w_2, \ldots, w_m$, find a collection of subsets, so that the union contains all the elements of E and has the least possible weight.

This problem can be solved using integer linear programming (specific to the case of LP, where the variables are integers) with the following modeling:

$$\text{minimize} \sum_{j=1}^{m} w_j x_j \tag{2.25}$$

subject to:

$$\sum_{j:e_i \in S_j} x_j \geq 1 \qquad i = 1, \ldots, n \tag{2.26}$$

$$x_j \in \{0,1\} \qquad j = 1, \ldots, m \tag{2.27}$$

Inequality 2.26 ensures that for each $e_i \in E$ element, at least one set S_j, which contains e_i, will be in the solution.

In this model, the variable x_j is worth 1 if S_j is in the solution. However, finding the optimal solution for this model is an NP-Complete problem. Performing a linear relaxation in this model, by replacing Equation 2.27 with Inequality 2.28, we obtain a linear programming model with efficient algorithms known for resolution.

$$x_j \in [0,1] \qquad j = 1, \ldots, m \tag{2.28}$$

However, the entire x_j variables no longer exist to inform which sets are in the solution. Instead, we will have x_j composed of real numbers informing the probability that S_j is in the optimal solution. A draw is then carried out based on the probabilities obtained, and a solution close to the optimal one is reached, where each set j is inserted in the solution with probability x_j. The draw is repeated as long as it does not satisfy the condition of containing all E elements. In the end, we get a collection of sets drawn based on probabilities that minimize the sum of their weights (lighter sets have a greater probability of being in the solution).

(ROST; SCHMID, 2019) implements a heuristic with an approach based on randomized rounding. The authors present a solution for the Virtual Network Embedding Problem (VNEP), which occurs due to failure to meet application requirements for using computational and communication resources between the distributed components (e.g., virtual machines, storage devices) of a cloud. The problem can be described as: given a graph designating computational requirements at the nodes and bandwidth requirements on the edges, an embedding of this Virtual Network in the physical network has to be found, such that both the computational and the network requirements are met (ROST; SCHMID, 2019).

Therefore, for employing randomized rounding for the VNEP, a convex combination of valid mappings $C_r = \{(f_r^k, m_r^k) | m_r^k \in M_r, f_r^k > 0\}$ must be obtained from the LP solution for for each request $r \in R$. Were f_r^k is the probabilities of request of the demand k request r. m_r^k is a valid mapping of request $r \in R$, and M_r is the set of valid mappings of request $r \in R$. The constraints are:

1. The profit of these convex combinations equals the LP's profit, and
2. the (fractional) cumulative allocations do not violate physical network capacities.

Rounding a solution is then done as follows. For each request $r \in R$, the mapping m_r^k is chosen with probability f_r^k, refusing r with probability $1 - \sum_k f_r^k$ (ROST; SCHMID, 2019).

We can summarize the concept of randomized rounding. Given an integer programming for a specific problem, randomized rounding works by:

1. Calculating a solution to its LP relaxation.
2. Decomposing this solution into arched combinations of solutions.
3. Probabilistically selecting solutions based on their weight.

2.4 CLOSING REMARKS

This chapter presents the concepts and fundamentals of the paradigms discussed and used as support for the development of the proposed solution to the elephant flow scheduling problem in DCNs. Besides, it describes the theory of models for forecasting time series that will be used to implement the hybrid forecasting model for elephant flows. The configuration parameters for the models described in this chapter are defined in Chapter 5, in Section 5.2.

3 TRAFFIC ENGINEERING SOLUTION

This chapter defines a traffic engineering solution based on SDN, named RDRH, to schedule elephant flows in DCNs, and the hybrid prediction model, called FARIMA-RNN. Finally, we discuss related works.

3.1 THE APPLICATION FOR TRAFFIC ENGINEERING

The flowchart of our TE solution is illustrated in Figure 7. The prototype comprises an SDN controller and the optimizer RDRH application. The proposed solution uses some features of the Hedera architecture (AL-FARES et al., 2010), which is the reference to compare solutions with a single SDN controller.

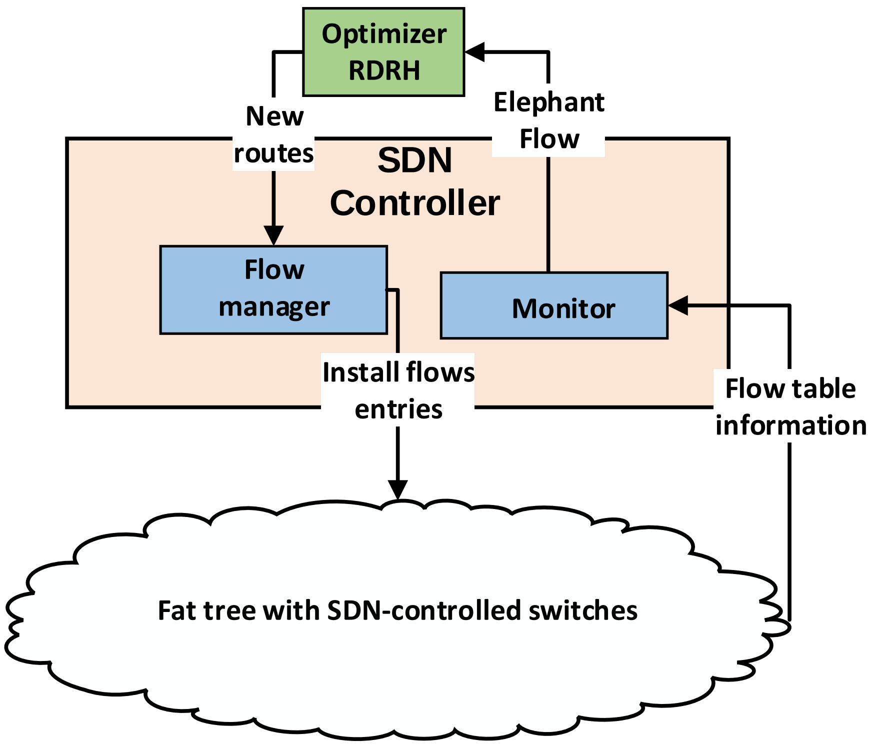


Figure 7 – The RDRH Application for Traffic Engineering

As explained in Section 2.1.1, the SDN paradigm is integrated with DCN to make traffic engineering more flexible and effective. In light of this, we explore this paradigm to our approach. Figure 8 shows the overview of the proposed TE solution's elements. The control plane consists of two components - an SDN controller with two modules, Monitor and Flow manager, and a network control application to schedule elephant flows.

The main SDN features used in our TE's solution are:

- The separation of the control plane and the data plane simplifies and makes network management efficient since a logically centralized controller controls the entire network. The flow scheduling application runs through the controller to operate the SDN-controlled switches in the data plane.

- The interface to the network-control application component. The Monitor module interacts with the network-control application through its "northbound" interface. REST API (HALEPLIDIS et al., 2015) allows the network-control application to read/write network state and flow tables.

- The Flow manager module is responsible for communicating between the SDN controller and controlled network devices. The communication between the controller and the controlled devices crosses what has come to be known as the controller's "southbound" interface. To operate, the controller uses a protocol to communicate with the SDN-controlled switches. This protocol operates based on an exact match of several flow fields such as input port, source/destination address, etc. An SDN switch must also communicate locally-observed events to the controller (e.g., link up or port failure).

We show in Figure 7 the features and details of the relationship among the modules of our prototype. The monitor module is based on Hedera's approach. The SDN controller collects all network topology and flow statistics information through the monitor module, besides, performing scheduling flow tasks.

The monitor module performs elephant flow detection by periodically consulting the edge switches. In the presence of mice flows, the controller sets information to forward with ECMP. Our criteria for identifying an elephant flow is a flow which exceeds a threshold of 10% of the link capacity. The network graph component has information about the network switches/hosts and enables interaction, via the Northbound API, for the application located in the Network Control Application.

The scheduling action directs the elephant flows from the path to which it was allocated to another based on the scheduler's decision. The path decision is based on randomized rounding. The details of the RDRH module are described in the next chapter. The action of installing new flow entries in the flow tables starts the elephant flow's forwarding process. The Flow manager

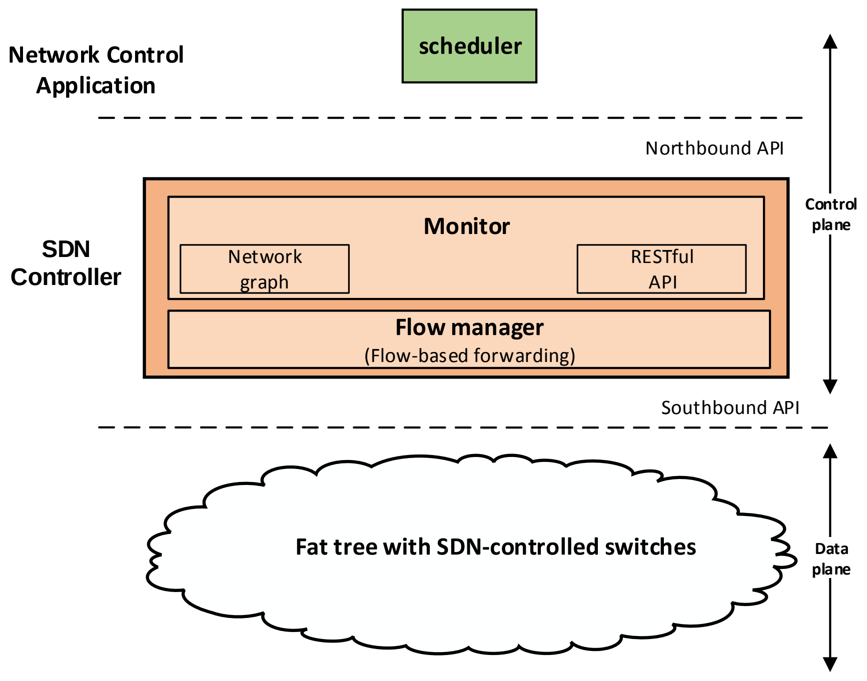


Figure 8 – Overview of the proposed TE solution

module configures the switches' flow tables, which based on the configuration, forward the flows to the destination port.

3.2 PROPOSED PREDICTION MODEL

This section describes FARIMA-RNN, the hybrid prediction model for elephant flows in DCNs presented in this thesis. FARIMA-RNN combines aspects of the FARIMA and RNN models, and provides a good fit for the traffic patterns identified in this thesis.

In this work, the Elman dynamic neural network (ELMAN, 1990) is used to develop the proposed hybrid prediction model. Although this RNN model has three decades, it is used in recent proposals (LI et al., 2019; MEHDIZADEH; FATHIAN; ADAMOWSKI, 2019) for short-term prediction. The Elman RNN becomes an excellent option to improve elephant flow short-term prediction accuracy. The superb performance is because the Elman Neural Network has inherent temporal capabilities that use feature information of the network's past state to perform future decisions, which is required for time series prediction (CHANDRA, 2015). Moreover, Elman recurrent neural network has some advantages, such as faster convergence and more accurate mapping capabilities than ANN models (WANG et al., 2016). Hochreiter and Schmidhuber (1997b) introduced a select type of RNN called LSTM intending to model long sequences (e.g., long-term time series), which is out of the scope of this thesis. Compared with LSTM, Elman's RNN has a simple structure and low training complexity (ZHANG; WANG; TANG, 2019).

The Elman architecture applies a recurrent layer (also known as a context layer), making a copy of the hidden layer outputs in the previous time steps, as shown in Figure 9. The circles represent the nodes of each layer. The representation of the recurrent layer in Elman networks is defined as

$$y_i(t) = \phi\left(\sum_{k=1}^{K} v_{ik} y_k(t-1) + \sum_{j=1}^{J} w_{ij} x_j(t-1)\right), \tag{3.1}$$

where $y_k(t)$ and $x_j(t)$ are the output of the recurrent layer and input layer, respectively. v_{ik} and w_{ij} represent their corresponding weights. ϕ is a sigmoid activation function.

Hybrid forecasting models merge the capacity of different time series models and combine their advantages, allowing, for instance, the simultaneous modeling of long- and short-term dependences, and non-linear structures (KATRIS; DASKALAKI, 2015). The development of hybrid forecasting models has become popular due to better performance than individual models (PANIGRAHI; BEHERA, 2017). The models proposed in (KATRIS; DASKALAKI, 2015; ZHANG; WANG; TANG, 2019) perform well when a time series is composed of a linear and a non-linear pattern. However, a real-world time series may be purely linear or purely non-linear or often contain a combination of linear and non-linear patterns. This demands a new hybrid methodology that can handle a time series with long-range dependence and a different combination of linear and/or non-linear patterns.

Figure 9 then presents a scheme for the representation of FARIMA and Elman's RNN that compose the FARIMA-RNN model. The FARIMA model can be described as the output of the fractional-order system directed by white Gaussian noise for $d \in (-0.5, 0.5)$. The use of the Skewed Student's t-distribution (SSTD) provides more general classes of noise (e.g., Gaussian) (ZHAN; XU; XU, 2015). This distribution allows innovations, i.e., the error terms are assumed to follow Student's t-distribution with zero mean and constant variance σ^2 (KATRIS; DASKALAKI, 2015). An Elman network has recurrent nodes, which are the same number as the hidden nodes. Each recurrent node receives input from a single hidden node and transfers its output to each node in the layer of its corresponding hidden node. Considering that the recurrent nodes depend only on the activations of the hidden nodes from the previous input, the recurrent nodes preserve state information among inputs. The learning algorithm used is

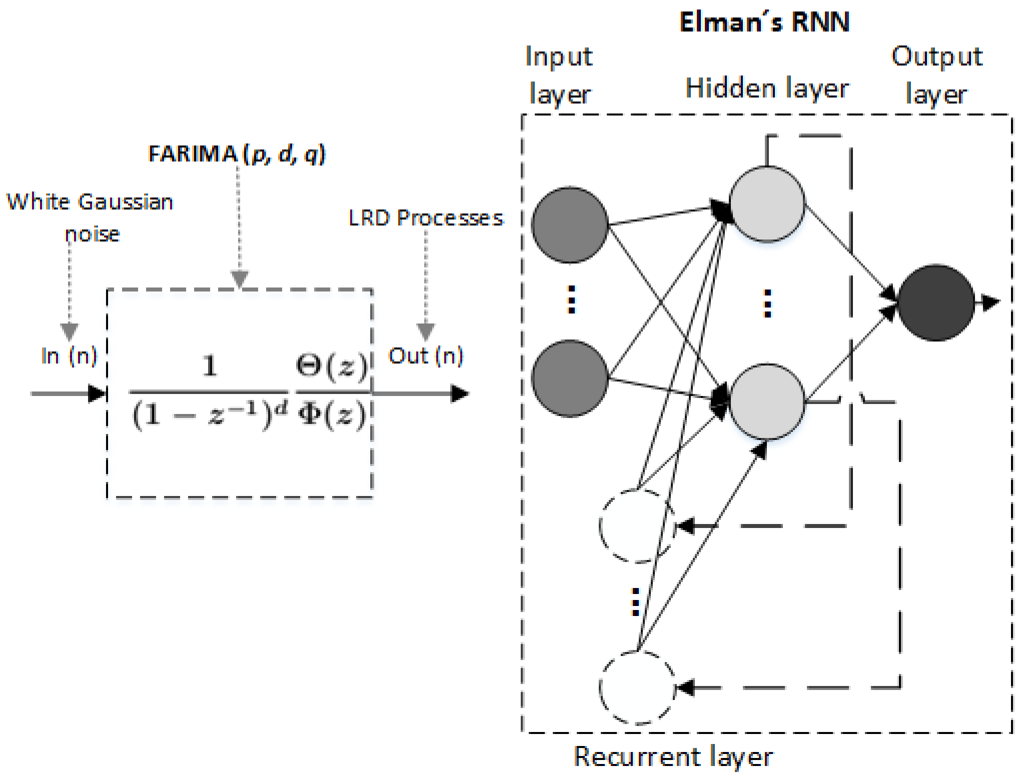


Figure 9 – Proposed FARIMA-RNN model. The representation of the FARIMA model can be defined as the output of the fractional-order process.

Resilient Backpropagation (RProp). This work uses a variant of RProp for the learning function called Jordan and Elman Resilient Backpropagation (JE_RProp) (BERGMEIR; BENÍTEZ, 2012). RProp is a local adaptive scheme, giving fast and robust supervised batch learning in neural networks (MARVUGLIA; MESSINEO, 2012).

The topology of the Elman's RNN is composed of four layers: (a) the input layer, represented by the leftmost circles; (b) the recurrent layer, represented by the dashed circles; (c) the hidden layer, represented by the lightly shaded circles; and (d) the output layer, represented by the darkly shaded circles. The number of nodes of the recurrent layer is always the same as that of the hidden layer. The dashed lines represent the input that recurrent nodes receive from hidden nodes. Only one neuron is used in the output layer. The representation of the models illustrated in Figure 9 has the formal presentation defined in Equation (3.2). Figure 10 shows the sequence of steps to reach the models' final prediction.

Equation (3.2) presents the combination of the models FARIMA-MLP, which was described in Section 2.2.7, and FARIMA-RNN. The prediction $\hat{y}_t$ at each time t is defined by

$$\hat{y}_t = \hat{y}_t^L + h_t^{NL}, \tag{3.2}$$

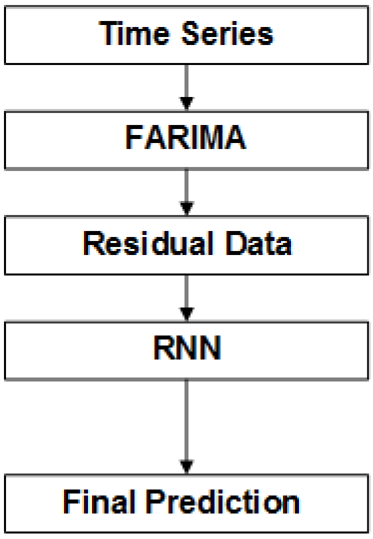


Figure 10 – Flowchart of the FARIMA-RNN model

where $\hat{y}_t^L$ are the predicted values from FARIMA, and h_t^{NL} are the predicted values from the MLP or RNN models. The steps to achieve the model described in Equation (3.2) can be represented by the flowchart depicted in Figure 10. According to the flowchart, the time series represent the selected dataset, which is described in the next section. In the following step, the time series is fitted to the FARIMA model, and residual data values are obtained by subtracting the FARIMA predictions from the original series. RNN models the residual data to get the predictions h_t^{NL}. The Final Prediction rectangle represents the prediction values returned by Equation (3.2).

More details about the proposed model is given in Algorithm 1. In Steps 1–2, initially, a dataset with elephant flows is used in the input. The normalization process between 0 and 1 is the most used for avoiding numerical problems during the training phase. It also makes it easier to compare different datasets that span several orders of magnitude. In Step 3, the function that estimates the best parameters of FARIMA is based on the Maximum Likelihood method. In the training (Step 5), Elman's RNN splits the residual dataset into batches to improve the performance. Lastly, the final prediction is obtained by combining the predictions obtained from the FARIMA model $\hat{y}_t^L$ with the predictions derived from RNN h_t^{NL}.

Algorithm 1: FARIMA-RNN

```
1 Given a dataset represented by elephant flows $y = y_1, y_2, \ldots, y_n$.
2 Normalize dataset in $[0, 1]$.
3 Obtain the best FARIMA $(p, d, q)$ parameters.
4 Obtain the residual of the FARIMA fitted.
5 Train Elman's RNN with FARIMA residual.
6 Obtain predictions using RNN model.
7 Final predictions are obtained by combining the FARIMA $(\hat{y}_t^L)$ with RNN predictions
    $h_t^{NL}$.
```

3.3 RELATED WORKS

This section describes the main works related to this thesis. The analyzed papers are divided into two sections, elephant flow prediction and elephant flow scheduling. This section also points out this thesis's contributions to the state-of-the-art about TE in DCNs.

3.3.1 Elephant Flow Prediction

To date, few research projects include time series analysis and prediction for DCNs. Most papers investigate cloud data centers. Recent works that consider elephant flows when predicting traffic in DCNs are (POUPART et al., 2016; LI et al., 2016). The paper (POUPART et al., 2016) describes the classification and prediction of flows through machine learning techniques. The authors of (POUPART et al., 2016) do not implement a time series analysis to identify autocorrelation in the data. Besides, they do not justify the criterion used to define the ANN's number of layers and nodes. To improve the prediction accuracy in inter-data center links, (LI et al., 2016) proposed a model that combines wavelet transform with ANN to enhance prediction accuracy. The prediction model adds information about sublink traffic and elephant flows explicitly. As elephant traffic data flows are sampled less frequently, (LI et al., 2016) use interpolation methods to construct the missing values to roughly align the data sample of the total traffic and the elephant flow samples. Besides, the authors construct the missing values using interpolation methods to manage the elephant flow information while keeping the monitoring cost low. The elephant flow prediction model proposed in (LI et al., 2016) works in a data center in production; however, the prediction model application considers only inter-data center locality. The hybrid forecasting model proposed in this thesis is applied to elephant flows in different locations, for example, intra-data center and inter-data center.

The main drawback of (LI et al., 2016; KATRIS; DASKALAKI, 2015) is that their results were not compared with related works. (KATRIS; DASKALAKI, 2015) implemented a hybrid prediction model based on FARIMA and MLP ANN. The proposed model in (KATRIS; DASKALAKI, 2015) does not consider RNN when exploring potential time series patterns. This gap favored the implementation of the hybrid model FARIMA-RNN proposed in this work.

(FANG et al., 2012) developed an architecture called Cloud Resource Prediction and Provisioning Scheme (RPPS), which includes the ARIMA model to automatically predict workloads in a cloud data center. RPPS implements proactive resource provisioning for cloud applications. The authors used real and simulated data but did not evaluate the results compared to other models. Statistical analysis using hypothesis tests was not performed. The analysis did not use the metrics Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) to evaluate the error when compared to other models. (DALMAZO; VILELA; CURADO, 2017) presented a state-of-the-art taxonomy for network traffic prediction and classified the models into two categories: Pattern Recognition and Time Series. Their major contribution is the study of the performance of several predictor models in a cloud computing environment. The solutions presented in (DALMAZO; VILELA; CURADO, 2017) did not consider model fitting or a method to evaluate the nonlinear time series. Also, the statistical analysis does not identify patterns in the datasets. Table 2 presents a summary of the main related works listed above.

Outside the context of data centers, (SILVA et al., 2020) presents solutions for identifying and predicting elephant flows at the Internet Exchange Point. The predict elephant flows process uses the regression-based statistical model called Locally Weighted Regression (LWR), derived from standard linear regression. Besides, it performed the classification of elephant flows in real-time in the data plane. A drawback of the solution proposed by (SILVA et al., 2020) did not consider that the behavior of elephant flows that have a non-linear pattern. The implementation of LWR presented a lower performance than models such as Simple Linear Regression and Nearest Neighbor Regression family as shown in (DELL'ACQUA et al., 2015).

(ALDHYANI et al., 2020) propose a hybrid model for network traffic prediction. The strategy uses clustering using fuzzy c-means to analyze the network data to improve time series. The method uses clustering with fuzzy c-means to analyze network data to improve time series. This makes it possible to deal with ambiguities from all network data to improve the time series models. The resulting clusters are used as input to the weighted exponential smoothing method. The clusters are combined and used as input for LSTM and Adaptive Neuro-Fuzzy

Inference System time series models. Two real network traffic traces were used to test and evaluate the proposed hybrid model. The authors compared the prediction results between the conventional models with the hybrid prediction proposed and state-of-the-art models. It is observed that the prediction results of the proposed hybrid model are more satisfactory. The authors did not analyze the execution times of the proposed model. However, the method has a pre-processing step with clustering and the weighted exponential smoothing method before applying the final forecast making the model prohibitive due to the possible high execution times.

Work	Prediction Model	Drawback	Major Contribution
(POUPART et al., 2016)	ANN Gaussian Process Regression Online Bayesian Moment Matching	No time series analysis was performed to identify autocorrelation in the data.	Implemented classification and prediction of elephant flows through machine learning techniques.
(LI et al., 2016)	Wavelet transform with ANN	Prediction model application considers only inter-data center locality.	Information about sublink traffic and elephant flows explicitly.
(KATRIS; DASKALAKI, 2015)	FARIMA combined with MLP ANN	Does not consider RNN when exploring potential time series patterns.	Combining of prediction models for network traffic.
(FANG et al., 2012)	ARIMA	The authors used real and simulated data but did not evaluate the results compared to other models. Statistical analysis using hypothesis tests were not performed.	Implements proactive resource provisioning for cloud applications.
(DALMAZO; VILELA; CURADO, 2017)	Weighted Moving Average, Exponential Moving Average, Poisson Moving Average, ARIMA, ARMA	Does not consider model fitting or a method to evaluate the nonlinear time series.	Study of the performance of several predictors, models in a cloud computing environment
(SILVA et al., 2020)	Locally Weighted Regression	The implementation of LWR presenteda lower performance than models such as Simple Linear Regression	Performs the classification and prediction of elephant flows in real-time in the data plane.
(ALDHYANI et al., 2020)	LSTM, Adaptive Neuro-Fuzzy Inference System	The possible high execution time.	Combination of clustering with LSTM and Adaptive Neuro-Fuzzy Inference System time series models.

Table 2 – Qualitative overview of related works with DCN traffic prediciton.

3.3.2 Elephant Flow Scheduling

ECMP (THALER; HOPPS, 2000) is the default protocol for multipath routing and load balancing in DCNs. For this reason, we chose it to compare with the heuristic proposed in this thesis. In ECMP, a switch locally selects the next hop from multiple equal-cost paths by calculating a hash value based on IP five-tuple: the source and destination IP addresses, transport port numbers, and protocol (for instance, by applying the function $x \mod n$, where x is the least significant octet of the IP address and n the number of interfaces (HOPPS et al., 2000)). Therefore, this result is used to direct the packet to the switch's output interface. ECMP has two major shortcomings. First, ECMP cannot effectively utilize the available bandwidth due to hash collisions (AL-FARES et al., 2010). Second, as mentioned in the previous chapter, ECMP can lead to load imbalance because it works independently of link load and flows size. There is no differentiation between mice flows and elephant flows. Thus, mice flows, which usually are latency-sensitive flows, frequently find themselves queued behind the elephant in the egress ports, suffering from long queueing delays (ZHANG; CUI; ZHANG, 2017).

The most cited heuristic for traffic engineering in DCNs with an available open-source version is Hedera (AL-FARES et al., 2010), which implements a centralized dynamic flow scheduling for the fat-tree topology. System monitoring occurs by sending information from links and flows to the controller. When an elephant flow is detected, the edge switch sends a message to the scheduler about the event. The scheduler re-routes elephant flow according to the result of the estimated demand for large flows and allocates it based on a greedy solution called Global First Fit. The proposal's goal is to maximize the network utilization, generating less delay or impact on the active flows in the schedule. With a global view of the network, it is possible to visualize routing and traffic demands. This view makes it possible for the scheduling system to visualize bottlenecks, not visualized by local switches.

Zhang et al. (ZHANG; CUI; ZHANG, 2017) propose an elephant flow scheduling algorithm called Fincher based on the stable matching theory. This algorithm obtains an optimal stable matching between flows and switches by matching all flows to their appropriate paths. The authors prove that the Flow-Switch Stable Matching (FSSM) problem is NP-hard.

Previous works investigated the impact of elephant flow scheduling on DCNs with the fat-tree topology. (AL-FARES et al., 2010; ZHANG; CUI; ZHANG, 2017) gave us the other flow scheduling and load balancing features and taught us to improve RDRH. Hedera was the first solution using heuristics to load balance data centers with SDN. Open-source versions of Hedera (AL-FARES et al., 2010) [1] made it possible to reproduce and develop TE research in DCNs. Fincher (ZHANG; CUI; ZHANG, 2017) presented a solution based on graph theory and demonstrated that the routing of flows in a fat-tree topology has NP-hard algorithmic complexity.

(LI; LU; FU, 2020) proposed a new dynamic elephant flow scheduling algorithm combining genetic and ant-colony (GA-ACO) algorithms. In SDN architecture, the proposed GA-ACO algorithm is used to get a global view of the network. It determines and re-routes elephant flows on the congestion links to the globally optimal path. The genetic algorithm is used in the global search for available paths, and these are passed as input to the ant colony algorithm.

[1] <https://reproducingnetworkresearch.wordpress.com/?s=hedera&submit=Search>

The authors conducted extensive experiments to verify the GA-ACO algorithm's performance compared with the ECMP and ACO-SDN. The simulation results show that GA-ACO is superior to ECMP and ACO-SDN. GA-ACO does not only minimize the full usage of links but also significantly increase the bisection bandwidth. This approach has some constraints due to the overhead in the elephant flow detection process.

In another research, (WANG et al., 2016) proposed a DCN architecture scheme named Freeway that gains from the existence of multiple shortest paths in DCN. This scheme avoids the trade-off by separating elephant and mice flows and forwarding them over dynamically partitioned low latency and high throughput paths. An M/G/1-based model is implemented to theoretically obtain the highest value of average delay over the path that will guarantee, for 99% of mice flows, their completion time before the deadline. Based on this bound, Freeway proposes a dynamic path partitioning algorithm to dynamically adjust the number of low latency and high throughput paths with varying traffic load.

Compared to ECMP, a solution based on path computation through hashing, our proposal distributes traffic flows more efficiently among alternative paths. Our RDRH solution is based on a centralized flow scheduling implemented on Hedera. By comparison, RDRH performs better, especially considering congestion and loss metrics. The authors of Fincher's algorithm performed a throughput comparison with Hedera and ECMP without a statistical validation. We implement a complete performance evaluation with a statistical validation. Fincher's proposal implements an optimization model considering the association between flows and switches. Our model has another approach that considers the optimization of network links. None of the existing research used Randomized Rounding to solve the elephant flows scheduling problem to the best of our knowledge. Table 3 presents a qualitative analysis of the state of the art and the proposal of this thesis.

Work	Heuristic	Drawback	Major Contribution
ECMP	Shortest path	Hashing collisions can lead to traffic load imbalance	Optimize multipath switched networks
Hedera	Global First Fit and Simulatted Anneling	Large delay in the switches	A centralized dynamic flow scheduling to the fat-tree topology
Presto	-	Split elephant flows into small units causing packet reordering, delay and processing	Design and implementation of a soft-edge load balancing scheme
Fincher	Stable Matching theory	The stable matching solution obtained is not necessarily optimal	Stable matching theory to perform elephant flow schedule in DCNs
GA-ACO	Genetic and ant-colony algorithms	GA can provide premature convergence to local minima resulting in low accuracy.	GA-ACO algorithm
Freeway	Dynamic Path Partition algorithm and an M/G/1-based model	Requires knowing the size of elephant flows before they start, so it is difficult to deploy it in practice	Partitions the forwarding paths into high throughput paths and low latency paths for elephant flows and mice flows, respectively
RDRH	Randomizing Rounding	The Randomized Rounding solution is not necessarily optimal	Randomized Rounding to perform elephant flow scheduling in DCNs

Table 3 – Qualitative overview of DCN load balancing solutions.

3.4 CLOSING REMARKS

This chapter presented a TE solution for elephant flow scheduling in DCNs and a hybrid model to predict elephant flows in the short-term. These proposals addresses interdisciplinary areas in computer science and statistics: graph optimization, linear programming, time series analysis, forecasting, and computer networks. Finally, the main related works were presented.

4 RANDOMIZED ROUNDING HEURISTIC SOLUTION

This chapter defines the modeling stage of the proposed randomized rounding heuristic based on a multicommodity network flow, and the problem of routing elephant flows in fat-trees. Section 4.1 describes the RDRH module, which is integrated with the traffic engineering solution defined in the previous chapter.

4.1 RANDOMIZED ROUNDING HEURISTIC MODULE

The Fat-Tree Routing problem for elephant flow scheduling introduced in Subection 2.3.2 is NP-Hard (WANG et al., 2016). Our goal is to solve it in polynomial time. We model FTR problem as a multicommodity flow, but without the Constraint 2.21 (Link Capacity). As the objective function seeks to balance the links' residues by maximizing the smallest residue, we expect that in an optimal solution there will be no link with usage that greatly exceeds its capacity.

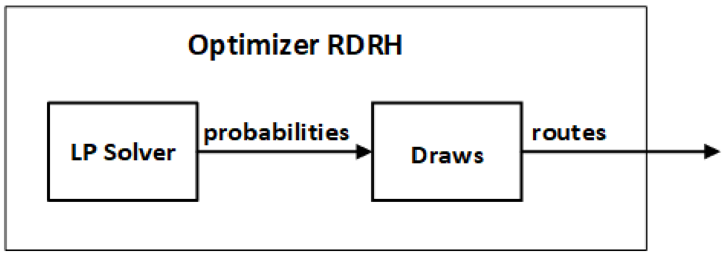


Figure 11 – Randomized Rounding Heuristic (RDRH) module

RDRH consists of two main functions: obtaining the probabilities of choosing the links through the solution of a linear programming formulation, and drawing the links to use based on the obtained probabilities. Figure 11 shows RDRH functions (represented by boxes) and the output of each RDRH's component.

We can formulate the FTR problem using the integer programming model below, based on the multicommodity network flow formulation described in the previous section.

$$\min y \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell} f_i(u,v)d_i - y \leq c(u,v) \quad \forall (u,v) \in E \tag{4.2}$$

$$\sum_{w \in V} f_i(u,w) - \sum_{w \in V} f_i(w,u) = 0 \quad \forall u \in V - \{s_i, t_i\}, \forall i \in \{1, \ldots, \ell\} \tag{4.3}$$

$$\sum_{w \in V} f_i(s_i,w) - \sum_{w \in V} f_i(w,s_i) \quad = 1 \quad \forall i \in \{1, \ldots, \ell\} \tag{4.4}$$

$$\sum_{w \in V} f_i(w,t_i) - \sum_{w \in V} f_i(t_i,w) \quad = 1 \quad \forall i \in \{1, \ldots, \ell\} \tag{4.5}$$

$$f_i(u,v) \quad \in \{0,1\} \quad \forall (u,v) \in E, \forall i \in \{1, \ldots, \ell\} \tag{4.6}$$

The objective function (4.1) minimizes the overall elephant flow traffic in the links. Equation (4.2) is similar to Equation (2.21) described previously in the multicommodity flow model; the difference lies in the fact that it allows the existence of an excess y in the links' capacity. However, the objective function (4.1) tends to minimize y, therefore overloads are allowed but avoided and reduced. The function (4.1) estimates the greatest excess bandwidth among links. The equations (4.3), (4.4), and (4.5) ensure that equations (2.22), (2.23), and (2.24) of the multicommodity flow model are satisfied.

Although solving a general linear formulation with a binary variable is an NP-hard problem, if we relax the integrality of these binary variables, we can solve the formulation in polynomial time (HOLMBERG, 2007). Thus, the original formulation's binary variables receive real values between 0 and 1 in the relaxed solution, but this fractional solution can be obtained in polynomial time. We can relax the FTR formulation by change the Equation (4.6) to $f_i(u,v) \in [0,1], \forall (u,v) \in E, \forall i \in \{1, \ldots, \ell\}$. We call this relaxed formulation the Modified Multicommodity Flow Relaxation (MMFR) model.

The objective function (4.1) allows better distribution of flows among links (load balancing), which makes it possible to improve throughput by allocating elephant flows on alternative paths. Mice flows are forwarded based on hashing and shortest path through ECMP, allowing the decrease of round-trip delay and loss rate metrics since the traffic is well balanced.

Algorithm 2 depicts the randomized rounding heuristic in pseudo-code. The algorithm's input parameters with the number of demands, origin, and destination of the demands. Based on these entries, the solution of LP relaxation MMFR is performed in Step 1. In steps 5 to 8, each elephant flow paths are drawn based on the optimal solution of the MMFR relaxation generated by the LP. In Steps 5 to 8, each elephant flow paths are drawn based on the optimal solution of the MMFR relaxation generated by the LP. The drawing in Step 6 is a random sample from an empirical distribution.

Algorithm 2: RDRH

```
Input:
$n$: number of demands.
$s[i]$: origin of the $i$th demand.
$t[i]$: destination of the $i$th demand.
Output: the path of each demand.
1 Let $p$ be the solution of LP relaxation MMFR, i.e., $p[i][u][v]$ is the value of $f_i(u, v)$ in
    the optimal solution.
2 $P[i] \leftarrow \{ \}$ // The path of the $i$th demand
3 foreach demand $i$ do
4     $u \leftarrow s[i]$
5     while $u$ is not $t[i]$ do
6         Draw a node $v$, where each $v$ has chance $p[i][u][v]$.
7         P[i] $\leftarrow P[i] \cup \{(u, v)\}$
8         $u \leftarrow v$
9 return $P$
```

4.2 EVALUATION OF TRAFFIC ENGINEERING SOLUTION

The controlled experiment used in this section was designed based on the procedures indicated in (PFLEEGER, 1995). The design of an experiment consists of the following steps:

- Conception;
- Design;
- Preparation;
- Execution.

4.2.1 Conception and design

The assessment aims to ascertain the RDRH solution's capacity to improve the allocation of elephant flows in DCNs. For this reason, experiments were carried out to measure the flow capacity of the links with elephant flows, and we also evaluated the links with mice flows.

We evaluate our proposed heuristic performance from different performance evaluation metrics such as:

- Throughput: characterizes how much data the switch can transfer per second between connected hosts. This metric is the most relevant to elephant flows in DCNs. The forwarding of the elephant flows requires a high and sustained throughput;

- Round-trip-delay: the time takes for a small packet to travel from source to destination and then back to the source.
- The packet loss rate will occur when the arriving packet or one of the already queued packets will be dropped (KUROSE; ROSS, 2012). Round trip delay and loss are essential to detect congestion symptoms.

We evaluated RDRH with part of the set of communication patterns in: (AL-FARES; LOUKISSAS; VAHDAT, 2008; AL-FARES et al., 2010; ZHANG; CUI; ZHANG, 2017). We used a synthetic workload, generated by iPerf and composed of two communication patterns following the characteristics:

- Staggered Prob $(EdgeP, PodP)$: a host sends to another host in the same edge switch with probability EdgeP; its same pod is sent with probability PodP, and the remainder of the network with probability $1 - EdgeP - PodP$.
- Random: a host sends to any other host in the network with uniform probability. Moreover, there are bijective mappings as well as ones where hotspots are present.

4.2.2 Preparation and execution

In the preparation phase of the experiments, we defined the tools used to implementation and evaluate the prototype.

- Mininet 2.3.0: Mininet is an open-source, easy-to-deploy, and lightweight network emulator that provides a programmable interface to define and build network configurations with virtualized elements. All experiments were conducted using the Mininet (LANTZ et al., 2015) emulator. It integrates several SDN components allowing the test and design of network architectures and services. Emulators enable the reproduction of experiments that best reproduce the hardware and overcome network simulations in terms of reliability, flexibility, and cost (MUELAS; RAMOS; VERGARA, 2018);
- Open vSwitch 2.5.4: software used by Mininet to create virtual switches for an SDN network;

- Ryu framework 4.26: Ryu is an SDN controller framework implemented in Python; it is widely used in academia and industry (ALIYU et al., 2020). This controller was selected due to network management's capacity and control applications development through the APIs available;
- iPerf 2.0.5: software capable of producing network traffic. The experiments were carried out using iPerf, which can calculate the available bandwidth between any two hosts by creating a client-server Transmission Control Protocol (TCP) connection (LEBIEDNIK; MANGAL; TIWARI, 2016);
- bwm-ng [1]: software used to monitor the flows' bandwidth;
- C++: general-purpose programming language created as an extension of the C programming language. The solution prototype was implemented in this language due to its performance, efficiency, and flexibility. The TE solution was integrated with RDRH by extending a project available on Github[2];
- IBM CPLEX (IBM, 2018): High-performance mathematical programming solver for linear programming, mixed-integer programming and quadratic programming. We use to resolve the LP MMFR.

Table 4 presents a summary of the tools and parameters of the experiments.

Specification	**Detail**
SDN emulator	Mininet 2.3.0
SDN controller	Ryu 4.26 - Software defined networking framework
Switch type	Open vSwitch(OVS) - 2.5.4 and OpenFlow 1.3 enabled
Traffic generator	iPerf 2.0.5
Traffic type	TCP
Network Topology	Three layered fat-tree with 4-pod and 8-pod

Table 4 – Details of the specification of the experiment.

We compare the impact on the performance of RDRH with Hedera and ECMP on a fat-tree topology. This impact was measured by allocating elephant and mice flows in topologies with 4-pod and 8-pod. Topologies with 4-pod have 16 servers, 20 switches, and 32 links. For 8-pod, there are 128 servers, 80 switches, and 256 links. In a total of 8 experiments, we evaluated

[1] <https://linux.die.net/man/1/bwm-ng>
[2] <https://github.com/Huangmachi/exp_EFattree>

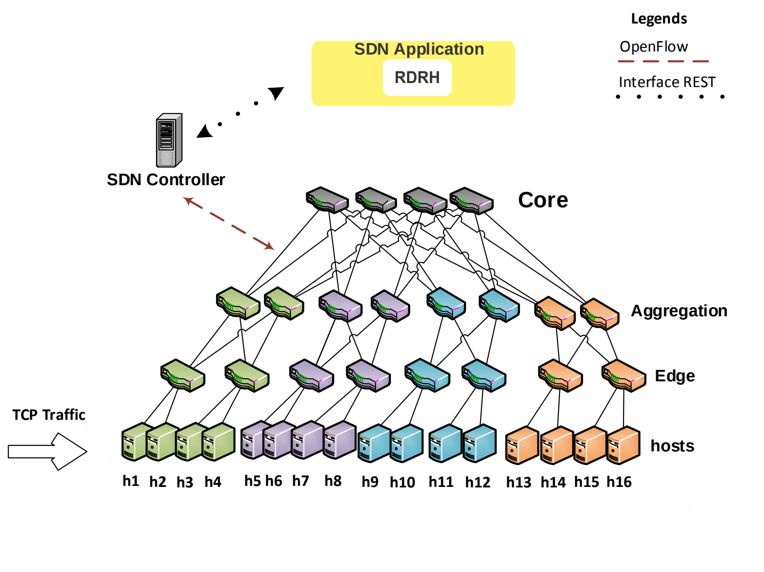


Figure 12 – 4-pod topology used in the experiments to assess the impact of elephant flow on DCNs.

the throughput, round-trip delay, and loss on two levels of use of a data link, varying the bandwidth by 100 Mbps and 1000 Mbps. TCP traffic was generated by iPerf complying with the communication patterns described previously, varying the amount of TCP flows between the source and destination hosts in one and five flows. These experiments were performed on a dedicated server running a 64-bit Ubuntu operating system with eight vCPUs and 24 GB of memory.

Figure 12 illustrates one of the scenarios used in experiments with 4-pod topology with the following steps:

- Create the topology based on the value of k (number of switch ports);
- Create hosts peers $(h_1 \ldots h_n)$ so that traffic is generated with *iPerf*;
- Initialize *bwm-ng* to monitor throughput;
- Mice traffic monitoring is performed by measuring the metrics of round-trip delay and loss effected through *ping* application;
- TCP traffic is generated by *iPerf* according to the bandwidth configuration tested in the topology.

The controller has a monitoring module described in Chapter 3 that runs a pooling in the switches to identify elephant flows that exceed 10% of the link capacity.

All experiments, compare RDRH to ECMP and Hedera to show performance and scalability, considering scenarios used in DCNs. ECMP is the most common load-balancing scheme used on DCNs, and the most cited heuristic with an open-source version is Hedera. Each experiment lasts for 60 seconds for each origin and destination and uses TCP flows with values of 1 and 5 for staggered and randomized communication patterns. We use TCP traffic generated by iPerf because it is the most predominant transport protocol used in DCNs.

We collect the running times of the three algorithms: RDRH, ECMP, and Hedera. These do not include the fat-tree build and link discovery times in the emulated testbed, because the goal is the algorithms' processing time. The discussion about computing time of the proposals evaluated in this thesis is in the next section.

4.3 PERFORMANCE EVALUATION OF THE RDRH FOR TE

In this section, the results of the experiments described in the previous section are analyzed.

4.3.1 Evaluation Metrics

The performance evaluation of the RDRH for TE was carried out using different metrics, such as throughput, round trip delay, and loss rate. The throughput's statistical evaluation is because this metric is more relevant for elephant flows in DCNs (ABDELMONIEM; BENSAOU, 2015). Routing elephant flows requires a high and sustained throughput. Round-trip delay and loss rate are essential to detect congestion indications. These metrics are applied to assess non-elephant traffic situations. All experiments compare the RDRH with ECMP and Hedera to show performance and scalability of the solutions, considering scenarios with DCN topologies. ECMP is the most common load balancing scheme used in DCNs, and the most frequently cited heuristic with an open-source version is Hedera.

4.3.2 Data Analysis

The results of the experiments for 4-pod and 8-pod, for 100 Mbps and 1000 Mbps, are illustrated in Figures 13–16. In the left column of the figures are the experiments' results for one TCP connection between source and destination, in the right column for five TCP connections. Figure 13a shows that RDRH outperforms ECMP in the normalized transfer rate for most traffic patterns, and outperforms Hedera in three patterns. When the number of TCP flows increases to five, as shown in Figure 13b, RDRH shows better results in five traffic patterns. As shown in most of figures (a) and (b), the throughput gain increases with the degree of communication locality in the staggered patterns.

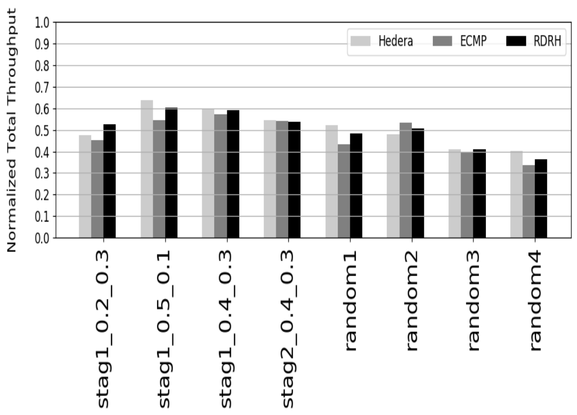
Figures 13c and 13d show round-trip delay. RDRH has a shorter delay on most of the Random traffic for 100 Mbps. In Figure 13c, ECMP showed a shorter delay in most of the stag pattern, but the delay is more significant in most random patterns. Figure 13e shows that the loss level for RDRH is higher than Hedera and ECMP in only two traffic patterns. In Figure 13f, the loss rate of the RDRH is lower than other algorithms in most of the patterns. The most visible reduction in packet loss is observed in all scenarios with a bandwidth of 1000 Mbps, except in Figure 16e, in which RDRH shows a loss higher than other solutions.

For 8-pod, the results in figures 15 and 16 show that RDRH varies in throughput. RDRH shows poorer performance than the other algorithms in the stag pattern of one flow experiment. RDRH outperforms throughput in three traffic patterns when running with five flows, as shown in Figure 16b. RDRH has a shorter delay in most traffic except stag1_0.5_0.1, and stag2_0.4_0.3 patterns for 1 and 5 TCP flows with 100 Mbps. Figure 13f shows that RDRH's round-trip delay is smaller than others in most patterns with an increase in the number of flows.

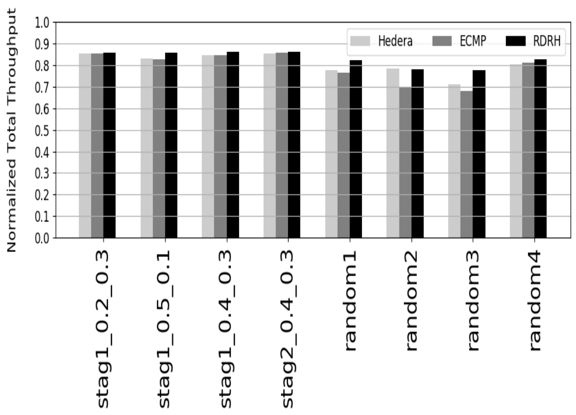
The throughput values illustrated in figures 15 and 16 show the transfer rate with close values in the random pattern. The RDRH algorithm has a round-trip delay and loss rate that outperforms Hedera and ECMP in most traffic patterns. Besides, the allocation of elephant flows by the RDRH improved the performance of mice flows. Round-trip delay and loss rate were measured with mice traffic.

The results of the RDRH were better than those of the other solutions in most of the metrics evaluated for 4-pod. Highlights for round-trip delay and packet loss. The throughput is less than or equal theirs in some scenarios for 8-pod, though round-trip delay and packet loss show improvements in performance in most cases. For 4-pod and 8-pod, the throughput values were very close among the evaluated solutions in some scenarios.

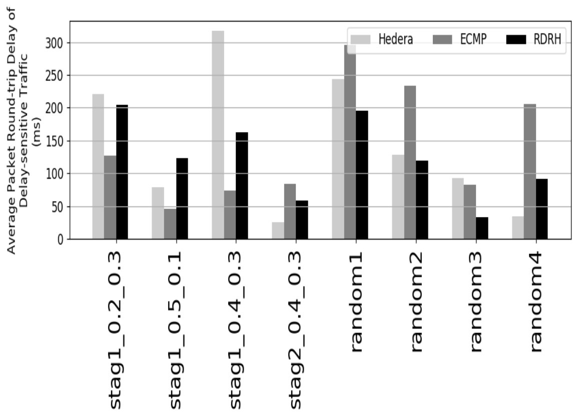
Although the RDRH throughput performance was not superior to that of the other solutions for the 8-pod scenario, we can conclude that this, in fact, does not restrain our solution. RDRH showed similar performance in several scenarios. Also, the results' insights are that it does



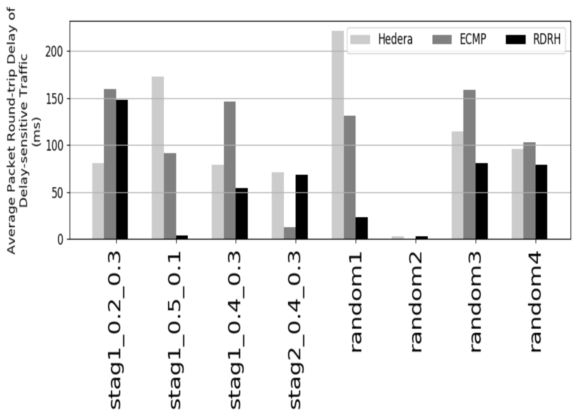
(a)



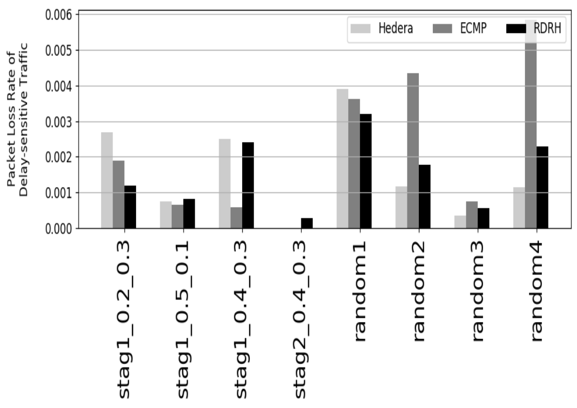
(b)



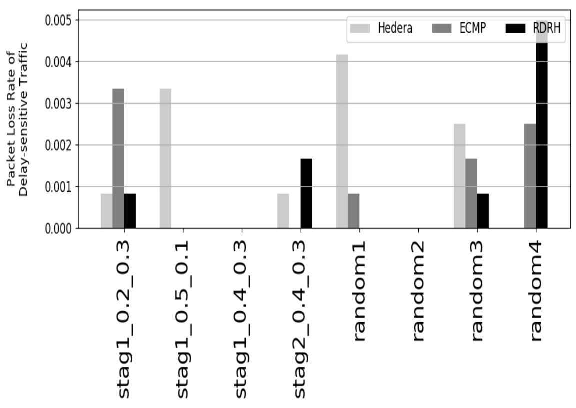
(c)



(d)

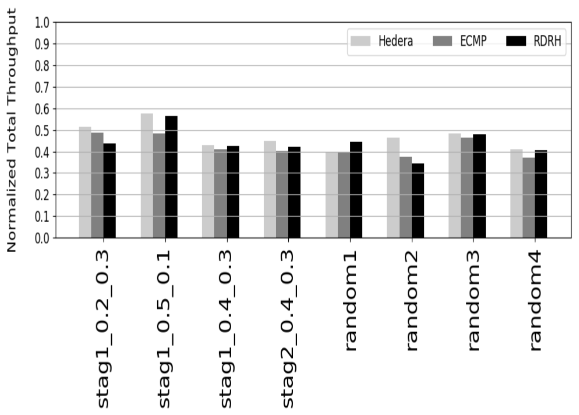


(e)

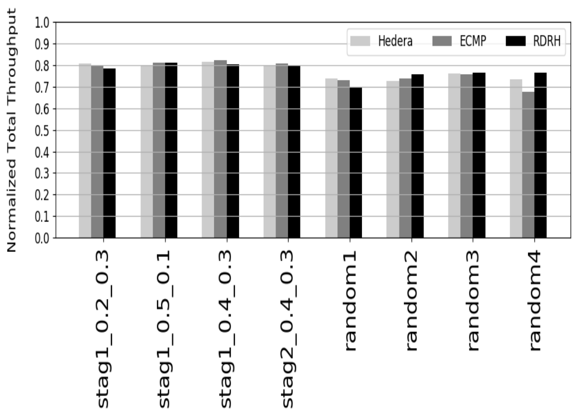


(f)

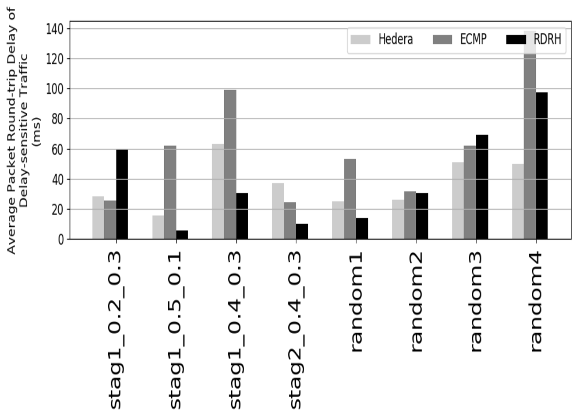
Figure 13 – Comparison of scheduling algorithms on a fat-tree topology of 4-pod with 100 Mbps. Experiments with one TCP flow are in the left column plot, and five flows in the right column plot.



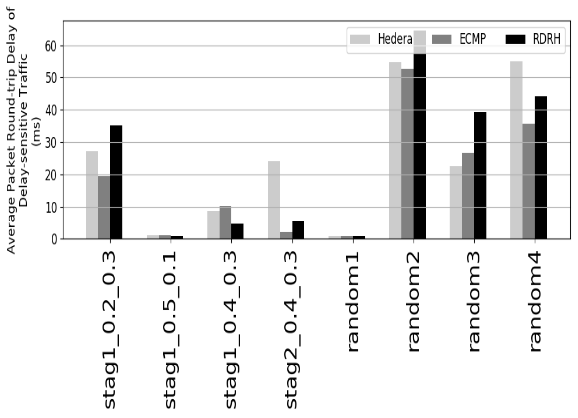
(a)



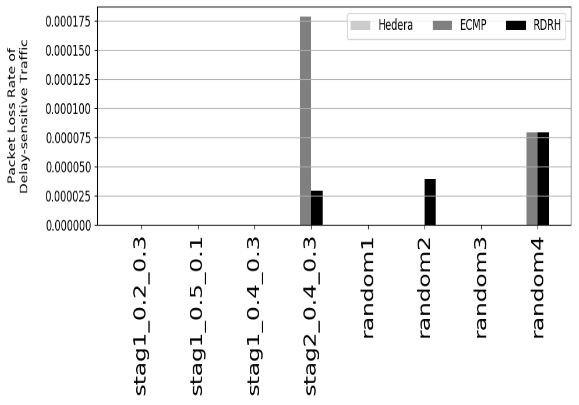
(b)



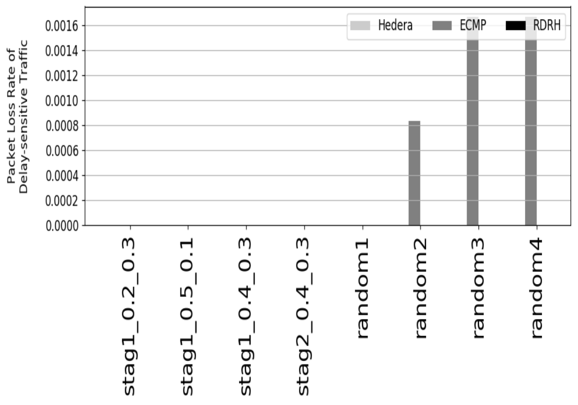
(c)



(d)

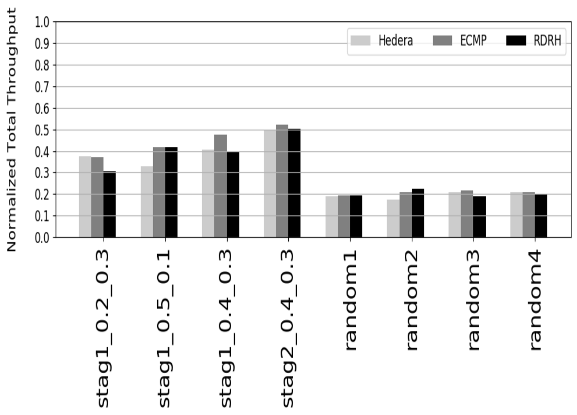


(e)

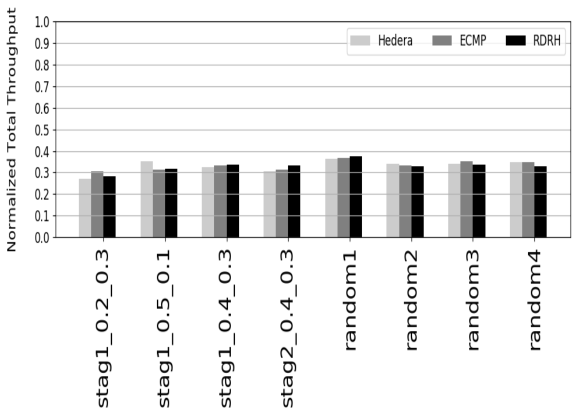


(f)

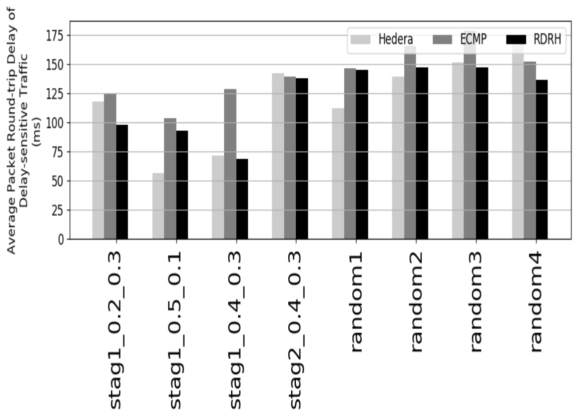
Figure 14 – Comparison of scheduling algorithms on a fat-tree topology of 4-pod with 1000 Mbps. Experiments with one TCP flow are in the left column plot, and five flows in the right column plot.



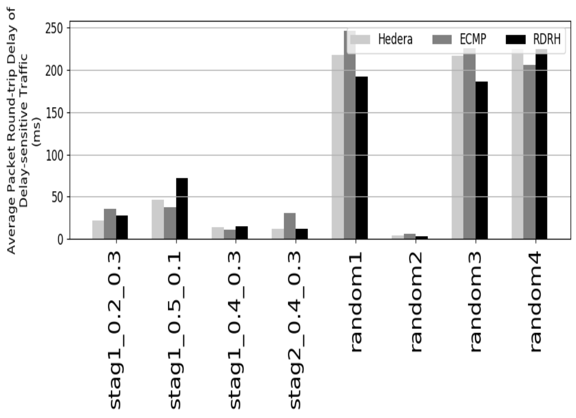
(a)



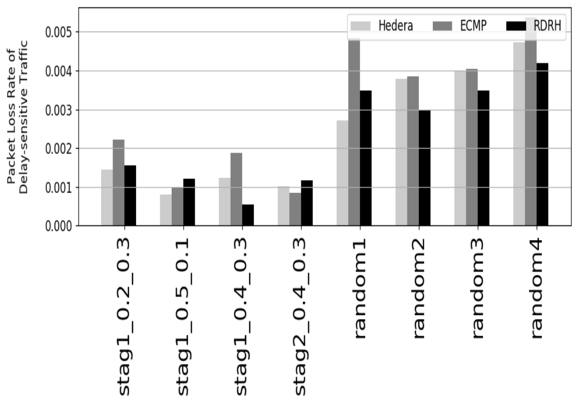
(b)



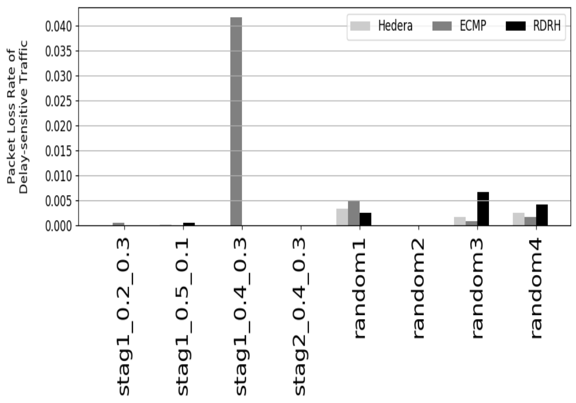
(c)



(d)

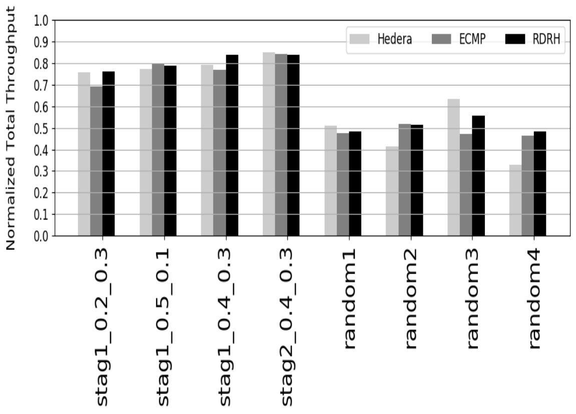


(e)

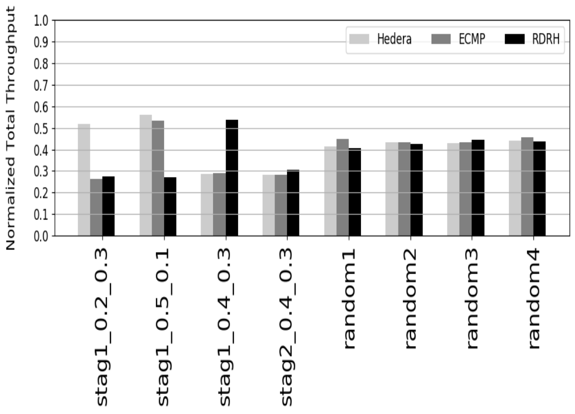


(f)

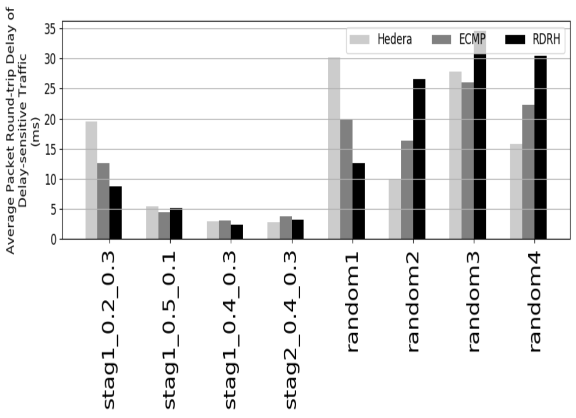
Figure 15 – Comparison of scheduling algorithms on a fat-tree topology of 8-pod with 100 Mbps. Experiments with one TCP flow are in the left column plot, and five flows in the right column plot.



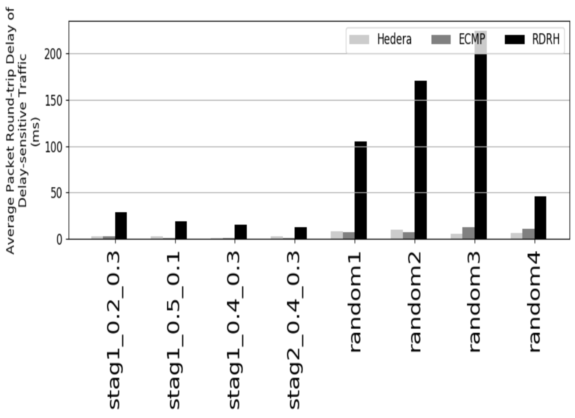
(a)



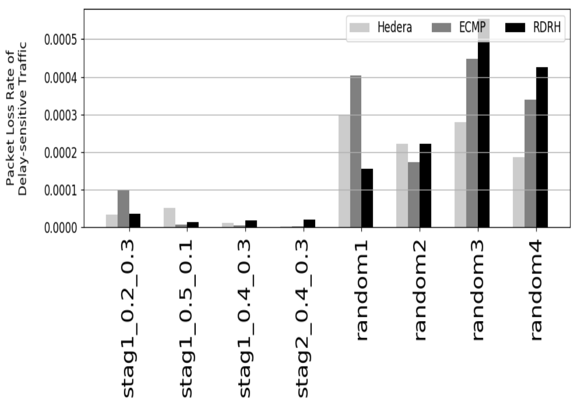
(b)



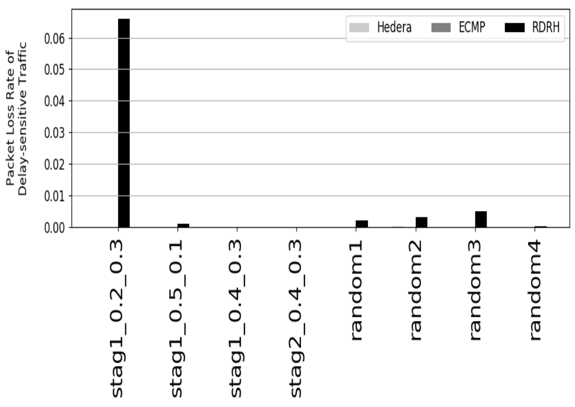
(c)



(d)



(e)



(f)

Figure 16 – Comparison of scheduling algorithms on a fat-tree topology of 8-pod with 1000 Mbps. Experiments with one TCP flow are in the left column plot, and five flows in the right column plot.

improve round-trip delay and loss rate for delay-sensitive traffic. These metrics are crucial for real-time applications. In the next section, we will check and quantify the uncertainty in these results through hypothesis testing. The metric throughput is evaluated because it is the most critical metric with respect to elephant flows in DCNs.

4.3.3 Statistical Validation

It is possible to verify and certify the performance metric's confidence values illustrated in the previous section. In hypothesis testing, the p-value is the probability of obtaining a result, which indicates the lowest level of significance for rejection of the null hypothesis. It is the most critical information for decision making in statistical analysis. A p-value provides information on whether a statistical hypothesis test is significant or not (GARCíA et al., 2010).

Hypothesis testing reveals which solution presents the best performance in elephant flow allocation. All the hypothesis tests conducted are provided by specifying a threshold significance level α of 5%. The use of the Shapiro-Wilk (SW) adherence test indicates what type of hypothesis testing to choose: parametric or non-parametric (TRIVEDI, 2011). P-values for all data sets resulted in amounts of less than the significance level of 5%. All SW tests suggest the use of non-parametric hypothesis testing because p-values were smaller than 5%. Therefore, we use the Friedman, Nemenyi Post-Hoc, and Wilcoxon Signed-Rank tests (WSRT) (GARCíA et al., 2010) to validate the results of our experiments. These tests are valuable for identifying the algorithm that presents the best performance.

Friedman's test aims to determine if we may conclude from a sample of results that there is a difference among the RDRH, Hedera, and ECMP. The formulated hypotheses conducting the Friedman's test are:

$$H_0 : \text{There is no statistical difference among the algorithms.} \tag{4.7}$$

$$H_a : \text{There is a statistical difference among the algorithms.} \tag{4.8}$$

For most fat-tree scenarios with 4-pod with bandwidth 100 Mbps, and 1000 Mbps, we obtain a p-value smaller than 5% with Friedman's test. Therefore, the hypothesis H_0 (4.7) must be rejected because there is a statistical difference between the algorithms' performance. For the 8-pod scenario with one TCP connection, Friedman's test showed a p-value of less than 0.05 only for stag1_0.2_0.3 for 100 Mbps, and random3 and random4 patterns for 1000 Mbps.

For the 8-pod scenario with five TCP connections, only the stag1_0.2_0.3, stag1_0.5_0.1, and stag2_0.4_0.3 patterns.

The Nemenyi post-hoc test indicates if there is a performance difference between algorithms pairs. For the Nemenyi test, the hypotheses are:

H_0 : The performance of algorithms **A** and **B** does not differ significantly.

H_a : The performance of algorithms **A** and **B** differs significantly.

For paired tests, Algorithm A is RDRH and B is Hedera or ECMP.

Tables 5–10 show results for this paired test. It is worth noting that this test was applied only to the solutions that showed a statistical difference by the Friedman test. The algorithms' performance differs significantly in most cases because of the alternative hypothesis H_a in fields with a p-value of less than 5%. Tables 5–6 show the p-values of the RDRH comparison with the other solutions. There is a difference for tests with one TCP connection in 19 combinations for 100 Mbps and 1000 Mbps. Tables 7–8 present results for testing with five TCP connections. The results are similar to the test for one TCP connection.

	100 Mbps							
	Stag1_0.2_0.3		**Stag1_0.5_0.1**		**Stag1_0.4_0.3**		**Stag2_0.4_0.3**	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
ECMP	0.093	-	0.32	-	0.0487	-	(<0.01)	-
RDRH	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	0.5794	(<0.01)	(<0.01)
	1000 Mbps							
ECMP	0.093	-	0.32	-	(<0.05)	-	(<0.01)	-
RDRH	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	0.5794	(<0.01)	(<0.01)

Table 5 – Nemenyi post hoc test for 4-pod with Stag traffic and one TCP connection.

Tables 9 and 10 present the results for the 8-pod topology. The columns list the traffic patterns that showed a statistical difference with the application of Friedman's test. Table 9 shows that there is a significant difference from RDRH to Hedera at 100 Mbps and with ECMP and Hedera to 1000 Mbps. For five TCP connections, RDRH differs from Hedera and ECMP only for the Stag1_0.4_0.3 pattern.

The Wilcoxon Signed-Rank test was used to determine which algorithms obtained the best performance in throughput only for the cases in which there was a significant difference

100 Mbps								
	Random1		Random2		Random3		Random4	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
ECMP	0.98	-	(<0.01)	-	(<0.01)	-	(<0.01)	-
RDRH	(<0.01)	(<0.01)	0.75	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
1000 Mbps								
ECMP	0.98	-	(<0.01)	-	(<0.01)	-	(<0.01)	-
RDRH	(<0.01)	(<0.01)	0.75	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)

Table 6 – Nemenyi post hoc test for 4-pod with Random traffic and one TCP connection.

100 Mbps								
	Stag1_0.2_0.3		Stag1_0.5_0.1		Stag1_0.4_0.3		Stag2_0.4_0.3	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
ECMP	0.093	-	0.32	-	0.0487	-	(<0.01)	-
RDRH	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	0.5794	(<0.01)	(<0.01)
1000 Mbps								
ECMP	(<0.01)	-	(<0.01)	-	(<0.01)	-	(<0.01)	-
RDRH	(<0.01)	(<0.01)	(<0.01)	0.69	0.0935	0.0044	(<0.05)	(<0.01)

Table 7 – Nemenyi posthoc test for 4-pod with Stag traffic and five TCP connections.

100 Mbps								
	Random1		Random2		Random3		Random4	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
ECMP	0.98	-	(<0.01)	-	(<0.01)	-	(<0.01)	-
RDRH	(<0.01)	(<0.01)	0.75	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
1000 Mbps								
ECMP	0.23	-	0.47	-	0.0934	-	(<0.01)	-
RDRH	(<0.01)	(<0.01)	(<0.01)	(<0.01)	0.1660	(<0.01)	(<0.01)	(<0.01)

Table 8 – Nemenyi posthoc test for 4-pod with Random traffic and five TCP connections.

100 Mbps						
	Stag1_0.2_0.3		Random3		Random4	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
RDRH	0.012	0.294	-	-	-	-
1000 Mbps						
RDRH	-	-	0.694	0.039	(<0.01)	0.8499

Table 9 – Nemenyi posthoc test for 8-pod with one TCP connection.

	100 Mbps							
	Stag1_0.2_0.3		**Stag1_0.5_0.1**		**Stag1_0.4_0.3**		**Stag2_0.4_0.3**	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
RDRH	0.9998	0.9999	0.914	0.993	-	-	-	-
	1000 Mbps							
RDRH	1	0.3342	1	0.9994	(<0.01)	(<0.01)	-	-

Table 10 – Nemenyi posthoc test for 8-pod with five TCP connections.

detected through the paired Nemenyi post-hoc test. The formulated hypotheses for the WSRT are:

H_0 : Algorithm **A** has performance less than algorithm **B**.

H_a : Algorithm **A** has performance greater than algorithm **B**.

We compare the performance of RDRH with Hedera and ECMP in Tables 11 and 12 for the 4-pod. The p-values of the paired WSRT smaller than 0.05 evidences that we rejected H_0; consequently, RDRH presented the best performance in thirteen comparisons for 100 Mbps with one TCP connection (Table 11). With bandwidth 1000 Mbps, it presented a lower performance than the other solutions. Table 12 presents the results for tests with five connections. The performance of RDRH was superior in thirteen comparisons for 100 Mbps and eight comparisons for 1000 Mbps.

In Table 13, the RDRH presented a lower performance in the analyzed patterns and did not present statistical differences for the other patterns for 100Mbps. At 1000 Mbps bandwidth, RDRH betters ECMP only at random3 and Hedera at random4. For 8-pods with five connections (Table 14), RDRH is best only for the stag1_0.4_03 pattern.

	100 Mbps															
	Stag1_0.2_0.3		**Stag1_0.5_0.1**		**Stag1_0.4_0.3**		**Stag2_0.4_0.3**		**Random1**		**Random2**		**Random3**		**Random4**	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
RDRH	(<0.01)	(<0.01)	(<0.01)	(<0.01)	0.9967	0.9056	(<0.01)	(<0.01)	(<0.01)	(<0.01)	0.931	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
	1000 Mbps															
RDRH	1	1	1	0.9999	1	0.9998	0.9998	0.9999	1	1	1	1	1	1	1	1

Table 11 – Wilcoxon rank test for 4-pod with one TCP connection.

The hypothesis tests conducted in this work contributed to some conclusions regarding evaluating the prototype RDRH. First, the variation in the size of the fat-tree topology influences the allocation of elephant flows. In the topology with 4-pod, the RDRH performed better. For topology with 8-pod, the performance was inferior to the analyzed algorithms.

100 Mbps																
	Stag1_0.2_0.3		**Stag1_0.5_0.1**		**Stag1_0.4_0.3**		**Stag2_0.4_0.3**		**Random1**		**Random2**		**Random3**		**Random4**	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
RDRH	(<0.01)	(<0.01)	(<0.01)	(<0.01)	0.9967	0.9056	(<0.01)	(<0.01)	(<0.01)	(<0.01)	0.931	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)
1000 Mbps																
RDRH	1	1	(<0.01)	0.0848	(<0.01)	0.9992	0.9963	1	1	1	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)	(<0.01)

Table 12 – Wilcoxon rank test for 4-pod with five TCP connections.

100 Mbps						
	Stag1_0.2_0.3		**Random3**		**Random4**	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
RDRH	0.9724	0.9104	-	-	-	-
1000 Mbps						
RDRH	-	-	0.8784	0.0371	(<0.01)	0.3578

Table 13 – Wilcoxon rank test for 8-pod with one TCP connection.

100 Mbps								
	Stag1_0.2_0.3		**Stag1_0.5_0.1**		**Stag1_0.4_0.3**		**Stag2_0.4_0.3**	
	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP	Hedera	ECMP
RDRH	0.9998	0.9999	0.914	0.993	-	-	-	-
1000 Mbps								
RDRH	1	0.3342	1	0.9994	(<0.01)	(<0.01)	-	-

Table 14 – Wilcoxon rank test for 8-pod with five TCP connections.

Second, the variation in bandwidth from 100 Mbps to 1000 Mbps caused a balance between the evaluated solutions, as there was a reduction in the statistical difference between the solutions being assessed. Finally, the variation in the number of connections from 1 to 5 between origin and destination, it was observed that for 8-pod, there was a more significant statistical difference for the staggered pattern.

4.3.4 Computing time

The results presented in Section 4.3.2 showed that the allocation of elephant flows by the RDRH improved the performance of mice flows. The metrics of round-trip delay and loss rate were measured with mice traffic. These metrics are crucial for real-time applications. Therefore, the allocation time of the elephant flow generated by the routing solutions evaluated in this work is a parameter that must be considered.

Table 15 reports the average computing time necessary to solve different instances of the FTR problem for the 4-pod topology. The average time was calculated considering the transmissions of all traffic patterns between hosts lasting 60 seconds. The values in brackets represent the number of TCP flows.

The RDRH has a tradeoff, gains in performance as indicated in the previous section, but the time required to allocate the elephant's flow is greater than that of other solutions. The longer average times is due to the use of software solvers such as Cplex.

100 Mbps					
Hedera(1)	**Hedera(5)**	**ECMP(1)**	**ECMP(5)**	**RDRH(1)**	**RDRH(5)**
1.22	2.99	770.92	767.66	8288.28	8269.81
1000 Mbps					
1.73	3.63	764.06	766.14	8625.16	8547.80

Table 15 – Average computation time (ms) for 4-pod topology.

RDRH runs with computing time higher than other approaches due to the need to periodically re-execute the algorithm to obtain the next cycle's routing, which considers the new demands and those demands that ended in the previous iteration. This process can be improved by using the following strategy: each cycle's necessary adjustments correspond to inserting and removing some columns in the model of the previous period and re-optimizing it, which can generally be done efficiently in a few iterations of the simplex method (BERTSIMAS; TSITSIKLIS, 1997). This methodology is called local sensitivity analysis. A concern with the proposed RDRH is solving a linear program in each cycle in polynomial time. Therefore, using a solver requires additional processing time, but that should not compromise final results. Besides, it is impossible to perform experiments due to the high use of the processor by the Mininet emulator for large-scale fat-tree topologies (e.g., k = 16, k = 32).

4.4 LIMITATIONS OF THE RDRH AND TRAFFIC ENGINEERING SOLUTION

This section describes the main limitations of RDRH solution defined in this chapter.

The main limitations of the RDRH are:

- Approaches that seek solutions to problems with exponential complexity through heuristics do not provide an optimal solution. The solution obtained is an approximation of the problem.
- A linear programming problem can be solved in $O((n+d)^{1.5}nL)$ time (VAIDYA, 1989), where n is the number of variables, d is the number of constraints, and L is the number

of bits used to encode the input.

  – In the linear programming model used by the RDRH, the number of variables is $O(\ell|E|)$ and the number of constraint is $O(\ell|E|)$. As due to the fat-tree topology $|E| = k^3/2$, we conclude that the RDRH takes $O((\ell|E|)^{2.5}L) = O(\ell^{2.5}k^{7.5}L)$ time to solve the linear programming model, which can be prohibitive for situations with a large number of demands (ℓ) or with many ports on the switches (k).

- Considering that the prototype's execution is in real-time, the optimizer module will be running several times, so it cannot have very high complexity.

- It can generate paths that are not necessarily the shortest between the origin and destination of demand. Therefore, the RDRH may produce routes that are not the shortest path between the origin and destination of the demand, since during the drawing there is no limitation on which link to follow.

The main limitations of the TE solution are:

- SDN controller centralizes data plan management. Considering an increase in topology, the number of messages exchanged between the controller and the switches can cause congestion.

- In OpenFlow, the controller needs to calculate and install the forwarding rule of each new flow in real-time, which may cause a computation and congestion on both data and control planes in networks with large topologies (ALSAEEDI; MOHAMAD; AL-ROUBAIEY, 2019).

- The Mininet network emulator allows the reproduction of experiments with a level close to reality (LANTZ et al., 2015). However, it has limitations due to hardware/software that affect its performance and scalability (ORTIZ; LONDOñO; NOVILLO, 2016). When we perform tests with bandwidth up to 1000 Mbps, the performance degradation of the CPU in the experiments' server occurs. A shortcoming in our solution is the computing time when increasing network parameters, e.g., k-pod value and bandwidth. Moreover, using a solver requires an additional processing time that could increase the elephant flows' forward time.

4.5 CLOSING REMARKS

This chapter presented the steps followed in designing the experiments used to evaluate the proposed TE solution prototype: methodology, variables, and the metrics of the controlled experiment adopted in this evaluation.

5 HYBRID PREDICTION MODEL VALIDATION

This chapter presents in Section 5.1 the initial statistical analysis and traces the pre-processing of Facebook's data center. Besides, we describe the prediction models' adjustment parameters in Subsection 5.2. Concluding, we define the metrics used to evaluate the forecasting models in Subsection 5.3.

5.1 EXPLORATORY DATA ANALYSIS OF A FACEBOOK DATA CENTER'S TRAFFIC

This section presents descriptive statistics and hypothesis tests used in the dataset. It is of great relevance to use real data traces to evaluate the performance of forecasting models. The traces to be considered during the analysis were collected through twenty-four hours at the Altoona Facebook Data Center. We analyzed realistic workloads, and the raw data sample exceeds 500GB. Although the period appears insufficient, this did not impede research of relevance and impact in the computer network community (ROY et al., 2015; WANG et al., 2019; DAB; FAJJARI; AITSAADI, 2017). The dataset was distributed in three clusters:

- Cluster A - Database
- Cluster B - Web Servers
- Cluster C - Hadoop Servers

The Mysql servers (Database) store user data, whereas Web servers provide Web traffic, and Hadoop Servers manages offline analysis and data mining. (ROY et al., 2015) concluded that the data locality identifies Facebook's DCN traffic patterns: intra-cluster, inter-cluster, or inter-data center. The analysis adopted in this thesis follows the same premise as (ROY et al., 2015), with the addition of inter-cluster & inter-data center traffic, in combination with the consideration of another statistical analysis approach.

Table 16 provides an overview of all datasets used in the experiments and the descriptive analysis. The locality is defined by the source of traffic and the particular cluster. Traffic generated within the clusters is classified as intra-cluster. When there is an exchange of traffic among clusters, it is classified as inter-cluster. An inter-data center exchange occurs between buildings in the same geographical region or across geographical regions. When the traffic is externally targeted, it counts as an inter-cluster & inter-data center. The flows collected in the raw data traces were aggregated every 60 seconds.[1] This work adopted matching rules to compose elephant flows based on the 6-tuple composed of timestamps, source and destination addresses, source and destination ports, and the transport layer protocol. The Aggregate Flows column displays the total number of elephant flows. An elephant flow, as defined by this analysis, is considered that which exceeds a threshold of 100 MB (AL-FARES et al., 2010; ZHANG; CUI; ZHANG, 2017). The uniform sampling rate is 1:30,000 per-packet-sampling (ROY et al., 2015). The trace samples were collected from a specific machine every minute. We define a sample size of 5000 as one of the sources used for prediction. This value was defined because Cluster C of the Inter-data center dataset has only 5037 aggregate flows. Therefore, 4000 samples were assigned to the training dataset. The testing set was assigned to the remaining 1000 samples.

Source	Cluster	Unit	Aggregate Flows	Sample	Training	Test
Intra-cluster	A	KB	25929	5000	4000	1000
	B	KB	122928	5000	4000	1000
	C	KB	32848	5000	4000	1000
Inter-cluster	A	KB	295606661	5000	4000	1000
	B	KB	27440053	5000	4000	1000
	C	KB	5491	5000	4000	1000
Inter-data center	A	KB	16980	5000	4000	1000
	B	KB	19319	5000	4000	1000
	C	KB	5037	5000	4000	1000
Inter-cluster & Inter-data center	A	KB	24544	5000	4000	1000
	B	KB	19315	5000	4000	1000
	C	KB	5036	5000	4000	1000

Table 16 – Overview of datasets

5.1.1 Aggregate Elephant flows Analysis

The technique of aggregating flows into macro flows is used in TE to improve the performance of links in large-scale networks (KAMIYAMA et al., 2014). SDN makes it possible to control flows individually, but for DCNs, the increase in the number of flows that must be managed causes performance problems. Therefore, the aggregation performed in this work

[1] https://www.facebook.com/network-analytics

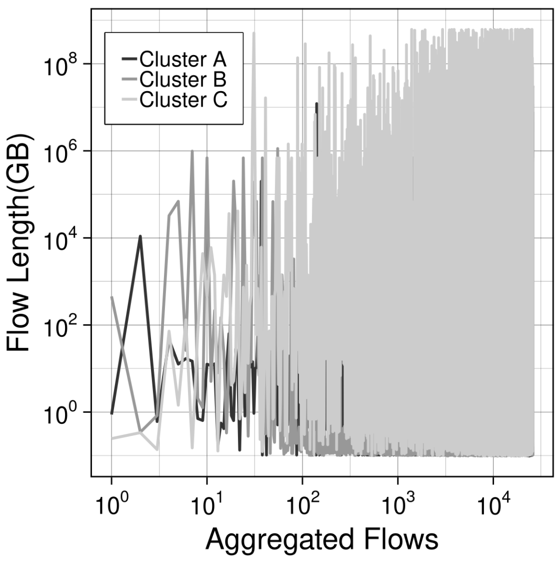
enables identifying elephant flows and allows these flows to be routed by TE mechanisms, as described in the previous chapter.

Figure 17 depicts the time series plots of the four traffic sources previously described. Each source contains the traffic generated by the three clusters. All plots are bursty, and the variability does not diminish significantly with aggregation. The horizontal axis represents the number of aggregate flows. Cluster C (Hadoop) shows a higher volume of flows than Cluster A (Database) and Cluster B (Web). Hadoop is a software framework for big data applications (PENG et al., 2014). (ROY et al., 2015) shows that cluster Hadoop generates most traffic on the data it analyzed. Therefore, it is following what was presented in Figure 17.

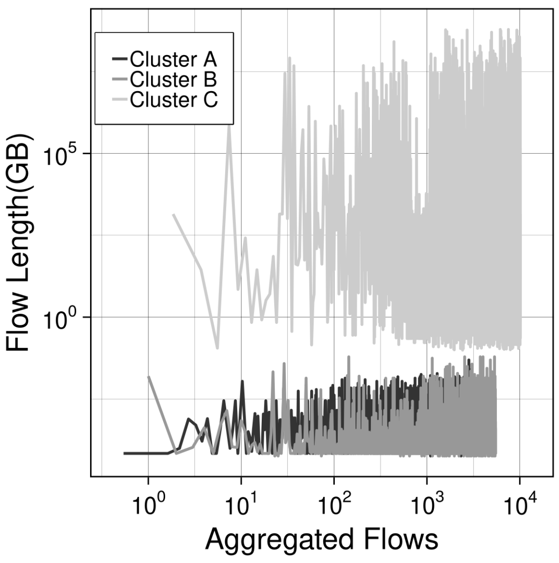
Table 17 includes the mean, standard deviation, skewness, kurtosis, and the coefficient of variation for each dataset. These values are calculated based on the elephant flows packet length. The high values in some clusters were due to the large volume of aggregate flows represented in KB. The development of the statistical analysis was performed with the R statistics software (CHAMBERS, 2002). Most prediction models implemented in this work use R packages due to its availability and ease of data processing. Also, there are support and documentation. The measures are initially represented in KB, but in Figure 17 were converted to GB. Cluster C represents data from the Hadoop servers. Based on the statistic mean, Cluster C identifies the most significant elephant flows. It was also concluded that Hadoop traffic is the predominant local cluster (Intra-cluster). Cluster B (Web servers) presented higher standard deviation values than those in the Intra-cluster, Inter-data center, and the Inter-cluster/Inter-data center scenarios. To evaluate the behavior of the series regarding the distributional shape, the skewness statistic[2] was observed as a right-skew in all datasets. Kurtosis analysis entails the distribution condition having a specified tail form. A leptokurtic distribution occurs when the Kurtosis value has an excess positive value. The coefficient of variation (CV) varies from a low of 701.80 to a high of 81126.84.

As displayed in Table 18, a second statistical test group identifies the test statistics values and *p*-values in parentheses. Tests performed on the datasets include normality, stationarity, randomness, and autocorrelation. The Weisberg–Bingham test is a variant of the Shapiro–Wilk test. The Weisberg–Bingham test was chosen based on the fact that it is powerful when analyzing asymmetric distributions (YAP; SIM, 2011). Experimental results determined that no dataset passed the test for normality. This test is essential to identify if the data are parametric or non-parametric. The clusters in bold font presented autocorrelation and were selected to

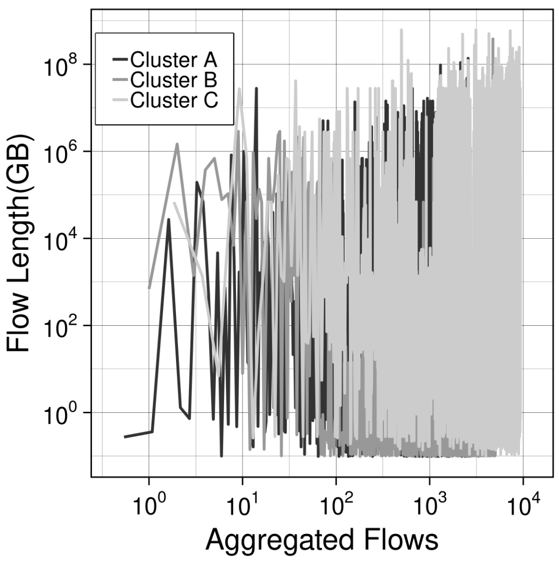
[2] The shape of the data distribution.



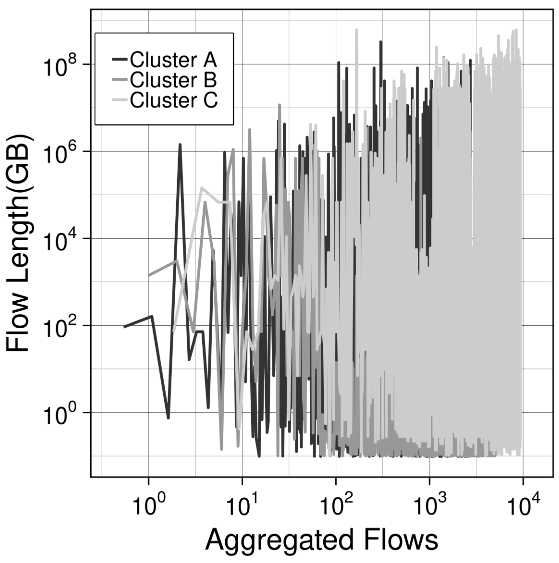
(a)



(b)



(c)



(d)

Figure 17 – Time series of all traffic sources: Intra-cluster (top left), Inter-cluster (top right), Inter-datacenter (bottom left) and Inter-cluster & Inter-data center (bottom right)

Locality	Cluster	Mean	St. Dev.	Skewness	Kurtosis	CV
Intra-cluster	A	63641.59	935880.8	24.36	810.21	1470.549
	B	322443.8	10164095	44.96	2296.09	3152.206
	C	4572952	38522770	12.31	166.96	842.4048
Inter-cluster	A	75.73	61437.93	2730.77	11873022	81126.84
	B	1370.1	605247.2	679.6	566585.7	44175.25
	C	3636361	29060621	15.14	268.58	799.1676
Inter-data center	A	346704	3579931	23.74	690.02	1032.562
	B	1153177	16004868	22.39	643.39	1387.893
	C	2906111	22428846	18.43	425.44	771.782
Inter-cluster& Inter-datacenter	A	429881.6	5274180	33.87	1588.34	1226.891
	B	1127077	16558904	23.29	679.37	1469.19
	C	3377108	23700759	15.93	331.57	701.8063

Table 17 – Descriptive statistics for all datasets. Units are represented in KB.

make the prediction.

Random testing runs concluded that two datasets (Intra- cluster C and Inter-cluster A) are random because the test's null hypothesis is for randomness. The remaining datasets are non-random. Therefore, it was determined that models that describe their structures to predict future values in the time series can be more successful (KATRIS; DASKALAKI, 2015). The ADF[3] test infers that all datasets are stationary. The Ljung–Box[4] test indicates that autocorrelation still exists in four datasets: Intra-cluster A, Intra-cluster B, Inter-cluster B, and Inter-data center C. Identifying high autocorrelation can improve the prediction. Therefore, the locations chosen (the bold lines) to carry out the forecast were those that, according to Table 18, presented data correlation.

Data center traffic characterization is essential to flow routing algorithms design. Long Range Dependence (LRD) detection is crucial to studying various protocols, algorithms, and characterizes emerging data communication environments (CROVELLA; BESTAVROS, 1997; GONG et al., 2005; WALDMAN; JR; RIBEIRO, 2004). LRD and non-linearity tests, as displayed in Table 19, were applied to the datasets. The Hurst often summarizes the LRD parameter (H), which defines auto-similarity (GONG et al., 2005). If $0.5 < H < 1$, the series will show the LRD. In the case of values close to 1, the LRD is stronger. On the contrary, the value of $H = 0.5$ indicates the absence of long-term memory. Using the H values from Table 19, except in Inter-data cen-

[3] An alternative hypothesis for the ADF test is statistical stationarity.
[4] An alternative hypothesis is autocorrelation.

Locality	Cluster	Normality Weisberg-B.	Randomness runs test	Stationarity ADF test	Autocorrelation Ljung–Box
Intra-cluster	**A**	**0.0407 (<0.01)**	**-4.2302 (<0.01)**	**-25.58 Lag = 29, (0.01)**	**10.008 (0.0015)**
	B	**0.0123 (<0.01)**	**-115.08 (<0.01)**	**-40.505 Lag = 43, (0.01)**	**13.582 (<0.01)**
	C	0.0955 (<0.01)	0.2096 (0.8339)	-32.165 Lag=32, (0.01)	1.25e-05 (0.9972)
Inter-cluster	A	0.3863 (<0.01)	-1.7153 (0.0863)	-42.871 Lag=43, (0.01)	0.1108 (0.7392)
	B	**0.0175 (<0.01)**	**-151.51 (<0.01)**	**-40.328 Lag=43, (0.01)**	**6.5866 (0.0102)**
	C	0.0997 (<0.01)	-5.7769 (<0.01)	-17.202 Lag=17, (0.01)	0.0031 (0.9552)
Inter-data center	A	0.0655 (<0.01)	-8.3958 (<0.01)	-25.608 Lag=25, (0.01)	0.3887 (0.533)
	B	0.0443 (<0.01)	-64.468 (<0.01)	-22.94 Lag=26, (0.01)	1.4864 (0.2228)
	C	**0.1015 (<0.01)**	**-4.1433 (<0.01)**	**-16.647 Lag=17, (0.01)**	**39.417 (<0.01)**
Inter-cluster & Inter-data center	A	0.0491 (<0.01)	-10.073 (<0.01)	-28.915 Lag=29,(0.01)	0.9135 (0.3392)
	B	0.0405 (<0.01)	-65.305 (<0.01)	-23.667 Lag=26,(0.01)	0.8579 (0.3543)
	C	0.1197 (<0.01)	-4.5097 (<0.01)	-17.184 Lag=17,(0.01)	0.2093 (0.6472)

Table 18 – Statistical tests for samples

ter C, Inter-cluster/Inter-data center C, it is concluded that all datasets are LRD. Non-linearity is performed through a White Neural Network (WNN) (LEE; WHITE; GRANGER, 1993), which verifies the significance of the linearity assumption. Four datasets presented non-linearity. This analysis is essential for the identification of the prediction models used in this thesis.

Source	Cluster	Hurst exponent	White test	(p-Value)	Suggested structure
Intra-cluster	A	0.5676	19.386	(<0.01)	Non-linear
	B	0.6141	1.9696	0.3735	Linear
	C	0.5291	0.4897	0.7828	Linear
Inter-cluster	A	0.5304	5.7812	0.0555	Linear
	B	0.6660	6813.6	(<0.01)	Non-linear
	C	0.5348	1.6803	0.4316	Linear
Inter-data center	A	0.5036	1.0393	0.5947	Linear
	B	0.6965	0.1644	0.9211	Linear
	C	0.4897	6.0104	0.0495	Non-linear
Inter-cluster & inter-data center	A	0.5115	31.157	(<0.01)	Non-linear
	B	0.6725	0.60981	0.7372	Linear
	C	0.4848	1.8669	0.3932	Linear

Table 19 – Tests for LRD and non-linearity

5.2 ADJUSTMENT PARAMETERS OF FORECAST MODELS

The previously analyzed data was grouped at four locations, each containing packet size information for applications running on each cluster. The selection of the applied prediction models was based on the results shown in Tables 18-19. Table 19 identifies the type of data structure (linear or non-linear). As previously stated, four datasets were selected as the criteria for applying the forecast models. The selection was based on the Ljung–Box test (18), which verified the presence of autocorrelation in a time series. The number of periods for short-term prediction (one-step-ahead) in all models was 60 s. In all the models, data are normalized in [0,1].

Locality	Cluster	Train	Test	ARIMA	MLP	RBF	Garch
Intra-cluster	A	4000	1000	(0,0,0)	(1,20,1)	(1,2,1)	(1,1)
	B	4000	1000	(0,0,0)	(1,2,1	(1,50,1)	(1,1)
Inter-cluster	B	4000	1000	(0,0,0)	(1,1,1)	(1,20,1)	(1,1)
Inter-dc	C	4000	1000	(0,0,0)	(1,50,1)	(1,50,1)	(1,1)

Table 20 – Parameters of the prediction models 1

As previously stated, the one-step ahead prediction was conducted by incorporating five thousand dataset samples. In Tables 20 and 21, the dataset was divided into two sets: training and test. Eighty percent of the samples were allocated to training the models and twenty per-

Locality	Cluster	Train	Test	Farima-MLP	Farima-RNN	LSTM
Intra-cluster	A	4000	1000	(3,0)/(1,50,1)	(3,0)/(2,2,1)	(50,32,1)
	B	4000	1000	(0,1)/(1,20,1)	(0,1)/(50,50,1)	(100,32,1)
Inter-cluster	B	4000	1000	(3,0)/(1,2,1)	(0,3)/(2,2,1)	(30,32,1)
Inter-dc	C	4000	1000	(1,1)/(1,10,1)	(1,1)/(50,50,1)	(50,32,1)

Table 21 – Parameters of the prediction models 2

cent to the test. Following the prediction error metrics described in the next section, Tables 20 and 21 identify the parameter combinations that present the lowest error rate. For ARIMA modeling, the model's fitting was done using the R package forecast. The Auto.ARIMA function returns the best set of parameters for the model. In all cases, the values of the parameters (p, d, q) indicate a white noise, which is identified as ARIMA (0, 0, 0) (HYNDMAN; ATHANASOPOULOS, 2017). The *rugarch* package (GHALANOS; GHALANOS; RCPP, 2017) aims to provide a flexible GARCH modeling environment using the R language, which recently turned into an essential tool for computational insights, perception, and data science (ANGADI; KULKARNI, 2015).

A normal distribution is used to model innovations; that is, the error (difference from the actual and predicted value) follows a normal distribution. The parameter set up of (1, 1) is adequate for capturing the conditional variance of the errors (ANAND; SCOGLIO; NATARAJAN, 2008; KATRIS; DASKALAKI, 2018). For ANNs models, the embedding dimension generated by the False Nearest Neighbors (FNN) algorithm (KATRIS; DASKALAKI, 2015) was used to select the input nodes. The experiments were performed with 1, 2, 10, 20, or 50 nodes in the hidden layer. During training, the final solution was based on a minimum RMSE calculated for the 4000 most recent training sample observations. The training was performed using back-propagation with an adaptive gradient descent algorithm and the 4000 epochs of training. Lastly, sigmoid was selected for the activation function of the hidden layer and linear output. For the RBF architecture, the analysis incorporated the same node selection procedure as with the FNN; the training was performed by the "RadialBasisLearning" function.

The implementation of the ANN models and Elman RNN was through the Stuttgart Neural Network Simulator (SNNS).[5] The "Auto ARFIMA" function of the *rugarch* package adjusts the parameters of the FARIMA model. Due to the skewness of the traces shown in Table 17, the fitting of the FARIMA model was performed with Student's-t distribution with skewness

[5] https://cran.r-project.org/web/packages/RSNNS/RSNNS.pdf.

"sstd". For nonlinear optimization procedures, the maximum likelihood estimation requires the *nlminb* optimizer or the augmented Lagrange method.

The LSTM model is implemented using the Keras library (CHOLLET, 2018) on top of TensorFlow machine learning framework (ABADI et al., 2016). The architecture has an LSTM layer with 32 units and a dense single layer with the sigmoid activation function. By tuning the network's parameters varying the *lookback value* (i.e., how many timesteps back the input data should go) in 30, 50, and 100 on 20 epochs. Table 21 shows in the column labeled LSTM the parameters that presented lower mean square error.

5.3 PERFORMANCE ANALYSIS FOR FORECASTING MODELS

This section describes the accuracy metrics used to evaluate the proposed model's performance, and then the results are evaluated.

5.3.1 Prediction accuracy

Section 5.2 describes the application of the prediction models. The performance analysis considers measures of accuracy as originated by the prediction error. RMSE, MAE, and MAPE (SHMUELI; LICHTENDAHL, 2015) are the most popular performance metrics for measuring predictive accuracy. The primary percentage error measure is MAPE because it has the advantage of an independent scale. The disadvantages of the MAPE and RMSE measures are that they give more weight to significant errors (SHMUELI; LICHTENDAHL, 2015).

Selecting the most appropriate model is difficult when various forecasting models are available for the prediction analysis of several datasets. In this work, the goal is to adopt the model that presents the best performance to forecast elephant flows in DCNs. The prediction model selection process considered the model's ranking, calculating the average position, and identifying the corresponding standard deviation (KATRIS; DASKALAKI, 2015).

To illustrate the rank model: suppose N prediction models are applied to k datasets. For each dataset, all models are ranked according to their RMSE, MAE, and MAPE values. Equation (5.1) uses rank to calculate the average position (AP) of each model, which summarizes its performance overall data sets. In (5.2), the standard deviation of the positions (SDP) calculates the variability of the positions that each model has for different datasets (KATRIS;

DASKALAKI, 2015). For each model i, is computed

$$AP_i = \frac{1}{k}\sum_{j=1}^{k} P_{ij} \tag{5.1}$$

where P_{ij} indicates the position of model i for dataset j, and

$$SDP_i = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(P_{ij} - AP_i)^2} \tag{5.2}$$

The model with the lowest possible concurrent AP and SDP was selected. The results were achieved using a defined linear utility function to evaluate the models (KATRIS; DASKALAKI, 2015).

For each model i the utility function U is defined by (5.3):

$$U_i = AP_i + SDP_i \tag{5.3}$$

The utility value was calculated for each performance metric: RMSE, MAE, and MAPE. An Average Ranking value is calculated as in (5.4):

$$ARv_i = \frac{1}{3}(U_i^{RMSE} + U_i^{MAE} + U_i^{MAPE}), \tag{5.4}$$

where a lower ARv value for a model indicates a more accurate prediction model (KATRIS; DASKALAKI, 2015).

The next chapter presents the evaluation of the results considering the metrics described above.

5.3.2 Data Analysis and Discussion of the Prediction Models

All forecasting models analyzed in 5.2 were applied to four selected datasets. The datasets that present autocorrelation are identified in bold font in Table 18. These four datasets were used to evaluate and compare the elephant flow prediction models.

Tables 22 and 23 show the performance results for each locality and their respective cluster arrangement. Both tables also show the average position within the seven prediction models, the standard deviation of each position, and each ranking's mean values. Their ranking position is also described in parentheses.

Cluster A of the Intra-cluster source is the only database selected from the sources because of its autocorrelation in Table 18. In the RMSE and MAE metrics, the models based on RNNs present errors smaller than the errors from other models in the Intra-cluster source. In Intra-cluster B, the analysis follows the same conclusion described previously. It is also worth noticing that the LSTM model performs poorly in terms of MAPE, with values 100.2095 and 100.0985 for Intra-cluster A and Intra-cluster B, respectively (see Table 23). Higher values of metrics indicate poor performance. Reported disadvantages of MAPE (KIM; KIM, 2016) are associated with fluctuations when the original time series carries small values. If the actual values are minimal (usually less than one), MAPE yields huge percentage errors, i.e., outliers.

In the Inter-cluster scenario, FARIMA-RNN outperforms all other models in RMSE and MAE. Also, this dataset presents non-linearity; for this reason, the ARIMA and GARCH models presented worse performance compared with the other models evaluated in the rank in Table 22. In Inter-data center scenario, FARIMA-RNN outperforms all the other models as indicated in Table 23. Overall, the reason that the LSTM model did not achieved a better position was due to MAPE.

The evaluation of the forecasting approaches is also displayed graphically in Figure 18 where the values for $1/U_i^{RMSE}$, $1/U_i^{MAE}$, $1/U_i^{MAPE}$ and $1/ARv_i$ (Average Ranking Equation (5.4)) are the inverse of the utility value $(1/U_i)$. The inverse value of the utility metric (U) indicates that higher values represent better performance. Figure 18 was plotted using the data from Table 24. Thereby, one may conclude that the FARIMA-MLP and LSTM models display similar performance, giving 0.25 and 0.23 as the reciprocal of their ARvi, as can be seen from Table 24. It is also worth noticing that the FARIMA-RNN model performs well according to all error metrics, and the RMSE suggests the absence of some extreme deviations between the actual and forecasted values.

The training and testing times are presented in Tables 25 and 26, parenthesis to the left (training) and right (testing) with measure times taken in seconds. Hybrid models have higher training times than other models for two reasons: the first is the process of fitting the FARIMA model to the data, followed by creating a neural network with the residuals of step one. The LSTM model presented higher training time due to the number (twenty) of epochs used by tuning the model, according to the description in Section 5.2. Lastly, from the hybrid models, the FARIMA-RNN outperformed all models in testing time.

This chapter presented the steps followed in designing the experiments used to evaluate the hybrid forecasting model FARIMA-RNN and the prototype of the proposed TE solution: the description of the stages of analysis and exploration of the real traces; the statistical techniques applied during the investigation of results; methodology; variables; and the metrics

Source	Cluster		Arima	MLP	RBF	Garch
Intra-cluster	A	RMSE	1.724	0.1083	0.1	1.434
		MAE	0.097	0.0368	0.0696	0.2475
		MAPE	100	100.175	205.281	283.3561
	B	RMSE	0.71	0.2348	0.2156	1.417
		MAE	0.064	0.0525	0.1317	0.2587
		MAPE	100	99.9656	318.8574	335.7927
Inter-cluster	B	RMSE	1.237	0.5085	0.2402	1.6129
		MAE	0.268	0.1876	0.1039	0.7232
		MAPE	100	98.8698	63.1386	70.6346
Inter-data center	C	RMSE	1.02	1.2218	0.3754	0.7133
		MAE	0.225	1.1186	0.2592	0.541
		MAPE	100	1266.691	194.1222	79.6104
RMSE evaluation	Avg. Position		6.00	5.25	3.00	6.25
	St. Dev.		0.82	1.26	0.82	0.96
	Value		6.82	6.51	3.82	7.21
MAE evaluation	Avg. Position		5.25	5.00	4.75	6.75
	St. Dev.		0.96	1.41	1.26	0.50
	Value		6.21	6.41	6.01	7.25
MAPE evaluation	Avg. Position		4.00	4.75	5.25	5.00
	St. Dev.		1.41	1.71	1.50	2.45
	Value		5.41	6.46	6.75	7.45
Model ranking			**6.15(5)**	**6.46(6)**	**5.52(4)**	**7.30(7)**

Table 22 – Performance Evaluation of Forecasting Approaches 1

of the controlled experiment adopted in this evaluation.

5.3.3 FARIMA-RNN limitation

- Machine learning models can present the overfitting problem, which consists in the model being unable to generalize to new data. We indicate in Chapter 6, Section 6.3 of future works, we indicate a technique proposed in the literature to solve this problem.

Source	Cluster		Farima-MLP	Farima-RNN	LSTM
Intra-cluster	A	RMSE	0.0871	0.0238	0.0304
		MAE	0.0054	0.0079	0.0305
		MAPE	9.3656	22.6287	100.2095
	B	RMSE	1.0293	0.0159	0.0442
		MAE	0.0489	0.0156	0.0443
		MAPE	12.878	45.9622	100.0985
Inter-cluster	B	RMSE	0.3905	0.0815	0.3172
		MAE	0.0788	0.0425	0.132
		MAPE	48.254	49.3084	104.3633
Inter-datacenter	C	RMSE	0.3971	0.0792	0.3639
		MAE	0.1527	0.0224	0.1491
		MAPE	101.275	27.5416	105.0217
RMSE evaluation	Avg. Position		4.25	1.00	2.25
	St. Dev.		1.26	0.00	0.50
	Value		5.51	1.00	2.75
MAE evaluation	Avg. Position		2.25	1.25	2.75
	St. Dev.		0.96	0.50	0.96
	Value		3.21	1.75	3.71
MAPE evaluation	Avg. Position		1.75	1.75	5.50
	St. Dev.		1.50	0.50	1.00
	Value		3.25	2.25	6.50
Model ranking			**3.99(2)**	**1.67(1)**	**4.32(3)**

Table 23 – Performance Evaluation of Forecasting Approaches 2

	ARIMA	MLP	RBF	GARCH	FARIMA-MLP	FARIMA-RNN	LSTM
1/U(RSME)	0.15	0.15	0.26	0.14	0.18	1.00	0.36
1/U(MAE)	0.16	0.16	0.17	0.14	0.31	0.57	0.27
1/U(MAPE)	0.18	0.15	0.15	0.13	0.31	0.44	0.15
1/AR(vi)	0.16	0.15	0.18	0.14	0.25	0.60	0.23

Table 24 – Performance Evaluation of Forecasting Approaches

Locality	Cluster	ARIMA	MLP	RBF	Garch
Intra-cluster	A	(0.74)(0.002)	(4.82)(0.002)	(4.66)(0.002)	(3.06)(0.009)
	B	(0.24)(0.002)	(4.84)(0.002)	(4.63)(0.002)	(1.27)(0.05)
Inter-cluster	B	(0.50)(0.003)	(5.27)(0.002)	(4.90)(0.002)	(3.93)(0.003)
Inter-dc	C	(0.26)(0.002)	(5.46)(0.002)	(4.67)(0.002)	(1.68)(0.003)

Table 25 – Training and testing time (s)

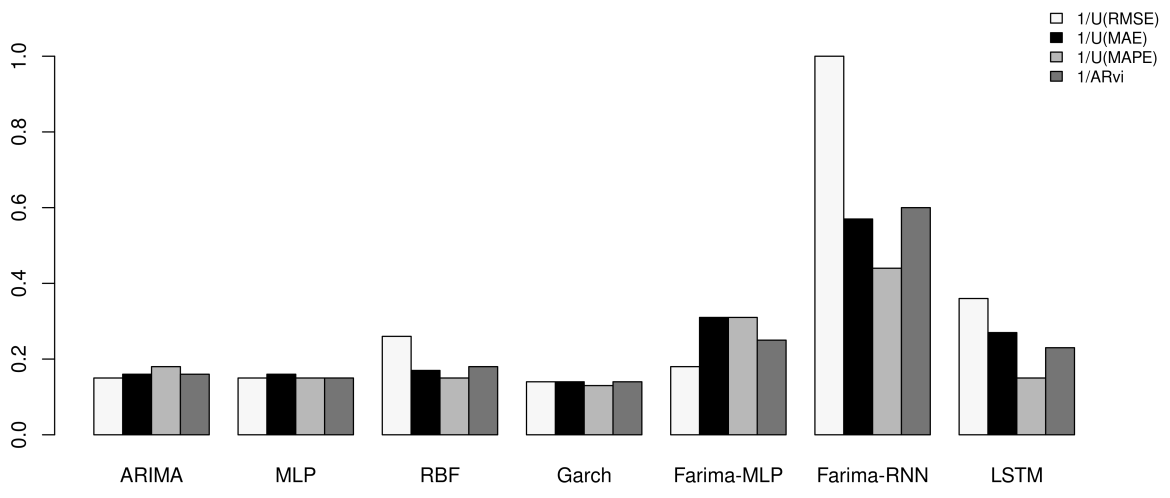


Figure 18 – Evaluation of Predictors approaches

Locality	Cluster	Farima-MLP	Farima-RNN	LSTM
Intra-cluster	A	(5.87)(0.002)	(100.36)(0.001)	(61.41)(0.063)
	B	(7.69)(0.002)	(1.64)(0.001)	(74.35)(0.062)
Inter-cluster	B	(10.26)(0.003)	(6.00)(0.001)	(54.68)(0.054)
Inter-dc	C	(2.03)(0.002)	(0.40)(0.001)	(63.89)(0.066)

Table 26 – Training and testing time (s)

5.4 CLOSING REMARKS

This chapter presented the steps followed in designing the experiments used to evaluate the proposed hybrid prediction model: the description of the stages of analysis and exploration of the real traces; the statistical techniques applied during the investigation of results; methodology; variables; and the metrics of the experiments adopted in this evaluation.

6 CONCLUSION

In this chapter, the hybrid forecasting model and the elephant flow scheduling heuristic proposed in this thesis are briefly revisited. In addition, the contributions of this research to the literature on forecasting and scheduling elephant flows are described. Also listed are responses to research questions. Finally, extensions and integrations of the model are discussed with heuristics to be explored in future works.

This thesis hypothesis analyzes and predicts actual data from Facebook's DCN located in Altoona, Pennsylvania (USA). Traces were pre-processed to identify elephant flows and forecast them on a short-term basis. In light of this, a time series prediction model with statistical methods and pattern recognition was implemented. This combination generated a new hybrid prediction model built with FARIMA and a Recurrent Neural Network, called FARIMA-RNN.

Moreover, we propose the RDRH solution for scheduling elephant flows in DCN with fat-tree topology. This solution uses linear relaxation with randomized rounding to route elephant flows, thereby distributing traffic more evenly to avoid overloading links.

The performance of the FARIMA-RNN model was compared to the ARIMA, GARCH, MLP, RBF, FARIMA-MLP, and LSTM models. Finally, existing elephant flow scheduling strategies were evaluated and compared with the proposed RDRH solution.

The results obtained have determined that the FARIMA-RNN model presents an advantage over the other predictors. The proposed model allows the lowest error rate in the evaluated metrics and the prediction of non-linear data.

The results obtained from the evaluation of the RDRH show that the performance is superior for 4-pod topology. Although the throughput performance is balanced for the solutions in the 8-pod topology, in general, there was a good performance for the metrics of round-trip delay and loss rate.

The main contributions of this thesis and future works are discussed in the following sections.

6.1 DISCUSSION ON CONTRIBUTIONS

The main findings of this thesis, presented initially in Chapter 1, are discussed below.

- The processing and time series analysis of real traces from the Altoona, Pennsylvania Facebook Data Center to identify scenarios where elephant flow prediction can be realized. The Facebook DCN database was made available in 2017. This made it possible to carry out an exploratory statistical analysis and considering the variation in the size of the packet in the flows as a time series. To the best of our knowledge, this thesis is the first work to carry out such analysis.

- A new hybrid prediction model built with FARIMA and a Recurrent Neural Network is proposed.

  The FARIMA-RNN hybrid forecast model proposal brought together the advantages that each model presents. The central insights were this combination to compose a model that performs better than those that exist in the state-of-the-art.

- Flow scheduling for DCNs based on an SDN that uses a randomized rounding heuristic is developed.

  To the best of our knowledge, the randomized rounding technique had not been explored to allocate elephant flows in DCNs. SDN paradigms enable new traffic engineering applications that exploit the global network view and flow patterns for better traffic control and management. In light of this, we propose a prototype for scheduling elephant flows in DCNs called RDRH. We performed experiments that reproduced a DCN's topology in an emulated environment. The source code is available in the Github repository [1].

6.2 RESPONSES TO RESEARCH QUESTIONS

In Chapter 1 of this thesis, the main research questions that guided this work were presented. In this section, these questions are reviewed and answered according to the results analyzed in this chapter. For the reader's convenience, the questions are listed in subsections, followed by their respective answers.

[1] <https://github.com/jmbezerra/exp_rdrhfattree.git>

6.2.1 RQ1: Which procedures are required to identify elephant flows in real data traces?

The DCN Facebook data traces used in this work were pre-processed and aggregated according to the 6-tuple composed of timestamps, source and destination addresses, source and destination ports, and the transport layer protocol. In the flow aggregation stage, the packet was used in the elephant flow classification based on the definition of elephant flow for those that exceed 100 MB.

6.2.2 RQ2: How to design a hybrid model for elephant flows prediction in DCNs?

For the design or development of a time series forecasting model, the necessary initial procedure is to analyze the series' behavior and check for patterns, for example, seasonality, trend, etc. In elephant flow traffic, due to the pattern showing LRD behavior, the FARIMA model performed better. The option chosen by Elman's RNN was due to the non-linearity of the data and the capacity that RNNs have in their architecture to store previous information and use it for forecasting. Before choosing Elman's RNN, tests were performed with a Jordan RNN, but Elman's presented the least error in the forecast results. A hybrid model, based on the characteristics described above, enhances the prediction of elephant flows.

6.2.3 RQ3: How does the proposed hybrid prediction model compare with the related state-of-the-art models?

To evaluate the performance of the proposed hybrid model, we followed the methodology presented in (KATRIS; DASKALAKI, 2015). The conventional metric for evaluating prediction models used by the state of the art is the forecast error, which is the difference between the predicted and the real value of the series. The most popular are the RMSE, MAE, and MAPE. These metrics are those most used by the research community as they cover statistics that allow a detailed analysis of the forecast error. To evaluate the sets of the three error metrics for each model, the average position and standard deviation were calculated. The sum of these values generated the utility value for each model. The models were ranked based on the smallest forecasting error.

6.2.4 RQ4: How does the proposed randomized rounding heuristic compare with the related state-of-the-art solutions?

The RDRH solution was compared with the Hedera and ECMP solutions to evaluate throughput, round-trip delay, and loss. These metrics were evaluated on an emulated testbed. Besides, the results were also evaluated with inferential statistics through hypothesis tests.

6.2.5 RQ5: What is the effect of applying a randomized rounding heuristic when it is used with different traffic patterns?

The RDRH heuristic was integrated with an SDN solution in order to improve TE in DCNs. We evaluated the solutions with the two traffic patterns presented in the papers by (AL-FARES et al., 2010; ZHANG; CUI; ZHANG, 2017; CHAO; LIN; CHEN, 2019). Although these patterns are synthetic, they reproduce well the traffic pattern behavior in data centers. Our RDRH prototype showed better performance in throughput in some cases, for example, with the Random traffic pattern. For round-trip delay and loss rate metrics, our proposal performed better in most of the scenarios evaluated. We have achieved a better use of the links and response time through the metrics previously described.

6.2.6 RQ6: What is the effect of optimizing elephant flow only? Does this improve network performance?

Although the RDRH focuses only on elephant flow, the performance evaluation results improved mice flows' performance. This was verified through round-trip delay and loss metrics, evaluated in Section 4.3.

6.3 FUTURE WORKS

Next, we will indicate future works that can be carried out potentially to improve on our approach.

- The regularization technique can control the overfitting phenomenon (BISHOP, 2006). This approach works by adding a penalty term to the error function (the difference between actual and adjusted values) after the fitting process. The purpose of this technique is to discourage the coefficients from reaching large values.

- Investigate the combination of FARIMA with other models of deep neural networks, such as LSTM. Further research (ZHAOWEI et al., 2020) could be conducted to investigate whether there are hybrid prediction models that reduce the problem of overfitting in learning-based models.

- Determine the advantages for DCNs of the SDN controller having advance information on elephant flow. Survey the requirements for implementing the FARIMA-RNN model to develop a traffic engineering application for integration with an SDN solution.

- Determine the other algorithmic approaches that have not yet been explored for scheduling elephant flows in DCNs. Recently, related works (LI; LU; FU, 2020; JINGWEN; MUQING; XIAOLAN, 2019) have proposed new heuristics for the scheduling of elephant flows. We will investigate the use of greedy heuristics and genetic algorithms.

REFERENCES

ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M.; KUDLUR, M.; LEVENBERG, J.; MONGA, R.; MOORE, S.; MURRAY, D. G.; STEINER, B.; TUCKER, P.; VASUDEVAN, V.; WARDEN, P.; WICKE, M.; YU, Y.; ZHENG, X. Tensorflow: A system for large-scale machine learning. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016. p. 265–283. ISBN 978-1-931971-33-1. Available at: <https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi>.

ABDELMONIEM, A. M.; BENSAOU, B. Reconciling mice and elephants in data center networks. In: *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. [S.l.: s.n.], 2015. p. 119–124.

AKYILDIZ, I. F.; LEE, A.; WANG, P.; LUO, M.; CHOU, W. A Roadmap for Traffic Engineering in SDN-OpenFlow Networks. *Computer Networks*, Elsevier North-Holland, Inc., New York, NY, USA, v. 71, p. 1–30, Oct. 2014. ISSN 1389-1286. Available at: <http://dx.doi.org/10.1016/j.comnet.2014.06.002>.

AKYILDIZ, I. F.; LEE, A.; WANG, P.; LUO, M.; CHOU, W. Research challenges for traffic engineering in software defined networks. *IEEE Network*, v. 30, n. 3, p. 52–58, May 2016. ISSN 0890-8044.

AL-FARES, M.; LOUKISSAS, A.; VAHDAT, A. A scalable, commodity data center network architecture. In: *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. New York, NY, USA: ACM, 2008. (SIGCOMM '08), p. 63–74. ISBN 978-1-60558-175-0. Available at: <http://doi.acm.org/10.1145/1402958.1402967>.

AL-FARES, M.; RADHAKRISHNAN, S.; RAGHAVAN, B.; HUANG, N.; VAHDAT, A. Hedera: Dynamic flow scheduling for data center networks. In: *Proceedings of the 7th USENIX Conference*. Berkeley, CA, USA: USENIX Association, 2010. (NSDI'10), p. 19–19. Available at: <http://dl.acm.org/citation.cfm?id=1855711.1855730>.

ALDHYANI, T. H. H.; ALRASHEEDI, M.; ALQARNI, A. A.; ALZAHRANI, M. Y.; BAMHDI, A. M. Intelligent hybrid model to enhance time series models for predicting network traffic. *IEEE Access*, v. 8, p. 130431–130451, 2020.

ALIYU, A. L.; ANEIBA, A.; PATWARY, M.; BULL, P. A trust management framework for software defined network (sdn) controller and network applications. *Computer Networks*, v. 181, p. 107421, 2020. ISSN 1389-1286. Available at: <http://www.sciencedirect.com/science/article/pii/S1389128620311105>.

ALSAEEDI, M.; MOHAMAD, M. M.; AL-ROUBAIEY, A. A. Toward adaptive and scalable openflow-sdn flow control: A survey. *IEEE Access*, v. 7, p. 107346–107379, 2019.

ALVAREZ-HORCAJO, J.; LOPEZ-PAJARES, D.; MARTINEZ-YELMO, I.; CARRAL, J. A.; ARCO, J. M. Improving multipath routing of tcp flows by network exploration. *IEEE Access*, v. 7, p. 13608–13621, 2019. ISSN 2169-3536.

ANAND, N. C.; SCOGLIO, C.; NATARAJAN, B. Garch; non-linear time series model for traffic modeling and prediction. In: *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*. [S.l.: s.n.], 2008. p. 694–697. ISSN 1542-1201.

ANDERSON, T.; CROVELLA, M.; DIOT, C. *Internet Measurement: Past, Present and Future*. 2004. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.457.6531&rep=rep1&type=pdf>.

ANDREYEV, A. *Introducing data center fabric, the next-generation Facebook data center network*. 2014. Available at: <https://code.facebook.com/posts/360346274145943/>.

ANGADI, M. C.; KULKARNI, A. P. Time series data analysis for stock market prediction using data mining techniques with r. *International Journal of Advanced Research in Computer Science*, v. 6, n. 6, 2015. <https://doi.org/10.26483/ijarcs.v6i6.2528>.

ARDALANI-FARSA, M.; ZOLFAGHARI, S. Chaotic time series prediction with residual analysis method using hybrid elman–narx neural networks. *Neurocomputing*, v. 73, n. 13, p. 2540 – 2553, 2010. ISSN 0925-2312. Pattern Recognition in Bioinformatics Advances in Neural Control. Available at: <http://www.sciencedirect.com/science/article/pii/S0925231210002687>.

BARAHONA, F.; CHUDAK, F. A. Near-optimal solutions to large-scale facility location problems. *Discrete Optimization*, v. 2, n. 1, p. 35 – 50, 2005. ISSN 1572-5286. Available at: <http://www.sciencedirect.com/science/article/pii/S1572528605000034>.

BASTAM, M.; SABAEI, M.; YOUSEFPOUR, R. A scalable traffic engineering technique in an sdn-based data center network. *Transactions on Emerging Telecommunications Technologies*, v. 29, n. 2, p. e3268, 2018. E3268 ett.3268. Available at: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3268>.

BENSON, T.; ANAND, A.; AKELLA, A.; ZHANG, M. Understanding data center traffic characteristics. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 40, n. 1, p. 92–99, Jan. 2010. ISSN 0146-4833. Available at: <http://doi.acm.org/10.1145/1672308.1672325>.

BENSON, T.; ANAND, A.; AKELLA, A.; ZHANG, M. Microte: Fine grained traffic engineering for data centers. In: *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*. New York, NY, USA: ACM, 2011. (CoNEXT '11), p. 8:1–8:12. ISBN 978-1-4503-1041-3. Available at: <http://doi.acm.org/10.1145/2079296.2079304>.

BERGMEIR, C.; BENÍTEZ, J. M. Neural networks in r using the stuttgart neural network simulator: Rsnns. *Journal of Statistical Software*, v. 46, n. 7, 2012.

BERTSIMAS, D.; TSITSIKLIS, J. N. *Introduction to linear optimization*. [S.l.]: Athena Scientific Belmont, MA, 1997.

BIALON, P. A randomized rounding approach to a k-splittable multicommodity flow problem with lower path flow bounds affording solution quality guarantees. *Telecommunication Systems*, v. 64, n. 3, 2017.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: springer, 2006.

BRANDT, S.; FOERSTER, K.-T.; WATTENHOFER, R. Augmenting flows for the consistent migration of multi-commodity single-destination flows in sdns. *Pervasive and Mobile Computing*, v. 36, p. 134 – 150, 2017. ISSN 1574-1192. Special Issue on Pervasive Social Computing. Available at: <http://www.sciencedirect.com/science/article/pii/S1574119216302322>.

CHAMBERS, J. *he R Project for Statistical Computing*. 2002. Available at: <https://www.r-project.org/>.

CHANDRA, R. Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time-series prediction. *IEEE transactions on neural networks and learning systems*, IEEE, v. 26, n. 12, p. 3123–3136, 2015.

CHAO, S.; LIN, K. C.; CHEN, M. Flow classification for software-defined data centers using stream mining. *IEEE Transactions on Services Computing*, v. 12, n. 1, p. 105–116, Jan 2019. ISSN 1939-1374.

CHIESA, M.; KINDLER, G.; SCHAPIRA, M. Traffic engineering with equal-cost-multipath: An algorithmic perspective. *IEEE/ACM Transactions on Networking*, v. 25, n. 2, p. 779–792, 2017.

CHOLLET, F. *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. [S.l.]: MITP-Verlags GmbH & Co. KG, 2018.

CHOLLET, F.; ALLAIRE, J. J. *Deep Learning with R*. 1st. ed. Greenwich, CT, USA: Manning Publications Co., 2018. ISBN 161729554X, 9781617295546.

CROVELLA, M. E.; BESTAVROS, A. Self-similarity in world wide web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, v. 5, n. 6, p. 835–846, 1997.

CUI, L.; YU, F. R.; YAN, Q. When big data meets software-defined networking: Sdn for big data and big data for sdn. *IEEE Network*, v. 30, n. 1, p. 58–65, 2016.

DAB, B.; FAJJARI, I.; AITSAADI, N. Online-batch joint routing and channel allocation for hybrid data center networks. *IEEE Transactions on Network and Service Management*, v. 14, n. 4, p. 831–846, 2017.

DAI, B.; XU, G.; HUANG, B.; QIN, P.; XU, Y. Enabling network innovation in data center networks with software defined networking: A survey. *Journal of Network and Computer Applications*, v. 94, p. 33 – 49, 2017. ISSN 1084-8045. Available at: <http://www.sciencedirect.com/science/article/pii/S1084804517302278>.

DALMAZO, B. L.; VILELA, J. P.; CURADO, M. Performance analysis of network traffic predictors in the cloud. *Journal of Network and Systems Management*, v. 25, n. 2, p. 290–320, Apr 2017. ISSN 1573-7705. Available at: <http://dx.doi.org/10.1007/s10922-016-9392-x>.

DELL'ACQUA, P.; BELLOTTI, F.; BERTA, R.; GLORIA, A. D. Time-aware multivariate nearest neighbor regression methods for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, v. 16, n. 6, p. 3393–3402, 2015.

DEMUTH, H. B.; BEALE, M. H.; JESS, O. D.; HAGAN, M. T. *Neural Network Design*. 2nd. ed. Stillwater, OK, USA: Martin Hagan, 2014. ISBN 0971732116.

ELMAN, J. L. Finding structure in time. *Cognitive Science*, v. 14, n. 2, p. 179–211, 1990. Available at: <https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1>.

FANG, W.; LU, Z.; WU, J.; CAO, Z. Rpps: A novel resource prediction and provisioning scheme in cloud data center. In: *2012 IEEE Ninth International Conference on Services Computing*. [S.l.: s.n.], 2012. p. 609–616.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn. *Queue*, ACM, New York, NY, USA, v. 11, n. 12, p. 20:20–20:40, Dec. 2013. ISSN 1542-7730. Available at: <http://doi.acm.org/10.1145/2559899.2560327>.

FU, Q.; SUN, E.; MENG, K.; LI, M.; ZHANG, Y. Deep q-learning for routing schemes in sdn based data center networks. *IEEE Access*, v. 8, p. 103491–103499, 2020.

GARCíA, S.; FERNáNDEZ, A.; LUENGO, J.; HERRERA, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, v. 180, n. 10, p. 2044 – 2064, 2010. ISSN 0020-0255. Special Issue on Intelligent Distributed Information Systems. Available at: <http://www.sciencedirect.com/science/article/pii/S0020025509005404>.

GHALANOS, A.; GHALANOS, M. A.; RCPP, L. *Package 'rugarch'*. 2017.

GOMEZ, S. E.; HERNANDEZ-CALLEJO, L.; MARTINEZ, B. C.; SANCHEZ-ESGUEVILLAS, A. J. Exploratory study on class imbalance and solutions for network traffic classification. *Neurocomputing*, v. 343, p. 100 – 119, 2019. ISSN 0925-2312. Learning in the Presence of Class Imbalance and Concept Drift. Available at: <http://www.sciencedirect.com/science/article/pii/S092523121930164X>.

GONG, W.-B.; LIU, Y.; MISRA, V.; TOWSLEY, D. Self-similarity and long range dependence on the internet: a second look at the evidence, origins and implications. *Computer Networks*, v. 48, n. 3, p. 377 – 399, 2005. ISSN 1389-1286. Long Range Dependent Traffic. Available at: <http://www.sciencedirect.com/science/article/pii/S1389128604003330>.

GONZALEZ, T. F. *Handbook of approximation algorithms and metaheuristics*. [S.l.]: CRC Press, 2007.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GOOIJER, J. G. D. *Elements of Nonlinear Time Series Analysis and Forecasting*. 1. ed. [S.l.]: Springer International Publishing, 2017.

GREENBERG, A.; HAMILTON, J. R.; JAIN, N.; KANDULA, S.; KIM, C.; LAHIRI, P.; MALTZ, D. A.; PATEL, P.; SENGUPTA, S. Vl2: A scalable and flexible data center network. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 3, p. 95–104, Mar. 2011. ISSN 0001-0782. Available at: <https://doi.org/10.1145/1897852.1897877>.

GUO, L.; MATTA, I. The war between mice and elephants. In: *Proceedings Ninth International Conference on Network Protocols. ICNP 2001*. [S.l.: s.n.], 2001. p. 180–188.

HALEPLIDIS, E.; PENTIKOUSIS, K.; DENAZIS, S.; SALIM, J. H.; MEYER, D.; KOUFOPAVLOU, O. Software-defined networking (SDN): Layers and architecture terminology. In: *RFC 7426*. [S.l.]: IRTF, 2015.

HAMDAN, M.; MOHAMMED, B.; HUMAYUN, U.; ABDELAZIZ, A.; KHAN, S.; ALI, M. A.; IMRAN, M.; MARSONO, M. N. Flow-aware elephant flow detection for software-defined networks. *IEEE Access*, v. 8, p. 72585–72597, 2020.

HE, K.; ROZNER, E.; AGARWAL, K.; FELTER, W.; CARTER, J.; AKELLA, A. Presto: Edge-based load balancing for fast datacenter networks. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. New York, NY, USA: ACM, 2015. (SIGCOMM '15), p. 465–478. ISBN 978-1-4503-3542-3. Available at: <http://doi.acm.org/10.1145/2785956.2787507>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computing*, v. 9, n. 8, p. 1735–1780, 1997. Available at: <http://arxiv.org/abs/https://doi.org/10.1162/neco.1997.9.8.1735>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. Available at: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HOLMBERG, K. *Optimization Models for Routing in Switching Networks of Clos Type with Many Stages*. 2007. Available at: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.1846>.

HONG, W.; WANG, K.; HSU, Y. H. Application-aware resource allocation for sdn-based cloud datacenters. In: *2013 International Conference on Cloud Computing and Big Data*. [S.l.: s.n.], 2013. p. 106–110.

HOPPS, C. et al. *Analysis of an equal-cost multi-path algorithm*. [S.l.], 2000.

HYNDMAN, R. J.; ATHANASOPOULOS, G. *Forecasting: principles and practice*. 2nd. ed. OTexts, 2017. Available at: <http://otexts.org/fpp2/>.

IBM. *IBM CPLEX Optimizer*. 2018. <https://www.ibm.com/analytics/cplex-optimizer>.

JAIN, S.; KUMAR, A.; MANDAL, S.; ONG, J.; POUTIEVSKI, L.; SINGH, A.; VENKATA, S.; WANDERER, J.; ZHOU, J.; ZHU, M.; ZOLLA, J.; HöLZLE, U.; STUART, S.; VAHDAT, A. B4: Experience with a globally-deployed software defined wan. In: *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. New York, NY, USA: ACM, 2013. (SIGCOMM '13), p. 3–14. ISBN 978-1-4503-2056-6. Available at: <http://doi.acm.org/10.1145/2486001.2486019>.

JINGWEN, X.; MUQING, W.; XIAOLAN, H. A traffic scheduling scheme for data center networks based on sdn. In: *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*. [S.l.: s.n.], 2019. p. 1417–1422.

JURKIEWICZ, P. Evaluation of elephant-based algorithms for flow table reduction under realistic traffic distributions. *arXiv preprint arXiv:2005.00173*, 2020.

KAMIYAMA, N.; TAKAHASHI, Y.; ISHIBASHI, K.; SHIOMOTO, K.; OTOSHI, T.; OHSITA, Y.; MURATA, M. Flow aggregation for traffic engineering. In: *2014 IEEE Global Communications Conference*. [S.l.: s.n.], 2014. p. 1936–1941.

KATRIS, C.; DASKALAKI, S. Comparing forecasting approaches for internet traffic. *Expert Systems with Applications*, v. 42, n. 21, p. 8172 – 8183, 2015. ISSN 0957-4174. Available at: <http://www.sciencedirect.com/science/article/pii/S0957417415004315>.

KATRIS, C.; DASKALAKI, S. Dynamic bandwidth allocation for video traffic using farima-based forecasting models. *Journal of Network and Systems Management*, Apr 2018. ISSN 1573-7705. Available at: <https://doi.org/10.1007/s10922-018-9456-1>.

KIM, S.; KIM, H. A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, v. 32, n. 3, p. 669 – 679, 2016. ISSN 0169-2070. Available at: <http://www.sciencedirect.com/science/article/pii/S0169207016000121>.

KLOPFENSTEIN, O. A randomized rounding heuristic to reroute tunnels in mpls networks. In: IEEE. *DRCN 2005). Proceedings. 5th International Workshop on Design of Reliable Communication Networks, 2005.* [S.l.], 2005. p. 7–pp.

KREUTZ, D.; RAMOS, F. M. V.; VERÍSSIMO, P.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-defined networking: A comprehensive survey. *CoRR*, abs/1406.0440, 2014. Available at: <http://arxiv.org/abs/1406.0440>.

KREUTZ, D.; RAMOS, F. M. V.; VERíSSIMO, P. E.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219.

KUROSE, J. F.; ROSS, K. W. Computer networking: A top-down approach . 6th. *Harlow, UK: Pearson Education Ltd*, 2012.

LANTZ, B.; HELLER, B.; HANDIGOL, N.; JEYAKUMAR, V.; O'CONNOR, B. *Mininet-an instant virtual network on your laptop (or other pc).* [S.l.]: March, 2015.

LEBIEDNIK, B.; MANGAL, A.; TIWARI, N. A survey and evaluation of data center network topologies. *CoRR*, abs/1605.01701, 2016. Available at: <http://arxiv.org/abs/1605.01701>.

LEE, T.-H.; WHITE, H.; GRANGER, C. W. Testing for neglected nonlinearity in time series models: A comparison of neural network methods and alternative tests. *Journal of Econometrics*, v. 56, n. 3, p. 269–290, 1993. ISSN 0304-4076. Available at: <https://www.sciencedirect.com/science/article/pii/030440769390122L>.

LI, H.; LU, H.; FU, X. An optimal and dynamic elephant flow scheduling for sdn-based data center networks. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 38, n. 1, p. 247–255, 2020.

LI, X.; ZHANG, L.; WANG, Z.; DONG, P. Remaining useful life prediction for lithium-ion batteries based on a hybrid model combining the long short-term memory and elman neural networks. *Journal of Energy Storage*, v. 21, p. 510–518, 2019. ISSN 2352-152X. Available at: <https://www.sciencedirect.com/science/article/pii/S2352152X1830450X>.

LI, Y.; LIU, H.; YANG, W.; HU, D.; WANG, X.; XU, W. Predicting inter-data-center network traffic using elephant flow and sublink information. *IEEE Transactions on Network and Service Management*, v. 13, n. 4, p. 782–792, Dec 2016. ISSN 1932-4537.

MANGILI, M.; MARTIGNON, F.; CAPONE, A.; MALUCELLI, F. Content-aware planning models for information-centric networking. In: *2014 IEEE Global Communications Conference*. [S.l.: s.n.], 2014. p. 1854–1860. ISSN 1930-529X.

MARVUGLIA, A.; MESSINEO, A. Using recurrent artificial neural networks to forecast household electricity consumption. *Energy Procedia*, v. 14, p. 45 – 55, 2012. ISSN 1876-6102. 2011 2nd International Conference on Advances in Energy Engineering (ICAEE). Available at: <http://www.sciencedirect.com/science/article/pii/S1876610211043116>.

MEDHI, D.; RAMASAMY, K. *Network routing: algorithms, protocols, and architectures.* [S.l.]: Morgan Kaufmann, 2017.

MEHDIZADEH, S.; FATHIAN, F.; ADAMOWSKI, J. F. Hybrid artificial intelligence-time series models for monthly streamflow modeling. *Applied Soft Computing*, v. 80, p. 873 – 887, 2019. ISSN 1568-4946. Available at: <http://www.sciencedirect.com/science/article/pii/S1568494619301723>.

MOLLAH, M. A.; YUAN, X.; PAKIN, S.; LANG, M. Rapid calculation of max-min fair rates for multi-commodity flows in fat-tree networks. *IEEE Transactions on Parallel and Distributed Systems*, v. 29, n. 1, p. 156–168, Jan 2018. ISSN 2161-9883.

MONTGOMERY, C. L. J. D. C.; KULAHCI, M. *Introduction to Time Series Analysis and Forecasting*. [S.l.]: Wiley, 2015.

MUELAS, D.; RAMOS, J.; VERGARA, J. E. L. d. Assessing the limits of mininet-based environments for network experimentation. *IEEE Network*, v. 32, n. 6, p. 168–176, November 2018. ISSN 0890-8044.

ORTIZ, J.; LONDOñO, J.; NOVILLO, F. Evaluation of performance and scalability of mininet in scenarios with large data centers. In: *2016 IEEE Ecuador Technical Chapters Meeting (ETCM)*. [S.l.: s.n.], 2016. p. 1–6.

OTOSHI, T.; OHSITA, Y.; MURATA, M.; TAKAHASHI, Y.; ISHIBASHI, K.; SHIOMOTO, K. Traffic prediction for dynamic traffic engineering. *Computer Networks*, v. 85, p. 36 – 50, 2015. ISSN 1389-1286. Available at: <http://www.sciencedirect.com/science/article/pii/S1389128615001565>.

PANIGRAHI, S.; BEHERA, H. A hybrid ets–ann model for time series forecasting. *Engineering Applications of Artificial Intelligence*, v. 66, p. 49–59, 2017. ISSN 0952-1976. Available at: <https://www.sciencedirect.com/science/article/pii/S0952197617301550>.

PENG, Y.; CHEN, K.; WANG, G.; BAI, W.; MA, Z.; GU, L. Hadoopwatch: A first step towards comprehensive traffic forecasting in cloud computing. In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. [S.l.: s.n.], 2014. p. 19–27.

PFLEEGER, S. Experimental design and analysis in software engineering. *Annals of Software Engineering*, Springer, v. 1, n. 1, p. 219–253, 1995. ISSN 1022-7091. Available at: <http://dx.doi.org/10.1007/BF02249052>.

POUPART, P.; CHEN, Z.; JAINI, P.; FUNG, F.; SUSANTO, H.; GENG, Y.; CHEN, L.; CHEN, K.; JIN, H. Online flow size prediction for improved network routing. In: *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. [S.l.: s.n.], 2016. p. 1–6.

RAMOS-PEREZ, E.; ALONSO-GONZALEZ, P. J.; NUNEZ-VELAZQUEZ, J. J. Forecasting volatility with a stacked model based on a hybridized artificial neural network. *Expert Systems with Applications*, v. 129, p. 1 – 9, 2019. ISSN 0957-4174. Available at: <http://www.sciencedirect.com/science/article/pii/S0957417419302209>.

RATHER, A. M.; AGARWAL, A.; SASTRY, V. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, v. 42, n. 6, p. 3234 – 3241, 2015. ISSN 0957-4174. Available at: <http://www.sciencedirect.com/science/article/pii/S0957417414007684>.

ROST, M.; SCHMID, S. Virtual network embedding approximations: Leveraging randomized rounding. *IEEE/ACM Transactions on Networking*, v. 27, n. 5, p. 2071–2084, 2019.

ROY, A.; ZENG, H.; BAGGA, J.; PORTER, G.; SNOEREN, A. C. Inside the social network's (datacenter) network. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. New York, NY, USA: ACM, 2015. (SIGCOMM '15), p. 123–137. ISBN 978-1-4503-3542-3. Available at: <http://doi.acm.org/10.1145/2785956.2787472>.

SABBEH, A.; AL-DUNAINAWI, Y.; AL-RAWESHIDY, H. S.; ABBOD, M. F. Performance prediction of software defined network using an artificial neural network. In: *2016 SAI Computing Conference (SAI)*. [S.l.: s.n.], 2016. p. 80–84.

Saber, M. A. S.; Ghorbani, M.; Bayati, A.; Nguyen, K.; Cheriet, M. Online data center traffic classification based on inter-flow correlations. *IEEE Access*, v. 8, p. 60401–60416, 2020.

SAK, H.; SENIOR, A.; BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: *Fifteenth annual conference of the international speech communication association*. [S.l.: s.n.], 2014. p. 123–130.

SHENG, H.; CHEN, Y. Farima with stable innovations model of great salt lake elevation time series. *Signal Processing*, v. 91, n. 3, p. 553 – 561, 2011. ISSN 0165-1684. Advances in Fractional Signals and Systems. Available at: <http://www.sciencedirect.com/science/article/pii/S016516841000040X>.

SHMUELI, G.; LICHTENDAHL, K. C. *Practical Time Series Forecasting with R: A Hands-On Guide*. [S.l.]: Axelrod Schnall Publishers, 2015. ISBN 0991576632.

SILVA, M. V. B. da; JACOBS, A. S.; PFITSCHER, R. J.; GRANVILLE, L. Z. Predicting elephant flows in internet exchange point programmable networks. In: BAROLLI, L.; TAKIZAWA, M.; XHAFA, F.; ENOKIDO, T. (Ed.). *Advanced Information Networking and Applications*. Cham: Springer International Publishing, 2020. p. 485–497. ISBN 978-3-030-15032-7.

SINGH, A.; ONG, J.; AGARWAL, A.; ANDERSON, G.; ARMISTEAD, A.; BANNON, R.; BOVING, S.; DESAI, G.; FELDERMAN, B.; GERMANO, P.; KANAGALA, A.; PROVOST, J.; SIMMONS, J.; TANDA, E.; WANDERER, J.; HöLZLE, U.; STUART, S.; VAHDAT, A. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. [s.n.], 2015. (SIGCOMM '15), p. 183–197. ISBN 978-1-4503-3542-3. Available at: <http://doi.acm.org/10.1145/2785956.2787508>.

SINGLA, A.; HONG, C.-Y.; POPA, L.; GODFREY, P. B. Jellyfish: Networking data centers randomly. In: *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX Association, 2012. p. 225–238. ISBN 978-931971-92-8. Available at: <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/singla>.

THALER, D.; HOPPS, C. *Internet Control Message Protocol*. RFC Editor, 2000. RFC 2991. (Request for Comments, 2991). Available at: <https://rfc-editor.org/rfc/rfc792.txt>.

TONG, X.; WANG, Z.; YU, H. A research using hybrid rbf/elman neural networks for intrusion detection system secure model. *Computer Physics Communications*, v. 180, n. 10, p. 1795 – 1801, 2009. ISSN 0010-4655. Available at: <http://www.sciencedirect.com/science/article/pii/S0010465509001519>.

TRIVEDI, K. S. *Probability & Statistics with Reliability, Queuing and Computer Science Applications*. [S.l.]: PHI Learning Pvt. Limited, 2011.

TSO, F. P.; PEZAROS, D. P. Improving data center network utilization using near-optimal traffic engineering. *IEEE Transactions on Parallel and Distributed Systems*, v. 24, n. 6, p. 1139–1148, 2013.

VAHDAT, A.; AL-FARES, M.; FARRINGTON, N.; MYSORE, R. N.; PORTER, G.; RADHAKRISHNAN, S. Scale-out networking in the data center. *IEEE Micro*, v. 30, n. 4, p. 29–41, July 2010. ISSN 0272-1732.

VAIDYA, P. M. A new algorithm for minimizing convex functions over convex sets. In: IEEE COMPUTER SOCIETY. *30th Annual Symposium on Foundations of Computer Science*. [S.l.], 1989. p. 338–343.

VARMA, S. Chapter 7 - congestion control in data center networks. In: VARMA, S. (Ed.). *Internet Congestion Control*. Boston: Morgan Kaufmann, 2015. p. 205 – 230. ISBN 978-0-12-803583-2. Available at: <http://www.sciencedirect.com/science/article/pii/B9780128035832000074>.

VIEIRA, F. H. T.; COSTA, V. H. T.; GONÇALVES, B. H. P. Neural network based approaches for network traffic prediction. In: *Artificial Intelligence, Evolutionary Computing and Metaheuristics*. [S.l.]: Springer, 2013. p. 657–684.

WALDMAN, H.; JR, F. L. T.; RIBEIRO, M. R. Appropriate teletraffic framework in ip over wdm. In: *XXII Simpósio Brasileiro de Redes de Computadores (SBRC 2004), pág*. [S.l.: s.n.], 2004. p. 647–660.

WALRAED-SULLIVAN, M.; VAHDAT, A.; MARZULLO, K. Aspen trees: Balancing data center fault tolerance, scalability and cost. In: *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*. New York, NY, USA: Association for Computing Machinery, 2013. (CoNEXT '13), p. 85–96. ISBN 9781450321013. Available at: <https://doi.org/10.1145/2535372.2535383>.

WANG, J.; WANG, J.; FANG, W.; NIU, H. Financial time series prediction using elman recurrent random neural networks. *Computational intelligence and neuroscience*, Hindawi, v. 2016, 2016.

WANG, P.; TRIMPONIAS, G.; XU, H.; GENG, Y. Luopan: Sampling-based load balancing in data center networks. *IEEE Transactions on Parallel and Distributed Systems*, v. 30, n. 1, p. 133–145, Jan 2019. ISSN 1045-9219.

WANG, W.; SUN, Y.; SALAMATIAN, K.; LI, Z. Adaptive path isolation for elephant and mice flows by exploiting path diversity in datacenters. *IEEE Transactions on Network and Service Management*, v. 13, n. 1, p. 5–18, 2016.

WANG, X.; ZHANG, C.; ZHANG, S. Modified elman neural network and its application to network traffic prediction. In: *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*. [S.l.: s.n.], 2012. v. 02, p. 629–633. ISSN 2376-5933.

WANG, Y.; WANG, X.; LI, H.; DONG, Y.; LIU, Q.; SHI, X. A multi-service differentiation traffic management strategy in sdn cloud data center. *Computer Networks*, v. 171, p. 107143, 2020. ISSN 1389-1286. Available at: <http://www.sciencedirect.com/science/article/pii/S1389128619309740>.

WAZLAWICK, R. S. *Metodologia de Pesquisa para Ciência da Computação*. [S.l.: s.n.], 2009. ISBN 9788535234107.

WEI, W. W. S. *Time Series Analysis, Univariate and Multivariate Methods*. 2. ed. [S.l.]: Pearson Addison Wesley, 2006.

WILLIAMSON, D. P.; SHMOYS, D. B. *The design of approximation algorithms*. [S.l.]: Cambridge university press, 2011.

XIA, W.; ZHAO, P.; WEN, Y.; XIE, H. A survey on data center networking (dcn): Infrastructure and operations. *IEEE Communications Surveys Tutorials*, v. 19, n. 1, p. 640–656, Firstquarter 2017. ISSN 1553-877X.

XIE, K.; TIAN, J.; WANG, X.; XIE, G.; WEN, J.; ZHANG, D. Efficiently inferring top-k elephant flows based on discrete tensor completion. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. [S.l.: s.n.], 2019. p. 2170–2178.

XU, H.; LI, X.; HUANG, L.; DENG, H.; HUANG, H.; WANG, H. Incremental deployment and throughput maximization routing for a hybrid sdn. *IEEE/ACM Transactions on Networking*, v. 25, n. 3, p. 1861–1875, 2017.

YAP, B. W.; SIM, C. H. Comparisons of various types of normality tests. *Journal of Statistical Computation and Simulation*, v. 81, n. 12, p. 2141–2155, 2011. Available at: <http://dx.doi.org/10.1080/00949655.2010.520163>.

YOON, M. S.; KAMAL, A. E.; ZHU, Z. Adaptive data center activation with user request prediction. *Computer Networks*, p. –, 2017. ISSN 1389-1286. Available at: <http://www.sciencedirect.com/science/article/pii/S1389128617301780>.

ZENG, J. H.; SHAO, M. *Facebook Network Analytics Data Sharing*. 2016. Available at: <https://www.facebook.com/groups/1144031739005495/about>.

ZHAN, Z.; XU, M.; XU, S. Predicting cyber attack rates with extreme values. *IEEE Transactions on Information Forensics and Security*, v. 10, n. 8, p. 1666–1677, Aug 2015. ISSN 1556-6013.

ZHANG, J.; YE, M.; GUO, Z.; YEN, C.; CHAO, H. J. Cfr-rl: Traffic engineering with reinforcement learning in sdn. *IEEE Journal on Selected Areas in Communications*, p. 1–1, 2020.

ZHANG, J.; YU, F. R.; WANG, S.; HUANG, T.; LIU, Z.; LIU, Y. Load balancing in data center networks: A survey. *IEEE Communications Surveys Tutorials*, v. 20, n. 3, p. 2324–2352, 2018.

ZHANG, Y.; CUI, L.; ZHANG, Y. A stable matching based elephant flow scheduling algorithm in data center networks. *"Computer Networks"*, v. 120, p. 186 – 197, 2017. ISSN 1389-1286. Available at: <http://www.sciencedirect.com/science/article/pii/S1389128617301603>.

ZHANG, Y.; WANG, X.; TANG, H. An improved elman neural network with piecewise weighted gradient for time series prediction. *Neurocomputing*, v. 359, p. 199–208, 2019. ISSN 0925-2312. Available at: <https://www.sciencedirect.com/science/article/pii/S0925231219308215>.

ZHAOWEI, Q.; HAITAO, L.; ZHIHUI, L.; TAO, Z. Short-term traffic flow forecasting method with m-b-lstm hybrid network. *IEEE Transactions on Intelligent Transportation Systems*, p. 1–11, 2020.