Dynamic Allocation of Resources Using Machine Learning and Quantile Regression by Harnessing the Power of Software Defined Networks

by

Ahmed I. Alutaibi
B.Sc., King Fahd University for petroleum and Minerals, 2001
M.Sc., University of Dalhousie, 2010

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Dynamic Allocation of Resources Using Machine Learning and Quantile Regression
by Harnessing the Power of Software Defined Networks

by

Ahmed I. Alutaibi
B.Sc., King Fahd University for petroleum and Minerals, 2001
M.Sc., University of Dalhousie, 2010

Supervisory Committee

Dr. Sudhakar Ganti, Supervisor
(Department of Computer Science)

Dr. Dr. Alex Thomo, Departmental Member
(Department of Computer Science)

Dr. Xiaodai Dong, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

In the last decade, data networks have shifted from the static deployment of resources to a dynamic approach. With the help of Software Defined Networks (SDN) and Network Function Visualization (NFV), information and data about the network can be collected. Also, deployment and allocation of resources can be delegated to a central controller. In this thesis we investigate the power of SDN and how central management of resources can help produce better and efficient data networks. It begins with an introduction to SDN and its capabilities. The added benefits of SDN over traditional network frameworks and topics that SDN contributed most to. We show the power of collecting data using SDN and how it enables different approaches to accomplish the needed task. This was facilitated by the programmability and the separation of the control and data planes. We tackle the simple task of measuring the delay between two communicating devices in the network. The results show that SDN is capable of providing a rich infrastructure to build future networks. Also, it illustrates that using SDN to measure the delay between devices in the network can give accurate results. The differences between the tested techniques is shown and evaluated. After collecting the data from the network, the next step is getting an insight on that data. Next we used collected network bandwidth data to predict future bandwidth usage. We used various prediction models to establish prediction intervals. We created a state of the art metric that evaluates and compares the performance of each model. We show that the network bandwidth is highly predictable and that dynamic allocation of network bandwidth is attainable. The next logical step is to act upon those insight which is investigated next. We establish the same prediction models investigated but instead of prediction intervals we establish upper quantiles. Prediction is done on data center resources data set. The results show that using quantile prediction can give guarantees on resources usage boundaries which implies a guarantee on service level agreements. Allocating just the needed resources, produce a more efficient data center and in turn cuts a lot of the needed energy. Our estimate show that upto 56% of power can be saved without violating the service level agreement.

SDN	Software Defined Networks
NFV	Network Function Visualization
SNMP	Simple Network Management Protocol
RTP	Real-time Transport Protocol
IP	Internet Protocol
TCP	Transport Control Protocol
QoS	Quality of Service
ATM	Asynchronous Transfer Mode
RFC	Request for Comments
API	Application Programming Interface
SLA	Service Level Agreement
OSI	Open Systems Interconnection Model
ONF	Open Networking Foundation
ISP	Internet Service Providers
MAC	Media Access Control
RTT	Round Trip Time
ICMP	Internet Control Message Protocol
ARIMA	Autoregressive integrated moving average
LQR	Linear Quantile Regression
OLS	Ordinary Least Squares
RF	Random Forests
GTB	Gradient Tree Boosting
FNN	Feedforward Neural Network
LSTM	Long Short-Term Memory
MSE	Mean Square Error
MAE	Mean Absolute Error
ML	Machine Learning

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Ganti, for believing in me and pushing me to achieve my goals. He was the rock that we leaned on when we needed support and the father that showed us the way.

My colleagues, for the help and support

Saudi Arabian bureau and the University of Majm'ah, for funding me with a Scholarship and the moral support when I needed it.

DEDICATION

This work is dedicated to my wife. Without her support and love i wouldn't be here. And to my mother who always saw me writing this since i was little.

Chapter 1

Introduction

In the last few decades, the internet evolved from a network exclusive to government and university usage into a worldwide fabric that connects an ever increasing number of user devices. The number of internet users are growing every day and it is estimated that 47% of the world population is using it today [1]. Most people that use the internet these days consider it as important as other public utilities [2]. Finding information these days has become a very easy and accessible task compared to searching for it in a library 30 years ago. It is easy to see how the internet became a platform for communication, information, co-operate work, and content sharing. But it is hard to imagine how it all came to be. The internet evolved by solving problems as they arise not by foresight and good design. It was designed to do basic communication with limited resources at its inception. Also, it wasn't designed for management, privacy, mobility, parallelism, or with security in mind. Over the years, those problems were fixed by adding plug-ins to the main protocols or inventing other protocols for special tasks. For example, Simple Network Management Protocol (SNMP) [3] was adapted to help in the management and monitoring aspects of the network. Also, Real-time Transport Protocol (RTP) [RFC 1889] was introduced to facilitate real time voice and video over IP [RFC 791]. Many more examples exist in today's networks which show how the internet and networks in general are built on weak foundation.

The evolution of the internet and networks has been driven by the demand for higher bandwidth nothing more. Management, Monitoring, Quality of Service (QoS), Security, Privacy, and many others were added or thrown over to the application side. This left holes, vulnerabilities, and inconsistencies in the network. If we were to redesign a new protocol for the internet we would design a protocol that will take

all of those aspects into consideration. Many attempts have been made through the years to bring better network design into protocols and architecture of the internet. Most of those ideas were never adopted because they implied an expensive paradigm change for the already good enough existing networks. For example, ATM [RFC 4454] (the Cadillac of networks as my supervisor calls it) has been designed with all the past experiences, problems, and innovations in mind. Although it needed a decade to die off, ATM still reminds us that not all good ideas are easy to implement or are feasible to adapt. Failure of ATM is credited to the cost of adoption. As a matter of fact, some implementations avoided the cost of replacing the current infrastructure by putting ATM over IP, which was inefficient and drove the cost of WAN links higher. So IP won because it was cheaper, familiar, and easy to upgrade. The difficulties in adopting new protocols can be summarized into the following:

1. One of the hurdles that faced the adaption of new protocols was and is organizational acceptance. The standardization life cycle takes very long which discourage innovation. When a good idea or a contribution is proposed to the Internet Engineering Task Force (IETF), it is first published as an Internet Draft then it is promoted to an RFC (Request for Comments). From this it might be promoted to a standard. This work flow can take years. The process can be accelerated if the proposal is backed up by manufacturers or scientific organizations.

2. Device manufacturers try to incorporate some of the features that they think are good additions to the standard protocols. This only widens the gap between them and makes the adaption of those protocols diminish. Making those features a standard is the best way to adopt it by the networking community which brings us to the first hurdle.

3. Upgrade costs as mentioned above. The adapted solutions should be low cost and easy to migrate to. It is enough these days to change an interface to upgrade a link from 1Gbps to 10Gbps.

4. Experimenting with new protocols in the research labs is very hard since the vendor devices are closed boxes. Manufactures have always been resistance to the programmability of their devices. They provide interfaces to configure them through remote connections or web interfaces but never allowing APIs

(Applications Programming Interface). This made it harder for researchers to test their ideas and made an obstacle for new innovations to come to light.

It is convenient to think that a new protocol that overcomes those hurdles and incorporate features (like management, monitoring, security, QoS, etc) will find its way into today's networks. Software Defined Networks (SDN) was introduced to answer and solve most of the concerns through a unified platform. It has a centralized view of the network which allows it to control and manage network devices. The power is attributed to the programmability of the control plane enabling the application to control how the network behaves.

In this thesis, we are interested in coupling this programmability with network and resources dynamic behavior. The next generation of networks and Data Centers will allocate resources depending on demand not on reservation. This notion inspired us to look for tools and investigate how we can allocate enough resources to strike a balance between efficient allocation and satisfactory services.

1.1 Overview of Chapter 2

We will begin with the description of SDN, its history and how the separation of the control plane from the data plane is a powerful idea to manage and administer resources. In this chapter we are going to introduce the architecture of SDN and its main components. The widely accepted implementation of SDN Openflow will be described in section 2.1 and its implementation will be laid out in 2.2. The Advantages of this new design over the legacy network architecture are vast. We are going to emphasize some of those in section 2.3. The next section is dedicated to issues that the research community has tackled and the open research areas in this field.

We present this chapter to give an overview of SDN and how it enables the creation of new protocols, tools, and control of the network. This chapter will pave the way for the next chapters which in some sense use SDN and its controlled programmable configurable behavior.

Chapter two contribution:

- Overview of SDN.
- Showing the power of separating the control and data plane.

- Discussing the solutions and research areas that use this new paradigm.

1.2 Overview of Chapter 3

We take lessons learned in the previous chapter to use SDN for the improvement of QoS. To facilitate better QoS for applications, the network must calculate and measure certain parameters. One of those parameters is the delay between the end-points of the connection. Measuring latency between a server and client is easy when its measured from those end-points. But measuring it from the perspective of the service provider or the network device in the middle is harder. These need to measure latency for a number of reasons like QoS, routing, security, or privacy.

Using the power of SDN and the knowledge of application and network behavior, SDN networks can adjust to different conditions and specifications. In this chapter, we investigate how delay measurement can steer the process of providing better QoS. In addition, we discuss a number of algorithms to measure delay and their implementations in an SDN environment. Limitations and suggestions to Openflow are addressed at the end of this chapter.

Chapter three contribution:

- Investigating the importance of measuring delay in the network.
- Building state of the art techniques that do not use any admin privileges on the measured devices, contrary to previous works.
- Evaluating the trade-offs between the techniques adopted and how accuarte each one of them.
- Show how Openflow shortcomings has affected the implementation of some methods and outline additions and suggestions to it to solve these problems.

1.3 Overview of Chapter 4

In this chapter we take the data provided by the network a step further. Taking the information collected to get an insight on the behavior of daily usage bandwidth. We use machine learning to predict how much bandwidth is going to be used in the next future time frame.

Machine Learning research has progressed tremendously in recent years. Major fields that machine learning pushed its frontier was prediction and data modeling. We will evaluate the applicability of a handpicked prediction models on predicting inter-day aggregate network traffic. We chose models that work best with multi-variate feature space. They represent linear, decision trees, and neural network models. Over the years, predicting network traffic has resorted to predicting point values. This approach is not descriptive enough and naively gives a shallow conclusion about the data. We propose using a quantile loss function that predicts boundaries or prediction intervals. Our results show that linear models fared well compared to their simplicity while Long Short-Term Memory Neural Networks gave best results across all experiments.

When allocating link bandwidth users, administrators, or service providers tend to over allocate. Most of that bandwidth is not used. The results of this chapter can be used to provision the needed bandwidth without over allocating it. This can help in cutting costs and/or introducing a dynamic behavior to fulfill the minimum requirements of the service.

Chapter four contribution:

- Highlighting the predictability of network traffic.
- Evaluating bounded prediction instead of point prediction.
- Comparing six prediction models.
- Establishing state of the art metric that finds the best model. This metric combines the prediction loss and the bound interval capture percentage.

1.4 Overview of Chapter 5

Inspired by the previous work in chapter 4, we use the models to test the predictability of Data Center resource usage. Data Centers have been growing in size and demand continuously in the last two decades. Planning for the deployment of resources has been shallow and always resorted to over-provisioning. Data Center operators try to maximize the availability of their services by allocating multiples of the needed resources. One resource that has been wasted, with little thought, has been energy. In recent years, the adoption of Network Functions Virtualization (NFV) has paved the way to allow for dynamic deployment of resources. This and the programmability

of the infrastructure have lead to more efficient deployments. SDN's ability to collect data and then combine NFV and programability promises a powerful framework.

In this chapter we examine the predictability of resource usage in a Data Center environment. We use the best preforming models from chapter 4 to predict the future time frame usage of some resources. Then we establish a framework to guarantee client Service Level Agreement (SLA). Our results show that using prediction can cut energy loss by up to 55%.

Chapter five contribution:

- Proving that dynamic networks and Data Centers are superior to static ones.
- Showing that guarantees on predicted quantiles can be translated to guarantees on service level agreement.
- The differences between the models prediction was not that significant. What we found significant was the importance of using prediction in general to allocate resources.
- Some resources were predictable more than others.
- Building a Data Center with prediction and conservation of energy in mind, we could see a cut to energy and cost.

The overall work pushes the idea of active and adaptable infrastructure where we use the tools provided by SDN. We wanted to show how we can use it to measure and collect data inside the network. Then use that information to make changes that improve the efficiency, cost, and management without negatively impacting the services provided.

Chapter 2

Software Defined Networks (SDN)

This chapter provides a brief overview of Software Defined Networks and their functionality. In networks, the decision process on how and where to handle data have always been done in a distributed manner. Distributed in the sense that each device has the needed logic to receive the data (packet) and through some information embedded in the header decides which output interface it should send it through. As illustrated in figure 2.1 both the logic (control plane) and the forwarding (data plane) are on each device. This approach is done on most layers of the OSI model. If a network administrator needs to manage traffic in the transport layer they need to install a device in that part of the network (e.g. firewall). This can be costly and hard to configure. The distributed nature of this approach causes the network to be harder to manage, monitor, secure, and troubleshoot. Also, many inconsistencies arise because of the dependency of those devices on their neighbors for topology information.

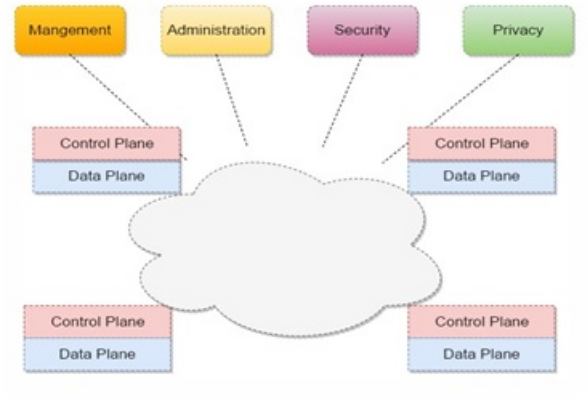


Figure 2.1: Distributed network control.

The research community has been experimenting with the classical architecture of protocols and networks for many years to bring them closer to a programmable centralized platform. In the mid 1990s Active Networks were proposed where the control plane was exposed. This helped the programmability and active management of the resources [4]. In late 1990's proposals were aiming to [5, 6, 7]:

- Allow programmability of the control plane.
- Network wide view.
- Allowing the experimentation with new ideas.

Centralized management of the network had been suggested in many papers in the mid 2000s to give a general view of the network [8, 9, 10, 11, 12]. Those efforts set the stage for the implementation of routing in software instead of hardware. In addition, they paved the road for separating the control from the data plane. Those ideas were faced with a lot of criticism which attacked the distributed nature of the solutions. One of the major concerns was that inconsistencies would be introduced if the implementation has more than one traffic controller but that was easily fixed with establishing a communication channel between them [12].

Even though the community was a bit skeptical of the centralized architecture, research in that area didn't stop and in 2008 the paper on Openflow [13] was published. It opened the door for the standardization of Openflow and realization of SDN in general. The team published at the same time their implementation of an Openflow controller which they called NOX [14]. Support from the equipments vendors, ISPs, and universities made the standardization of the new protocol a reality.

Openflow is an implementation of SDN but is not the only implementation. It is driven by the Open Networking Foundation (ONF). In the next sections we will discuss SDN in general and the implementation of Openflow in particular. Also, how Openflow translates the SDN ideas into a protocol.

2.1 SDN and Openflow Emergence

In the past ten years, the network community witnessed the emergence of a new approach to networking called Software Defined Networks (SDN). It revolves around the idea of separating the control plane from the data plane. This is implemented by

taking the control functions from the switches (and routers) and implementing them in a separate software module sub-system. The forwarding functionality of hardware will be left unchanged. This separation of control plane and data plane, promises lots of powerful features that can push network and Data Center innovation forward. From figure 2.2 we can see how SDN separates the control and data planes (compared to figure 2.1) and how a centralized controller has all the needed functionality to manage the network. The centralized view of the network enables the management application and administrators to control the flow of data more efficiently. Also, it enables collection of data from the network devices and servers. Quality of service can be implemented with ease. Also, security and privacy can be enforced on network wide policies. Depending on the state of the network, service demands, and application condition the network/services can adopt to dynamic changes. This can be accomplished by sending commands to network devices/servers.

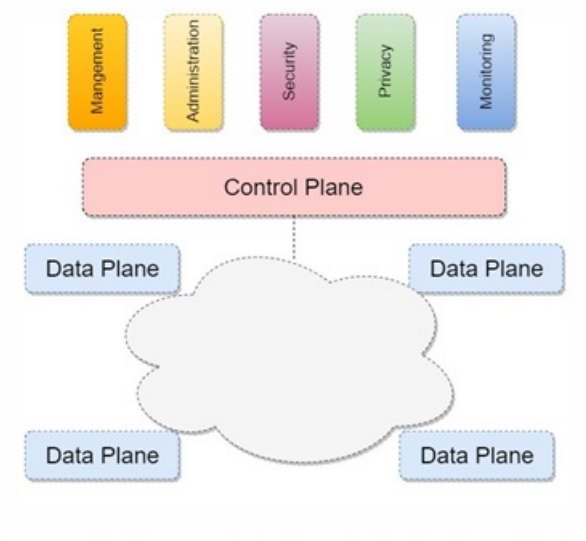


Figure 2.2: SDN architecture.

Openflow [13] has been the main protocol realization of SDN. It has been standardized in 2011 and is driven by the ONF (Open Networking Foundation). The protocol is supported by all the major network device manufacturers. The Openflow functionality mandates small changes at the switch logic to accept, store, and handle Openflow functionality. In this context, devices that don't support SDN are called legacy devices and those that support the protocol are called Openflow enabled devices. Device vendors these days manufacture their devices in one of three types of

operation. Some devices support hybrid operation where they can work either legacy or Openflow. This allowed a smooth transition to Openflow but left the burden of the adaptation of new the paradigm on network administrator shoulders.

2.2 Openflow Implementation

A network flow is defined by the fields in its header. Those fields can be destination, source IP, port number, MAC address, or any characteristics that distinguish it from other data streams. Flow tables in legacy switches store information on which other devices are connected to its physical ports. When a switch receives a packet, it sends an ARP (Address Resolution Protocol) broadcast request to all its neighbors asking if any of them has the destination address of this packet. When it receives the reply, it sends the packet through the port it received it from and stores this information in its forwarding table. In Openflow, the table architecture is extended to include three major abstract types of information which are:

In physical port	Source MAC Address	Destination MAC Address	Ethernet Type	VLAN ID	IP TOS	IP Protocol	IP Source	IP Destination	TCP Source port	TCP Destination port	MPLS Labels

Figure 2.3: Matching fields.

- Matching rules: The received packet is matched against header fields. Priorities can be specified. The matching can be on any or a subset of the header fields shown in figure 2.3 [13]. The process of matching can be specified completely or can be wildcarded. Using wildcards in the matching process can help matching multiple flows with one entry in the table.

- Counters: When packets arrive at the switch and a match is found, the counters field is incremented. The counters field helps in monitoring and control of flows passed though the switch. Those counters can be received or sent packets, received or sent bytes, drops, errors, and many more.

- Action: This field is where the switch gets instructions on how to deal with flows matched. Those actions can be one of the following:

- Output: pointing to which output port to send the flow, either physical or virtual ports can be used.
- Set queue: This action puts the flow into a queue for QoS purposes.
- Drop: to discard a packet matched using this table entry.
- Push and pop tag: used to work with VLANs and MPLS flows.
- Set field: This action is used when changes to certain fields are desired in the header.

At the start, those tables are empty. When an Openflow switch receives a new packet, it sends that packet's header to the controller. The controller through the logic and algorithm running on it decides which action to take for that flow (e.g. output it on port number 3 or drop packet). This process consists of two control massages between the switch and the controller. The first control message is called packet-in and it is sent asynchronously to the controller. The reply is a synchronous packet-out message. The Openflow protocol consists of many more messages between the controller and the switch. For example, there are Hello, Echo, Feature request, get and set Config, port and switch stat, and many more. The Openflow protocol (OFP1.4) has 34 control messages.

Another powerful feature of Openflow is making a separate table for groups of flows. Group tables are used to address and treat broadcast, multicast, and anycast easily. Those tables have their own parameters but are very similar to normal flows tables.

When flows are matched, they are put into a matching pipeline that takes it from one table to another. Starting from the flow entry with the highest priority, the packet is matched and the action is applied (if the matching process is positive). Then it is matched against the next flow entry and so on until there are no more entries to match. This pipeline of match and action chain is powerful in applying multiple rules on the same flow or covering a range of flows.

The communication between the controller and the switch is secured by the TLS transport protocol. Each switch and controller pairs should authenticate the other by exchanging secure certificates. Although the control channel is secured, SDN in general has its own security issues.

The centralized architecture of Openflow makes it easy to program the controller with algorithms that not only find the shortest path but find the best path with the

current resources. This is possible because the controller have the complete topology information of the network and has the monitoring capabilities to get the current link state. To avoid programming each functionality into the controller, most of the major implementations of the controller are using Applications Programmable Interfaces (API). This is an added layer where applications can configure and query the controller.

2.3 Advantages of SDN

The advantages of SDN over classical networks can be summarized as follows:

- Programmability: The whole idea of bringing the control to software was proposed to enable program controlled networks. This coupled with complete network information, guarantees that decisions are made with the needed knowledge and full control.

- Innovation: As mentioned in previous sections, the research community had a lot of difficulties in testing their new ideas because of the closed nature of vendor devices. With SDN, the devices are used only for forwarding while the controller is programmed separately. This opened up the door for innovation. Now it is easy to build a simple program to take over the network control or even send a simple API REST command to change the behavior of the network. Also, it enabled researchers and technology companies to experiment with new ideas and design the network to serve special needs. For example, Google implemented their SDN version [15] to better control and monitor their WAN network and to cut cost on those links. This brings us to the next advantage of SDN.

- Cost: Since SDN removes the control logic from switches, it simplifies them. Also, it allows manufacturers to design those devices to be cheaper and target forwarding plane improvements only. Some estimates suggest that cost cuts can be as high as 50% [16]. The cost reduction is a result of using cheap middle ware devices. Also, usually administrator overprovision links to avoid congestion problems which results in utilizations of 30%-40% (our main focus in chapter 4). With SDN the utilization can be driven up to near 100% [15] which cuts down on the number of links need or the bandwidth capacity.

- Efficiency: The overall view the SDN provides to the end application establishes a powerful tool to manage the network with good knowledge of its state. When the controller is consulted about a new flow, it checks the current flows it has already installed in the switch and from that it can decide on which path to take. As a result, the controller logic can build a network with more tolerance (e.g. load balancing), robustness, and security.

- Security and Privacy: In classical networks, if an administrator wants to implement security they need to install a device between two network segments. Doing this makes the two segments isolated and traffic flow between them only if the administrator allows it. In SDN, rules can be implemented in any of the SDN enabled devices since flows can be allowed or dropped. It is easy to see that every device in an SDN network can be treated as a forwarding and security device. This also enables a more in depth and detailed policy enforcement of the intended security model. Another aspect is separation of traffic where different users can have different views of the network and its services.

- Virtualization: In the past VLAN and MPLS tried to introduce the concept of multi-tenets networks but it had been a tough task to automate. With SDN it is easy to create multiple virtual networks on top of one physical network. To take it even a step further one physical network can have multiple controllers on the same network. Flowvisor [17] is a project that facilitate splitting the network into multiple network slices. It resides between the controllers and the network devices and enables each controller to have their own view of the network.

- Quality of Service: having a more general view of the network and the ability to provision resources are key aspects in making QoS feasible in SDN. Most of QoS implementations in classical networks are either done per device or by end to end reservation. Both of those techniques have their weaknesses that don't guarantee overall network efficiency. However, in SDN, the full information on network resources can be abstracted to find best path to take where parameters of the application, links, and devices can be taken into consideration.

- Management and Monitoring: Network administrators have been using various tools for management and monitoring. Most of them were expensive and hard to learn. Even devices from the same vendor can differ on how they are configured. Also, monitoring needed to enable certain protocols that can take more

device resources. With SDN, The protocol is designed with generalized view and control which makes the management process easier and have a broader look. In addition, the counters field incorporated in the flow tables can give abstracted monitoring parameters. The next chapter will investigate how can an SDN network provide a powerful tool enabling to measure the latency between communicating parties.

2.4 Open Research Areas

Although SDN research in general is a broad subject, we need to mention some of the great efforts in various areas of research. This will show the power of SDN and will allow us to refer to those works more easily. We can tackle the research problems and how the research community contributed to solve them through the following breakdown of the areas of interest.

2.4.1 Network Programming and Network OS

Since the beginning of networks, researchers have been trying to suggest and facilitate a networking platform that resembles computers. It is easy to see that the separation of software and hardware in computers allowed them to be OS independent. Every user can load up and install their preferred operating system. The same is envisioned in networks. Researchers are constructing the building blocks to enable a network operating system that will run all the needed features on any network topology or technology. To accomplish this network programming languages should be designed to satisfy the separation of low level hardware and the high level network wide policies.

Pyretic [18] was introduced as a language designed to abstract the network view and functions. The network programmers don't need to know the details of the underlying network architecture; instead they implement policies using simple network wide high level policies. Also Pyretic enables programmers to write policies in a modular fashion, in the sense that they can write multiple functions without worrying that they will interfere with each other. Since monitoring is an essential part of a good network programming decision process, Pyretic has incorporated querying the network for stats and parameters. Finally the language has an abstract view of the network topology which allows for a more complete control over the network. The Pyretic project was extended and absorbed by the Frenetic [19] project. Frenetic is a

high level network programming language that has the following goals:

- Abstraction of the low level architecture and enabling the control by high level network wide instructions.
- Platform independence.
- Allowing modular programming with emphasis on consistency and functionality.
- Building the language in such a way that it complies with the standard models known in present day programming languages.
- A comprehensive documentation on the language to enable wider community participation and contribution.

Frenetic implementation consists of three main functions

- Network query: The language handles the low level communication with network devices to get counters, stats, and aggregate summarization.
- Network wide policies.
- Network configuration: The language shields the user from the low level details of implementation of configuration and allows them a more general control.

Using the idea of Functional Reactive Programming, Procera [20] was designed as a policy declarative language. Depending on network conditions the language can react to changes and events in the network. It uses a declarative and compositional architecture to achieve its goals. Using Procera, uCap [21] was implemented to deliver strong tools to home networks where for example the policies can be declared to limit bandwidth used for each device.

2.4.2 Virtualization

The virtualization paradigm has been driving cloud environment for two decades. In the same manner, network researchers see that applying the same ideas in networks will result in similar benefits. Device manufacturer provide the infrastructure needed while the software is independent of those devices. In addition, seamlessly the network provides multiple virtual networks that operate without interfering with each others. To achieve this goal, a lot of research has been done in this area. The Open vSwitch

[22] project is a software implementation of an Openflow switch. It can support connecting virtual and physical interfaces and multiple instances can run on the same or different machines. This allows for the implementation of networks within software.

Allowing multiple controllers on the same physical network is feasible but runs into a number of problems. Security is one of them, in the sense that each controller has full access to the flow table of all switches. Also, each controller's view of the network may not be correct since it doesn't take into account the presence of other controllers in the network. For this and many other reasons, the Flowvisor [19] project was introduced. It resides in the southbound layer of the Openflow network. Flowvisor creates a slice of network to each of the controllers and shields each controller's information from the others. Also, it provides a correct view of the network resources regardless of the number of controllers in the network.

The OpenStack-Opendaylight [23] project is integrating an SDN controller (Opendaylight [24]) with an Open infrastructure facilitator (OpenStack [25]). Through the Neutron [26] module of the OpenStack, users can build a network with switches, routers, and hosts with a click of a button. All done in software and the created network can be used with other OpenStack modules (e.g. install different Operating Systems or services).

2.4.3 Debugging

Since networks are growing ever closer to softwarization, it is easier to automate the process of correcting mistakes and removing inconsistencies. VeriFlow [27] is proposed as a tool to verify rules sent to switches to detect network wide inconsistencies. OFRewind [28] proposes a method to record events and communication between the controller and the managed device and play them back for troubleshooting proposes. In FlowChecker [29] and ConfigChecker [30], Binary Decision Diagrams are used to check invariants manually or automatically. These approaches take into consideration the presence of multiple controllers and different Openflow implementations.

NICE [31] is a tool that has the ability to check for bugs and report them to the programmer. It uses the notion of a network model and finding loops, black holes, and application faults. The tool ndp proposed in [32] is a debugging tool used to insert breakpoints and back tracing into the Openflow functionality.

2.4.4 Monitoring

Since Openflow had already incorporated modules, parameters, and functions to implement good monitoring tools, the task of designing monitoring applications was easy. Most research has been targeting the creation of a monitoring framework. In OpenSketch [33] the researchers proposed separating the monitoring plane from the control by the use of hashing, filtering, and counting to get the needed data. Another framework for monitoring is PayLess [34] which is an efficient tool to collect statistics from the network. In addition, it provides a REST API making it easier for users to query for monitoring information. PayLess minimizes the number of probes done between the controller and the devices by using an adaptive probing algorithm.

Trying to archive the same goals of low overhead data collection, OpenTM [35] was designed to bring the topology knowledge with probing to gather accurate monitoring information. In this project the developers used routing information in the controller to know the topology and flows installed then propping one switch in the path to figure out the total bandwidth used by that flow. In doing this technique OpenTM cuts down on the number of switches probed and enough accurate information is gathered to build a monitoring module. Also, in OpenSample [34] the proposed idea is to use sFlow to sample traffic parameters periodically and use TCP sequence number to imply changes in traffic. As a result, the information collected can give statistics of the network with very low overhead.

2.4.5 Security

It is clear that SDN can implement the security rules very easily by implementing them in the controller. It is enough to block certain IP addresses or port access on the network or a portion of the network. So Firewalls and IDSs can be all implemented on the controller. In addition, the security issues that exist in legacy networks still persist on SDN networks. Saying that, SDN provides powerful tools and views of the network allowing innovation in security generally.

Pedigree [36] built a security model that checks the validity of host connecting to the network before allowing flow from them. It works on identifying authorized processes on the hosts and blocking non-authorized processes using a tagging technique. In FRESCO [37] the researchers built a security framework that allows the creation of high level policies. The framework includes a scripting language and an API layer to interact with the framework. Since SDN allows for easier implementation of anomaly

detection algorithms, the researchers in [38] used four anomaly detection algorithms used in legacy networks. They proved that using those algorithms is effectively easier in SDN.

The idea of SDN networks comes with its own security problems. They can be summarized in the following points:

- Controller or Switch vulnerabilities.
- Control channel vulnerabilities.
- Trust between devices and faking of traffic flows.

In Kandoo [39] the proposed architecture introduced the idea of hierarchal multi-controller network. This approach invested in the idea of separating the responsibilities of each network segment to different controllers. In addition, higher controllers in the hierarchy have a more general view of the network. The design proposal helps the network isolate and detect certain kinds of attacks. AVANT-GUARD [40] is a security system built to combat the vulnerabilities between the control plane and data plane. The system targets bad behaving TCP connections and filters them and defines thresholds on the number of flows that can be initiated from one part of the network.

2.4.6 Quality of Service (QoS)

The research in QoS has been targeted from the first days of SDN because of the powerful tools SDN provides. There are a lot of papers in the literature that approach the subject from different perspective. Hedera [41] and DevoFlow [42] detects elephant flows and directs them to the best path. The same was proposed by [43] but the detection is done in host OS rather than the switch. On contrast MiceTrap [44] proposes that it is more important to detect mice flow and aggregate them than elephant ones.

B4 [15] is a Google's implementation of a centralized QoS of its Data Centers WAN connections. They target better load balancing to achieve high throughput. In [45] and [46], using collected network information, the researchers propose path selection techniques.

A problem that has been speculated upon since the inception of SDN was the scalability of networks using SDN. The bigger the network, the bigger the flow tables needed to direct traffic. Also, larger networks means a lot of control messages

exchanged between the switches and the controller. This puts a lot of load on the controller and might cause a network failure. Both of those problems are under the scalability issues of SDN. In DevoFlow [42] the use of wildcards was extensively adopted to cut down on the number of flows. Also, in Kandoo [39] the researchers suggested a layered hierarchical architecture of controllers where each controller is responsible for a zone in the network. Only the root controller is aware of the whole topology. This architecture cuts down on the number of flows needed and the number of switches each controller manages.

Chapter 3

Delay using Software Defined Networks, Limitations, Challenges, and Suggestions for Openflow [1]

3.1 Introduction

Quality-of-service has been a hot topic for researchers since the inception of computer networks. Delivering a good experience to the end user is a hard problem because in most cases the end-to-end connections run through different networks with different configurations and priorities. This made guaranteeing minimum service to the end user difficult. Many of today's applications and services need minimum network parameters to function correctly. Some applications put hard bounds on some parameter especially in today's IoT devices [47][48]. For example, media streaming have minimum requirement on bandwidth while being tolerant to delay and jitter (by using buffers). On the other hand, online video games have low bandwidth requirements but less tolerant about high delay and delay jitter. An example of a service that is affected by all parameters is video conferencing which requires minimum parameters across the board.

When we started thinking of a solution to this problem, we knew that a way to measure performance between the two ends of the service is mandatory to gauge

[1] This chapter is part of the work published in the ICCNCS 2020: International Conference on Computer Networks and Communication Systems 2020 [153]

the quality of the service presented. From this information the network can allocate more bandwidth, divert traffic to other paths, or even block the service. Measuring traffic parameter in a LAN environment is easy because it has a single administrative domain. In some implementations [49, 50, 51], agents were installed in the network to measure the load and allocate desired resources. Sadly this cannot be implemented in the greater part of client server deployments. For this reason the capabilities of SDN came into the picture. SDN enables the measurement of these parameters in a predetermined manner with end points that are not necessarily in the control of the network administrator. With the power of SDN the controller can time flows, fabricate packets, alter packets, and get counters to measure certain parameters.

In this chapter we study techniques that measure the Round-Trip-Time (RTT) between communicating parties using SDN networks (we will use the terms RTT and delay interchangeably). As for why we choose RTT instead of other parameters was that RTT is a byproduct of bandwidth. Unfortunately the available bandwidth between two hosts in the network (or the internet) is hard to measure since it spans multiple networks. Those networks belong to different operators and in multiple jurisdictions. An idea similar to speed testing, it might be easier to fill the connection with data and see how much bandwidth it can serve. This will only serves to measure the current condition of the connection and trying to periodically do it will cause saturation, overhead, and congestion in the network. On the other hand the RTT can be estimated and measured by a number of techniques that add little or none to the existing traffic. Also, delay jitter is the variation in delay measurements.

In most previous works that dealt with measuring delay, authors have categorized delay measurements to two main categories: passive and reactive. In the passive approaches, the algorithm didn't introduce any extra packets but used the information in the traffic to measure delay. On the other hand, the reactive approaches introduced probing by sending extra traffic in the network to get better measurements. In our case we didn't categorize any of the algorithms as passive or reactive because all of them add extra packets either between the hosts being tested or between the controller and switches. We instead chose the path of showing the algorithm load on the controller, network, and how many extra packets it introduced.

Since 95% of the internet traffic is TCP [52] (non peer-to-peer), we concentrated on techniques that favor measuring TCP flows. In addition, most TCP applications are designed with time constraints in mind which is an additional reason to measure delay. Also, TCP protocols have rules and policies that can be exploited to conduct

measurements. These include port numbers, flags, sequence numbers, and unique packets.

3.2 Previous Works on RTT Measurements

Before the invention of the idea of SDN, passive delay measurement was studied in a lot of works. In [53] the authors proved that delay in the network follow a shifted gamma distribution with varying parameters. Different aspects of network measurements were studied in [54] including the effect of delay on timing compression, available bandwidth, and queuing time scales. Two passive measurement techniques were studied in the work in [55]. Those were the SYN-ACK and the burst difference techniques. Both showed accuracy within 90% of the correct measurement. The researchers in [56][57] used passive measurements to get accurate delay measurements. In a recent work [58] the authors studied the applicability of timestamps in measurements of delay and the effects of stretched acknowledgments on the passive measurement.

When looking at the literature of active measurements of delay, we find that most works add agents in end points or add middle point testing nodes. A tool called Surveyor [59] was implemented to measure various network parameters. Another software system was built with the intention of creating a network monitoring system. This system was called NIMI [60] and the work discusses the challenges faced with such implementation. In [61] the authors lay down the design of a network monitoring framework.

In addition, one-way measurements were discussed in many works including [62] where two intermediate measuring devices were used to modify timestamps. Those devices would then use the packet's timestamps to measure the one-way-delay of the connection. Also, a set of synchronized nodes were used to monitor packets and measure their in-flight time in [63]. GPS time with the packet arrival time was used in [64] to measure the one-way-delay.

All the previously mentioned research works were implemented in what we will call here classical networks. These works conducted their experiments in one of three ways. The first and the most used methodology, was to collect network traces. This allowed the researcher to analyze the collected data with ease. Although this method can be satisfactory for its intended use, it ignores the ever changing nature of network traffic and burdens the collecting and analyzing devices. The second, method is installing

software in one of the end points to intercept traffic and analyze it. This process can be accurate but is not applicable if the end host is not under the control of the testing authority. The third method is to install devices on the path that is intended for measurements. This approach is similar to the work in this chapter and to what SDN can achieve.

In SDN research, most works focused on measurements as a whole. In [65] the researchers introduced OpenNetMon which is a mechanism to measure delay between two switches inside the SDN network. This is done by making one switch send a packet to another switch and calculating the time difference upon receiving a notification of reception. Also, the work in [66] used probing packets sent in a loop between the controller and the two end switches. The recent work in [67] did performance measurements using traces collected near the server.

We here state that the target of our study is the end-to-end RTT not RTT inside the SDN network. In most implementations of client-server, the connections run through multiple domains that are not accessible to the SDN controller. For this reason and others mentioned in the introduction, the above works [65, 66, 67] do not apply. All of our algorithms obey the rule where the algorithm is not allowed to access information at any of the end points. We went above and beyond to be compliant with the stated rule.

This chapter is organized as follows. The first section defends the use of delay as a metric and how it enables controllers to provide better service to end users and applications. The second section discusses the algorithms implemented to measure delay in SDN networks with the advantages and disadvantages of each. Next we show the experiments conducted and their results. Finally we look into the hurdles that faced our implementation with Openflow and suggest additions to broaden the scope of its capabilities.

3.3 Delay and QoS

Delay between two hosts can be attributed to a number of different delay types. These are propagation delay, transmission delay, processing delay, and queuing delay. Since we are measuring the total delay experienced using a device in the middle of the communication link, we can safely say that the delay measured is the same delay experienced by the end hosts.

In today's internet, no matter how little information you get about the condition

Table 3.1: Application Classes and Their Demands

Class	**Application Example**	**Demand**
Interactive	www, Chat, Telnet	Low delay, low bandwidth
Bulk	www, ftp, P2P	high bandwidth (best effort)
Real time	Multimedia, games	Low delay, constant bandwidth
Email	SMTP, POP	Best effort
Other	DNS, application specific	Varies

of the connections, it is a valuable information. After all, most applications running in the internet are built on top of best effort protocols and are designed to tolerate various situations. Or are they? It is obvious that today's applications are becoming more demanding with expectations of minimum requirements. With that in mind we can expand our knowledge of the RTT collected to tailor the needed conditions for the applications. As we mentioned before applications can be demanding in one or more aspects or none. A lot of research has been done on the characterization and identification of application traffic [68][69]. By characterizing the application, either by learning them from the behavior or following predefined rules, the controller can serve connections with the needed demands. In many works applications can be split into 5 classes [70]. Table 1 shows an example of such applications and their demands.

Translating the RTT into allocated bandwidth is straight forward, that is giving the flow enough bandwidth for it to traverse the network in a calculated time. Conditions that are outside the controller domain are monitored and the controller can change the traffic to adapt to them.

In essence, the controller knows how much it is acceptable to delay flows and from that it can allocate the needed bandwidth (management and provisioning of queues [71]) or choose different routes. This will introduce some kind of priority between different flows. Added to that the knowledge of the behavior of the traffic will allow the estimation of how much data will be sent per-unit of time. Such priority system was utilized to assign bandwidth to applications in the Google B4 network [15]. This will allow the controller to keep the average delay within acceptable boundaries which cater to the characteristics of the application. To illustrate these ideas, let's tackle the following points.

- A best effort application like email can be given lower priority with high delay

and lower bandwidth until the network has enough resources.

- Some games need fast response times between servers and players. Having more than 150ms RTT is not acceptable. So diverting the client to a server closer to the player is required. This is usually done in software but allowing the network to measure and choose the best location is robust and efficient.
- Estimation of RTT is essential in queue management and buffer allocation [71, 72, 73].
- A changing delay might mean congestion on the path between the communicating parties [74][75].
- In systems where central leader or the closest node discovery is used [76][77][78], information about RTT is essential.

3.4 Delay Measurements

Compared to classical networks, SDN provides the means to detect new flows, gather flow information, and make programmable decisions. Also, an SDN network can create artificial packets or modify flow packets that can be inserted into flows. Having that in mind, we started looking for the best techniques that utilize the capabilities of SDN. The following subsections provide description of the implemented techniques, the algorithms, and pros and cons of each.

3.4.1 Effect of Measurement Load on the Controller

Since SDN is a programmable network, the controller learns the delay periodically to each host in the network. This information is kept in a database which needs not be updated frequently except if there are big changes. Also the time for the periodic test need not be short especially if the measurement is consistent. Since applications can start many flow streams between server and client, the controller should not be testing for each flow separately. Instead it should measure the delay to the end points regardless of how many connection they have. Also the controller can offload the testing and information maintenance to another device. We are going to discuss the load of each algorithm and in the Openflow suggestions section, remedies are going to be addressed.

3.4.2 ICMP

Every network engineer's first choice of measuring of the round trip time is ping. This tool makes use of the ICMP (RFC 792)[79] request and reply. It is mandatory for any network stacks to include the implementation of the basic echo and reply. The measuring host (A) sends an ICMP echo request to the other host (B) while keeping track of the sending time T1. Upon receiving the echo request, host (B) replies with an ICMP reply. When host (A) receives the reply it records the current time T2. The round trip time is the difference between the two recorded times (T2-T1). We broke down this process because we would like to emphasis the differences between the normal use and our implementation in an SDN environment.

To implement the ping mechanism in an SDN network one can either use the hosts communicating or the controller. The first approach requires the presence of an agent in the host or a root account where the controller can execute commands. This is impractical and will introduce vulnerability in the network. The other choice is to make the controller imitate the measuring hosts. This has the flexibility of being able to initiate the measurements at the correct time while keeping the collected data centralized. The test made by the controller can be different from the ones made by host. It can be made in such a way that it doesn't introduce congestion and be separated by controlled time periods to measure different conditions. The algorithm is as follows:

1. Upon receiving a new flow from host (A), the switch consults the controller of which path to send the flow to reach the destination (host B).

2. After installing the path in the switches, the controller must wait for a predetermined time to allow for path establishment.

3. Then the controller creates 2 ICMP echo requests. One has host (A) as the source and host B as destination. The other has the inverse where the source of the ICMP packet is host B and the destination is host A. Both packets are sent to one of the switches in the path preferably the closest to the flow source (to allow more time for path setup. Also it is guaranteed on the path).

4. A timer is started T1. This timer can be either kept in a tracking table or embedded in the packets body. The latter is easier and more efficient to implement.

5. At the same time the controller should install flow matching for the replies to those two packets in the switch. It is preferable to send the flow modifications before the two ICMP requests.

6. Upon receiving the replies from the switch, the controller reads the current time (T2) and subtracts it from the time recorded in step 5 (T2 –T1)

7. The measured time is not the RTT between the hosts since it includes the RTT between the controller and the switch (SRTT). So we need to subtract that time from the result in step 7. The controller can measure this excess time by doing a ping to the switch or using the LLD messages (used in Openflow) to measure how long does it take to reach that switch.

$$RTT = T2–T1–SRRT \tag{3.1}$$

8. The table matching entry (in 5) in the switch should be removed after this process is complete.

Since we are fabricating packets in the network it is essential to extract those packets from the network or errors might occur upon receiving them by the hosts. The power of SDN relies on being able to control the environment where such scenario will not happen. Compared to classical networks implementation, there are no additional requirements in the end hosts. But as in other implementations that relies on ICMP, this technique have its problems. Here we are going to list both pros and cons of the ICMP technique.

Pros

- Simple to implement with low overhead.
- Controller can conduct the measurement at any moment.
- The measurement is done separately for each host which enables the collection of data on individual hosts.
- Can make measurements for TCP and UDP connections.

Cons

- ICMP is a low priority traffic. Some ISPs will prioritize other traffic over ICMP. This will make the measurement of the RTT inaccurate [80].
- There are security concerns with ICMP enabled devices [81]. It has been a best practice in the security domain to disable ICMP replies. This was done to mitigate ping flood and denial of service attacks.

The installation of matching table entry in the switch to capture the ICMP reply will cause all ICMP replies (between the two hosts) to be sent to the controller. Up to the latest version of Openflow, support for matching on ICMP sequence numbers does not exists. It is not possible to selectively catch the replies without catching other ICMP replies (initiated by the hosts). This can be ignored because the time between installation and removal of table entries are in milliseconds and it will be empirically unlucky for it to happen at the same time.

3.4.3 TCP 3 way handshake

Another technique that has been used to measure the delay passively in classical networks is TCP 3 way handshake. The setup of TCP connection is always initiated by a sequence of packets that are identified by their TCP flags. The first is the SYN packet which is sent from host (A) to host (B) (to establish the connection). If host (B) can serve the application on the destination port it will reply with a TCP packet its SYN-ACK flags is set. Upon receiving this packet host (A) will confirm the reception with an ACK packet. This will complete the 3 way handshake process and the application can then have a reliable connection. The TCP layer will guarantee the reliability of the connection by including sequence numbers in the header which serves the upper layers with an ordered stream of data. As shown in figure 3.1 (a), the time between sending the SYN packet and the reception of the SYN-ACK is the RTT. This is only true if the measuring entity is the initiating host. In SDN networks, we rely on the controller and switches to measure the time and it is hard to know when the packet was sent by the source. This means that to implement such a technique, the controller must rely on switches in the path between the communicating parties (figure 3.1 (b)).

In the early Openflow implementations, matching on TCP flags was not possible. In later Openflow implementation this was incorporated and we can use it to

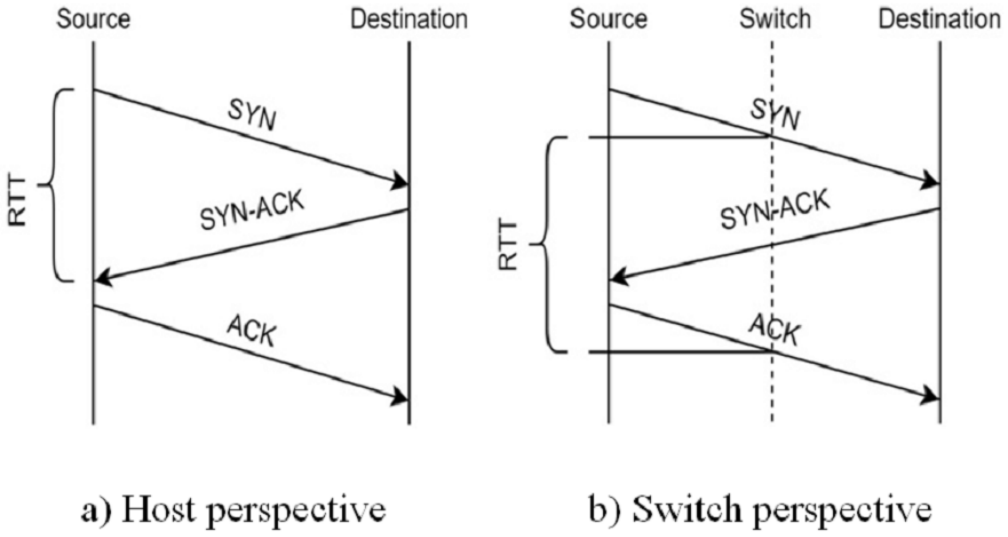


Figure 3.1: TCP 3 way handshake.

detect both the SYN and SYN-ACK flags. The obstacle we encountered with the implementation was detecting the third leg of the connection establishment. Since the third packet only has the ACK flags set, it can match any acknowledgement sent from source to destination. Our solution to this obstacle was to setup a match on the ACK packet as soon as we receive the SYN packet. When the controller receives the intended packet from the source, the matching on the switch is removed. The reader might speculate that the controller will be overwhelmed with acknowledgments before the match table entry is removed. This is partly correct and it depends on how far the controller is from the source and if the source is the host that sends the data. In addition, TCP at the beginning of communication will not send many packets because it will go through its slow start phase. In our experiment the number of excess acknowledgement packets received by the controller is around 10 (first burst). This problem could have been mitigated if Openflow had the option of sending one packet to the controller (or predefined number of packets). We will discuss a number of suggestions to Openflow that would solve such difficulties. To implement this technique we suggest this approach:

1. The controller installs a table matching for both the SYN and SYN-ACK packet in the switch. Since the controller doesn't know when the connection will be established it will install the entries preemptively. A path to the destination (host B) should also be installed so as not to delay the packet. Waiting for the controller to setup the path for the new connection will add additional delay.

The same is done for the SYN-ACK packet which has the reverse source and destination parameters.

2. When the controller receives a SYN packet it records the time (T1).
3. The controller installs a table match entry in the switch to capture the ACK packets from host A. The packets sent from the switch to the controller should be a copy of the original so not to delay the data and introduce errors.
4. When the controller receives the SYN-ACK it records the time (T2)
5. Then the controller removes both the table matching on SYN and SYN-ACK.
6. At reception of the first ACK packet from host A, the controller saves the time (T3) and calculate the RTT (T3 – T1). In contrast to the ICMP algorithm, the controller does not need to subtract the RTT from controller to switch since this delay is a shift in the overall time.
7. The algorithm can calculate the RTT of both side of the connection to the switch by the using the collected times. This can help in information gathering on individual end hosts.

Pros

- Can have information on both end hosts which can be used individually.
- Gather information passively.

Cons

- Introduces overhead packets that may overwhelm the controller and the network if the controller has a long delay to the switch.
- On bigger networks the preemptive installation of entries for both SYN and SYN-ACK can fill the limited memory dedicated for table matches.
- Can measure the delay only at the beginning of the connection.
- Connection establishment might take more time than normal TCP communication [82].

3.4.4 Sequence Numbers

As we mentioned previously, TCP keeps track of packet ordering using a fields in the TCP header dedicated for sequence numbers and Acknowledgment numbers. The receiving entity indicates that it has received a particular packet by acknowledging it (usually by setting the acknowledgement flag and sending the sequence number plus one as an acknowledgement number). The sending node can transmit up to a maximum number of bits controlled by what is called a window size.

Using sequence numbers to measure the delay between two hosts is intuitive. When a packet with sequence number X is detected, wait until an acknowledgment of X + 1 (or greater than X) is received and that will be the RTT from the destination host to the switch. But Openflow doesn't provide the tools to do this process cleanly. First of all, Openflow doesn't provide matching on sequence numbers (or arbitrary header fields!). Secondly, there is no conditional logic in the switch which can help detect sequence numbers greater than certain values. Also, as we mentioned in the 3 way handshake we only need to detect one packet and Openflow doesn't support such functionality. We implemented an algorithm with the current capabilities of Openflow with the intention of showing the problems that may arise. We could have made changes to the Openflow standard to fix these problems but it is easy to tailor a problem and individually solve it to show good results. This was not the intention of this work.

1. Match on SYN and save source and destination (match on the FIN flag is used to track when the session stops).

2. After a small time period the controller installs a table match on all packets. There should be 2 (one for each host). This is done to catch a packet from each source.

3. Also at the same time the controller installs a table match on ACK packets (for both end hosts). The table match has a lower priority than the one in step 2.

4. The moment the controller receives a packet from one of the end hosts it removes the table match entry in step 2 for that host.

5. Because we have a match on ACK we will receive ACK packets (the ACK with the sequence number we want comes with a source that is the destination of the packet in step 4). When the controller receives the needed ACK packet

(the sequence number in 4 plus 1 or just greater than), remove the matching on ACK packets.

6. The time between receiving the packet in step 4 and step 5 is the RTT between the host and the switch. Since we are doing this step for both hosts then we have 2 measurements of the RTT.

7. The steps from 2 to 6 can be repeated to get better measurements until we get the FIN flag.

8. Print number of packets received from both hosts (so we know how many excess packets we received).

At first look this algorithm might look very demanding and produce bad results but we found that it can work very well in interactive traffic (light weight). A dedicated device can be used to mitigate the problems with this technique.

Pros

- Can probe the two hosts while the session is running.
- Uses information from the communication session.

Cons

- The most troubling problem is how much information is sent and processed by the controller. Although the time period between installing the table match entries and removing them is small(milliseconds) , the controller is overwhelmed with packets that cause it to delay processing to the point of getting wrong measurements. We are going to look deeply into the results in coming chapters.
- Some TCP implementations take advantage of techniques that delay the response acknowledgement packets. These are Delayed Acknowledgment and Selective Acknowledgments.
- Harder to implement and introduce lots of overhead in the network and unnecessary processing in the controller.

3.4.5 TCP connect

In our journey to find novel ways to measure the delay that use the capabilities of SDN, we tested the applicability of some techniques. One of the techniques that use the mechanism of the TCP protocol is the ability to exploit the information in the TCP header to initiate a new TCP connection. To test the RTT to the server, the controller can initiate a TCP connection between it and the server using the information intercepted in the traffic packets. The SYN packet can be recreated with the same header information except for the source IP and MAC addresses which can be changed to those of the controller. As a reply, the controller can receive one of two packets either a SYN-ACK packet or a RST packet. Both will give the correct RTT to the server. The controller should subtract the RTT to the switch in the path between the two communicating parties. The reader might predict there will be errors in the TCP stack and that will be the case only if the TCP connection was not handled properly. The controller should acknowledge the SYN-ACK packet and then it should close the session properly so it doesn't hold an unused socket at the server.

So with this we handled one end of the connection, how can we deal with the other end? Let us ask this question, what happens when we do the same for the client? As a standard implementation of the TCP stack, if an incoming TCP connection tries to connect to a non existing port, the host replies with a packet indicating the problem. This packet has the RST flag set. The RTT can be calculated by subtracting the time when the SYN packet was sent from the time when the packet with the RST is received (same as the implementation for the server side). Similar to the connection to the server, the communication protocol standard should be respected so not to introduce errors. The client will keep on sending this packet with ever increasing times periods until it receives an acknowledgement from the controller.

This approach might have problems if the communicating parties have firewalls that only accept connections from certain subnets or IPs. Also, it might introduce some security issues if the security auditors are looking closely at out of the ordinary connections. For the most part, this approach will work and TCP by definition has to react accordingly.

1. The controller uses the information of the established connections to create two new SYN packets with the controller as source and the two end hosts as destination.

2. The packets are sent to a switch in the path of communication between the two hosts.

3. Upon reception of the replies the RTT is calculated.

4. The TCP protocol standard should be respected in the process by closing the session with the server and acknowledging RST packet.

Pros

- Little overhead in the network and simple to implement.
- Can be initiated at any time to get better measurements of the network changing conditions.
- Measurements can be done separately for each host for better information on individual host delays.
- Can be implemented for UDP traffic (same way!).

Cons

- The most concerning problem is that a bad implementation can introduce errors in the TCP stack and applications can react unexpectedly to such connections.
- Firewalls might block connections coming from the controller. To mitigate this, the controller can use the communicating party's information. This can introduce other issues that are not in the scope of this work.
- Security concerns might arise out of unusual connections.

3.4.6 Other Considerations

To measure delay one can use other techniques that we didn't implement. This was due to the limitation of Openflow or due to the way the SDN work as a whole. We here list those and why they were not considered.

TCP Timestamp

A header option introduced in RFC1323 [83] added a time stamp in the TCP header. When a host wants to measure the RTT to the other end, it inserts the time in the TCP header. When the other end receives this packet, it copies the timestamp to the header of the acknowledgement to that packet. Upon reception, the measuring host compares the time in the header to the current time to calculate the RTT. Most administrators turn off this feature for security concerns. Also, Openflow doesn't provide the means to implement a coherent implementation to intercept this information. To show why this is the case, let's imagine trying to set a time to be echoed back to one of the hosts. First, the controller has no means of knowing when the packet will pass through the switch to install the time option. Also, detecting the acknowledgement to that packet means sending all acknowledgments from the destination because Openflow doesn't match on that field. Although setting fields is present in recent Openflow implementation, they do not include TCP specific fields [84].

Network Trace Recording

In works previously discussed, the most dominant implementation of measurement of delay was using network traces. In SDN it is easy to record the network traffic by using the controller or any agent that the controller can access. From the recorded trace, the controller can apply the techniques that we mentioned in this work to measure the delay. We did not implement this technique because it will have the same results and problems of the sequence number technique discussed above.

Burst difference

Another method that was used to measure the RTT was the time difference between two consecutive bursts. The nature of TCP allows the sending host to transmit as much data as it can. That behavior is throttled by the window size of the receiver. This will make up the first burst. When the receiver acknowledges the first burst the sender can resume sending data which is the second burst of the communication. The time difference between the two bursts can be identified as the RTT. In SDN, to detect those two burst the controller need to receive most of the packets sent or need to excessively poll the switch meters to pinpoint when the second burst happened. Because Openflow doesn't provide the means to solve these problems cleanly, we refrained from implementing this method.

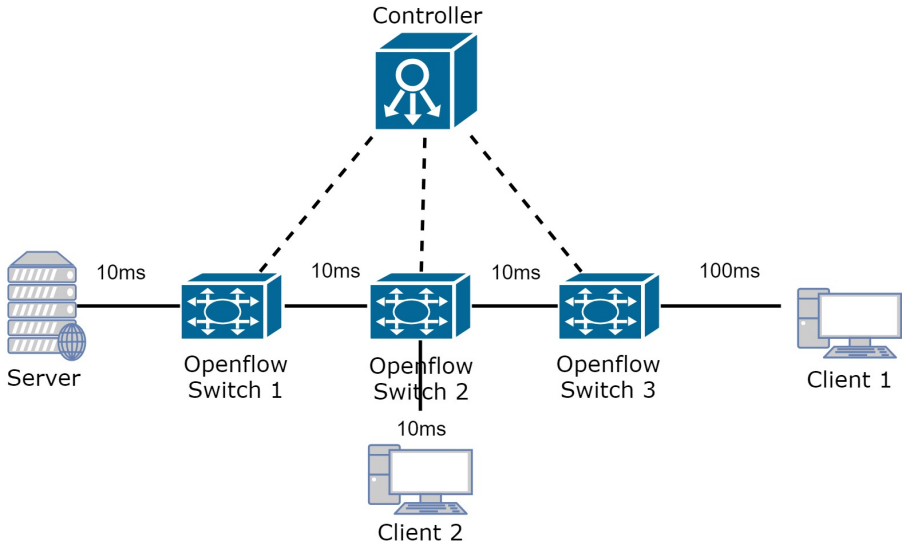


Figure 3.2: Test network

3.5 Experiments and Results

As a controller of choice we found that Ryu [52] was the most versatile and supported the most suitable Openflow version [84]. Mininet [85] was used to build a test network as shown in figure 3.2. All links have bandwidth of 10 Mb/s and a delay of 10ms except between S3 and C1, which was set to 100ms. We built the network this way so we can measure the RTT between the Server and Client 1 while introducing competing traffic from Client 2. We first tested the four algorithms in the ideal case where there is no other traffic in the network and then repeated the experiment with congestion on the link between Switch 2 and Switch 1. The test was conducted one thousand times for each algorithm and scenario. This was done to examine the differences between the four implementations and how they react to different network conditions. The switches used in the experiment are OVS [86] switches and since they communicate with the controller through the loop-back address, the delay between the controller and the switches was ignored. We used Iperf (bulk traffic) and Netcat (light traffic) to generate TCP traffic.

As a virtual environment, the test bed gave us insight into the the mechanics of the algorithms and how they affect the network and controller. It is understandable that a virtual environment cannot produce accurate measurements but it gave relative measurements (between the algorithms) that show what are the advantages and disadvantages of each. Also, implementing the algorithms exposed the difficulties and

Table 3.2: Experiment results, time in milliseconds

	Ideal Network					Congested Network				
Algorithm	ICMP	TCP # Connect	3 Way # Handshake	Seq # Interactive	Seq # Bulk	ICMP	TCP # Connect	3 Way # Handshake	Seq # Interactive	Seq # Bulk
Mean	269.5	265.4	275.7	307.3	seconds	2174	2191	2143	seconds	seconds
STD	6.3	3.2	2.8	3.7	seconds	223	221	446	seconds	seconds
Excess Packets	2	2	10	5	5K	2	2	10	5	4K

shortcomings of Openflow.

Table 3.2 shows the mean and standard deviation of each algorithm measurement. The standard ping test between the server and client is 262ms. The results show accuracy above 95% and that using the TCP connect algorithm gives the best results. This result can be attributed to the fact the TCP traffic take precedence over ICMP. The differences between TCP connect and the 3-way handshake is due to a delay contributed by the sending host where it needs to send its acknowledgement (first data packet to be sent by the source of Iperf). As for the Sequence Number algorithm we conducted two different tests. One representing bulk traffic (Iperf) and the other representing interactive traffic (Netcat). We made two tests to show the differences between the behaviors of each algorithm. As can be seen from table 3.2 the sequence number algorithm can produce results that can be considered good if the traffic is interactive. Nevertheless, if the traffic is bulk then it will create an overwhelming traffic at the controller which cause the results to be in seconds.

The table also shows the results of conducting the tests in a congested network environment. The ICMP and TCP connect gives good results with low variability. Also, the 3 way handshake gives better results but with a larger standard deviation resulting from the excess traffic and queuing of ACK packets. As for the sequence numbers algorithms, both produce high measurements of delay. Although the interactive traffic produced much lower values, it still gave results that were not acceptable compared to the other three techniques.

3.6 Additions and Suggestions for Openflow

We started this work to show how to measure end-to-end RTT using SDN and to investigate the capabilities and limitations of Openflow. This work gave us an insight into the issues that hinders Openflow from being a better programmable network framework. In this section we are going to discuss those issues and suggest solutions

to them. Stating that, we don't claim originality of those ideas (some are implemented in other SDN implementations). We are trying to expose the problems we faced and solutions for them.

3.6.1 Sending a Set Number of Packets

From what we have seen in both the sequence numbers and the TCP 3 way handshake subsections and the results, there are large excess number of packets. This problem stems from the inability of the controller to manage the number of packets sent to it. If Openflow had an option to receive only one packet or a certain number of packets then this problem would've been mitigated. Also, this option can have a time period. Upon expiration the action is enabled again.

This option can help in the previously mentioned implementations and many others that need to probe certain flows. One application that comes to mind is in the security domain where the controller needs to inspect packets on regular basis. Also, from management and monitoring perspective there exist applications that need to receive samples of the traffic without overwhelming the controller with too many packets.

3.6.2 Matching on Arbitrary Protocols

This issue has been discussed in the literature thoroughly. Openflow supports Ethernet as a Datalink layer and IP as a network layer. In recent versions, support for TCP, MPLS, and others were added. With ever changing network protocols, it is hard to keep up with the new additions and implementations of each protocol. Also, a generalized implementation of a programmable network should work with all kinds of protocols. The works in P4 [87] and Openstate [88] introduced new frameworks for a more generalized programmable non-protocol specific implementation.

We suggest a middle ground between Openflow and non protocol specific implementation where Openflow can have the option to match on a hex mask that represent the targeted packet header. The switch will have the task of applying the mask to the incoming packets (XOR or similar operations). This operation is very cheap and can be done in hardware. In the controller the specified matching is translated to this hex representation using an interpreter. This process can be done to supported and non-supported protocols which makes the switch a more generalized device.

3.6.3 Logic in Openflow

Security researchers and administrators will not like this suggestion but the capabilities that come with such addition are too big to ignore. The BEBA project [89] is discussing a similar approach. Although we want the switch to stay as simple as possible, introducing a non-Turing complete instruction set will make Openflow switches more powerful network devices. We suggest adding variables, conditions, and basic mathematical operations to this instruction set. Just like Bitcoin [90], keeping the instruction set non-Turing complete makes the instructions set clear, simple and secure. The ability to count, branch, and modify behavior depending on the situation is a powerful tool. After all, if the control plane is already compromised then the whole networks is.

3.7 CONCLUSIONS

In this work we investigated the importance of delay and how it can be used to provide better services to applications. In addition, how the measurement of delay help the network assign a tailored quality-of-service. A number of techniques to measure end-to-end delay were discussed and implemented. We saw the implementations results and the problems that accompany them. We then discussed improvements to Openflow that would solve some of these problems.

Chapter 4

Network Traffic Prediction using Quantile Regression with linear, Tree, and Deep Learning Models [1]

4.1 Introduction

The progress in machine learning over the past decade has paved the way for better and more accurate modeling of processes. This coupled with the ability to collect and process large amounts of data, allowed for the extraction of meaning and value out of such data. Prediction is one of the most studied machine learning application. The benefits of prediction are countless and task dependent. In the field of Computer Networks, traffic analysis, classification, and forecasting have been improving with the advancements in machine learning and Data Science.

In the previous chapter we investigated the power of SDN and how it can be used to collect information and data from the network. In this chapter, we are going to take it to the next step which is using that data to have insight on the behavior of traffic. This is accomplished by predicting how much traffic is needed in an arbitrary future time frame.

In the mid 90's, the consensus on self-similarity and Long Range Dependence of network traffic came to fruition [91, 92, 93, 94]. This opened the door for researchers

[1] This chapter is part of the work published in the IEEE 45th Conference on Local Computer Networks (LCN) 2020

to investigate the predictability of network traffic and the applicability of modeling it. The literature has tackled the predictability of network traffic from many angles. One aspect that was studied in traffic prediction was using statistics to model and predict the dynamics of traffic [95, 96, 97, 98, 99]. Another was characterizing traffic pre-flow behavior where the flows were predicted either as elephant or mice flows [100, 101, 102, 103]. Identification of applications [104] [105] and application types [106, 107, 108] are also fields of study that use network traffic for forecasting. We will investigate the predictability of per-day traffic time series.

The benefits of getting accurate prediction for the network traffic are vast. Consider the ability of a client to subscribe only to what they need to use per-day. The difference between bandwidth needs at peak times and off times can be double or even more. Figure. 4.1 shows five day bandwidth used between two subnets [109]. Predicting traffic can also help in scheduling, provisioning, and planning for bandwidth usage. In addition, it can help in resource allocation, detection of anomalies, congestion, automated QoS, and management of the network.

In this work we will look into using quantile loss to build Prediction Intervals (PI) for aggregate network traffic. The models used range from linear models to tree based and neural network. We will study the applicability of those models to produce good estimate of the PI and to compare the results using multiple model setups.

4.2 RELATED WORK

Collecting network traffic usage at equal intervals and studying its characteristics is part of the general field of time-series analysis and research. Mathematical models have been proposed to describe the behavior of such data. These models can be characterized as linear and non-linear. Both types have many models, each with its advantages and disadvantages. Predicting network traffic has been studied in many works that covered those two categories. In particular for linear models, most works covered the applicability of Autoregressive (AR) models and its derivatives to network traffic prediction. Using ARIMA, the researchers in [110] predicted network traffic in campus network. Also, video traffic was predicted in [111] using the same prediction model. ARIMA was used in many other works [112, 113, 114, 115]. A derived model used Fractional Gaussian Noise (FARIMA) [116] to predict internet traffic [117]. Another linear model used HoltWinters in [113] for the forecasting traffic.

Because of the simplicity of linear models and how they lack the long range depen-

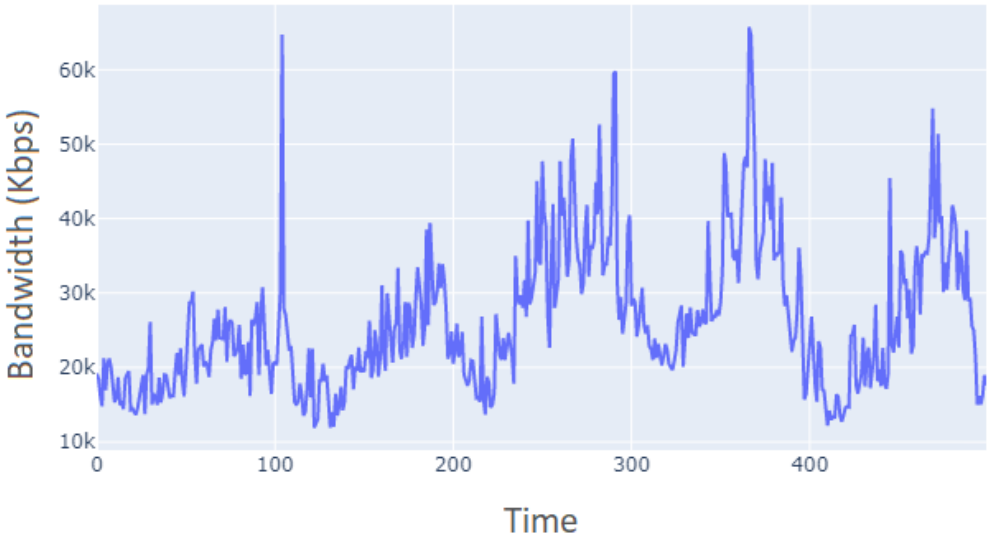


Figure 4.1: Used bandwidth in a five day cycle in kbps.

dency researchers started testing non-linear models. The results gave better prediction models. Some even used both to combine the strengths and benefits [118]. Before the breakthroughs in the Artificial Neural Network (ANN) specially in dropout, activation, and overfitting, predictions were basic. These were used to predict network traffic [119, 120, 121]. Later years, better architectures were adopted to forecast network data usage. In [122] the authors used backpropagation and gradient descent to predict university future bandwidth usage. A more in depth study was made in [123] to predict different traffic sources using Feed Forward Neural Networks. This architecture was also used in [124][125].

In recent years, the research on ANN achitectures reached a consensus that the best ANN to model time series data is Recurrent Neural Networks (RNN) and its alternative Long short-term memory neural network (LSTM). These were used in the works [126, 127, 128] and showed that they are the best in capturing the LRD and self-similarity of network traffic.

Most works highlighted here use uni-variate feature space where only the bandwidth is taken as a feature. We expand the feature space to different time periods. This will help the models to identify patterns in the data.

In this work we will study the applicability of some of those models and others to establish prediction intervals for network traffic. We will start by discussing why prediction intervals can be a powerful tool. Then we are going to investigate the models used. The last part of the paper will examine the data, experiments and results.

4.3 Prediction Intervals (PI)

There is no model that can claim exact description of a real world process. Models are proposed to try to get as close to reality as possible. There will always be some hidden variables, noise, or errors in the data modeled against. Also, some models can capture a number of features while disregarding others. That's why when comparing between models, the researchers must include a comparison measurement of the error. It has been the norm for prediction models to predict a point and then study how far it is from the real value. But recently the focus was shifted to produce models that can give upper and lower bounds on the predicted data. It is much more meaningful to say "the annual wage of a professional python programmer is between 100K-127K dollars [129]" than to say "the annual wage for a professional python programmer is 110K". The first statement is more meaningful because it gives a better insight into the data and shows the possibility of getting a wage in the upper percentile. Also, it shows estimation, uncertainty, and distribution of the data. It is even more meaningful to quantify those limits by saying "the 90th percentile of the wage is 127k".

Any model can describe a real process in the following way:

$$h_i = f(x_i; \Theta) + e_i \tag{4.1}$$

Where the function f is the estimate of the true $\hat{f}$. For regression, the observed values x_i combined with the model parameters Θ produce the estimation of the model h_i. e_i represents the error induced in the model. It has been the norm to treat the error as an independently and identically distributed (iid) value. This implies that it has variance σ^2 and a normal distribution of $\mathcal{N}(0, \sigma^2)$. A minimization of the cost function 4.2 will give the least square estimate of the model parameters[130].

$$C(\Theta) = \sum_{i=1}^{n} (h_i - y_i)^2 \tag{4.2}$$

Where h_i is the process being modeled, y_i is the estimated model output, and n is the set of data collected to describe the process. In linear regression (4.3)

$$y_i = \Theta_2 x_i + \Theta_1 + \epsilon \tag{4.3}$$

With the same error assumptions stated above, the minimization of the errors through the minimization of the likelihood estimate of the coefficients is as follows:

$$\hat{\sigma} = \underset{\Theta_1,\Theta_2}{\operatorname{argmin}} \frac{N-1}{N} \sum_{i=1}^{n} (y_i - \Theta_1 - \Theta_2 x_i)^2 \tag{4.4}$$

or the variance of the maximum likelihood of the estimation. As a result of our assumption that the error is normally distributed, we find the PI of the linear model from its proprieties as:

$$\begin{aligned} Upperlimit &= y + \sigma \cdot \underset{\alpha/2}{\mathrm{z}} \\ Lowerlimit &= y - \sigma \cdot \underset{\alpha/2}{\mathrm{z}} \end{aligned} \tag{4.5}$$

and we get for example the 95% prediction interval

$$[\hat{\mu} - 2\hat{\sigma}, \hat{\mu} + 2\hat{\sigma}]$$

This solution is limited by its definition. The model should be linear. The error is assumed to be iid and normally distributed. This formulation doesn't describe the majority of problems. If we look at the problem as an optimization of residuals, we find a general formulation that can extend beyond linear models. From statistics we know that the mean is the minimization of squared sum of residuals, while the median is the minimization of the absolute sum of the residuals. From the later we can assume that minimization of the sum of asymmetrically weighted absolute residuals will yield a quantile [131]. This is accomplished by giving different weights to negative and positive values. The most straightforward implementation of this loss function is:

$$C(\beta_i|\alpha) = \begin{cases} \alpha\beta_i & \text{if } \beta_i \geq 0, \\ (\alpha - 1)\beta_i & \text{if } \beta_i < 0, \end{cases} \tag{4.6}$$

Where α is the desired quantile and β_i is the difference between the predicted values y_i and the observed values h_i as in 4.7.

$$\beta_i = h_i - y_i \tag{4.7}$$

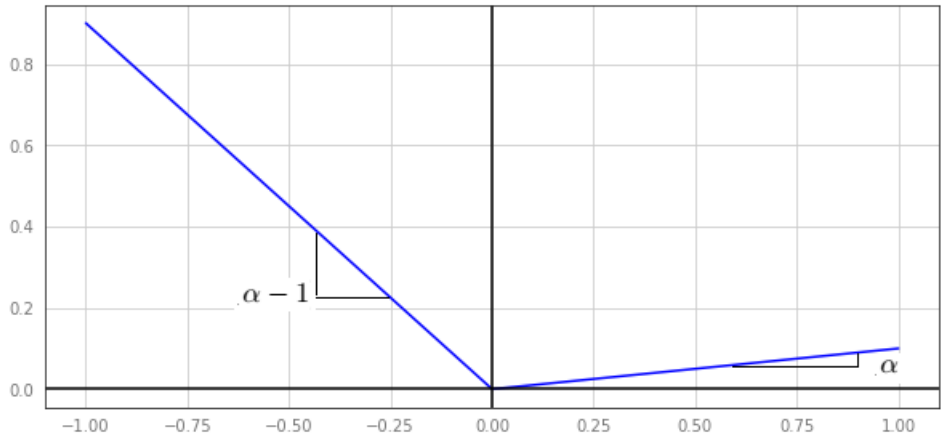


Figure 4.2: 90th Percentile loss compared to error.

Figure 4.2 shows the 90% quantile and how errors in the positive and negative are penalized differently. This formulation will be the basis of the loss functions we are going to use it through this work to find the PI in every prediction model.

4.4 Prediction Models

To cover a wider range of models we looked into the applicability of the most used linear, and non-linear prediction models. We did not consider ARIMA and its variations for two reasons.

- They were studied extensively in the literature.
- Those models are built for univariate data and we would like to take other features into consideration (time of day, day of the week).

We will start by describing linear models then tackle tree based and neural networks models.

4.4.1 Linear Quantile Regression (LQR)

This is the most basic form of regression as described in equation 4.3. The the error μ is assumed normal and has the iid property. LQR is essentially a regression of the median 50% of the loss function in 4.6. This model prevails (compared to other linear models) when modeling heavy tailed data[132].

4.4.2 Ordinary Least Squares (OLS)

OLS is another linear prediction model that has a similar form to LQR. It minimizes the mean instead of the median. To change the prediction from the mean to the median (and then quantile), we can add the result of the prediction (mean) to value of the inverse transform CDF of the median (or quantile). The reliance on the assumption that the errors are normally distributed, highly affect OLS compared to LQR. Outliers can have a greater impact on OLS estimates [133]

Both linear models produce good results when the stated conditions hold. They are flexible to changes and their time complexity is cheap ($O(p^2n + p^3)$ [134] p is the number of features). Also, they benefit greatly from changing training data to a supervised data.

4.4.3 Random Forests (RF)

Random Forests [135] is a branch of decision trees that uses ensemble learning to do classification and regression. Decision trees tend to overfit the training data and for that RF was proposed to train on different parts of the data (using bagging) [136]. Taking the average of the ensemble $T_b(x)$ will yield [137].

$$F(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \ .$$

To find the PI using RF, we are faced with the same problem in OLS. In [138] the authors suggested building a distribution out of all leaf observations (not only mean).

4.4.4 Gradient Tree Boosting(GTB)

Similar to RF, GTB [139] combines multiple estimators to produce predictions. But rather than using averaging to combine the results of individual trees, GTB uses boosting to combine weaker models to produce stronger ensemble. The model is built in a greedy additive way.

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

where $F_m(x)$ is the estimation minimization function of the loss and h_m is a decision tree at step m fit to the pseudo-residuals of the loss function which is [139].

$$h_m = \arg\min_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x_i)).$$

Tree based models vary greatly and one can be strong in one aspect while being weak at another. We choose RF and GTB because they predict the outcomes with two different opposing priorities. RF looks to the whole decision tree to reduce the variance. On the other hand, GTB deals with weak learners (smaller trees) to reduce bias. It has been suggested that GTB produces better results because of its versatility, simplicity, faster form, and ability to incorporate automatic predictor selection [140]. The computational complexity of RF is $O(n^2 p n_{trees})$, while for GTB it is $O(npn_{trees})$ [134].

4.4.5 Feedforward Neural Network (FNN)

A field of Machine Learning that has become very popular in recent years is Neural Networks (NN). It has become a very prominent prediction method and research intensive. The simplicity, fast convergence, flexibility, and adaptability to non-linear processes are some of the advantages of NN. There are many types of NN architectures each claim to solve one type of problem better than the others. The simplest form of NN is the Feedforward Neural Network (FNN) shown in Fig. 4.3 where the information moves from the input to the output while passing by an activation function (usually sigmoid). There are many types of FNN (Autoencoder, Convolutional, etc) but here we are going to investigate its simplest form. This in part to show the differences between it and the more specialized type which will be discussed next. In FNN, the NN consists of input nodes where the data to be trained or predicted is fed. It is followed by one or more hidden perceptron (nodes) that each gets activated or not depending on the sum of the products of the weights connected to it. This is done in sequence until the information propagates to the output. In training the difference between the predicted values and the actual is used to adjust the weights backward (Backpropagation). The sigmoid 4.8 function is used for activation instead of a step function because it has a continuous derivative which is very helpful in the backpropagation phase.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.8}$$

FNN has benefited from the advances in the field in recent years. Especially from advances in drop out, regularization, optimization, loss functions, and fast processing [141]. Also the sheer amount of data available made it easier for researchers to experiment with different implementations and techniques.

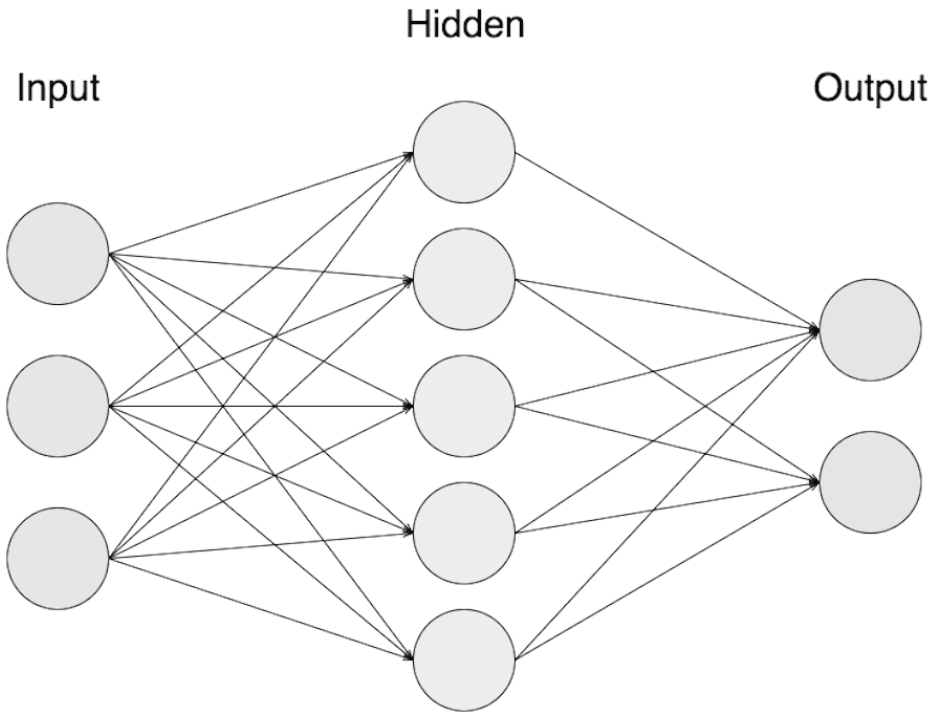


Figure 4.3: Simple Feedforward Neural Network.

Since NN uses a straightforward loss function, we can use equation 4.6 directly as our loss function. This limits our experimentation with other loss functions (better ones) but it should be sufficient to our purpose.

4.4.6 Long Short-Term Memory (LSTM)

The promising results of FNN and its variants, opened the door for the research and development to tackle other architectures. One of those was Recurrent Neural Network (RNN). In this architecture, the internal nodes have memory of the previous inputs. This state is kept hidden but influences the output of the node. RNN have been proven to be better predictors of sequences, handwriting [142], and speech recognition [143]. A variant of RNNs is LSTM where internal structures are introduced to solve the vanishing and exploding gradient problems [144]. The internal structures are called gates and they are:

- Input gate - This gate accepts the new inputs and through an activation segmoid allows or prevents the input to influence the internal state of the LSTM. Then it uses a weighing function (usually tanh) to push the input internally.

$$\begin{aligned} i_t &= Sigmoid(W_i \cdot [S_{t-1}, x_t] + \alpha_i) \\ \hat{C}_t &= tanh(W_c \cdot [S_{t-1}, x_t] + \alpha_c) \end{aligned} \tag{4.9}$$

Where S_{t-1} is the state of the previous node.

- Forget gate - This gate acts as a suppressor to weaker states from previous and current time slot. Both inputs are run through another sigmoidal function 4.10.

$$F_i = Sigmoid(W_f \cdot [S_{t-1}, x_t] + \alpha_f) \tag{4.10}$$

- Output gate - Similar to the input gate, the output gate consists of a sigmoid function followed by a weighing function. This influences the next step $t + 1$ where the first function decides on which information to be passed and how important it is.

$$\begin{aligned} O_t &= Sigmoid(W_o \cdot [S_{t-1}, x_t] + \alpha_o) \\ S_t &= O_t * tanh(C_t) \end{aligned} \tag{4.11}$$

LSTM's ability to keep information in its memory allowed it to be a good predictor for sequenced data, time series , and context sensitive training [145].

The computational complexity of NN can differ widely depending on the internal structure, number of epochs, and if the nodes are fully connected or not. In [146] the authors deduced that an N-2-N encoder can have $O(N^4)$. But in its simplest form an FNN can have a $w * w_i$ where w is the number of weights. With an n number of training samples, and e as the number of epochs, we get

$$O((w * w_i) * n * e)$$

Both the forward direction and the backprobgation have the same computational complexity.

4.5 Data and Experiments

Network traces that show real daily production in-bound or out-bound traffic are rare. Needless to say that traces with enough aggregated data to represent a wider sample is even rarer. The aggregation property is essential here because it is easy to

have large unpredictable fluctuation on smaller network/user traces [147]. For that we chose two data sets, one is the GÊANT [109] (trace between zone 12 and 13) which has been widely used in network traffic research. Another is a trace collected in the University of Victoria. Both traces were a collection of 15 minutes of bandwidth use at a gateway router. The data has been cleaned out of outliers after labeling the minutes, hours, days of the week, days of the month, and months.

The data in the UVic dataset was normalized to the range of the data in GÊANT dataset. This was done to be able to compare the loss with similar scaling between the two datasets. In all models, the data was trained on three months and a half and tested on half a month.

It has been the norm to test the applicability of a model with the Mean Square Error (MSE) or Mean Absolute Error (MAE). For PI we have to take into account if the prediction was inside the prediction interval [59]. This is a delicate balance between keeping the prediction interval tight enough to lower the error from bounded intervals. Also, to keep those bands wide enough to capture more predictions. We found that dividing the average MAE of each interval by the percentage of predictions bounded by the intervals can give a good overall predictor of the model performance compared to the others.

A big aspect of study in model design is tuning of hyperparamenters. Going into the details of choosing each model parameters can take the space of a whole different work. But for our intended purpose we chose and tuned our hyperparameters to the best practices employed by each model's literature. We used grid search for models with small number of parameters (LQR, OLS, RF). We chose 0.1 as our learning rate which gave good results compared to processing intensity. For the decision tree models we chose the number of estimators to be 1000, the minimum sample leaf to be 9, and maximum depth to be 3.

For NN models we found that introducing more than one hidden layer gave a decaying gain of 1% or less in line with results in [128]. We noticed that keeping the input layer closer to the daily cycle of the data gives best results (processing vs convergence, especially for LSTM).

We are going to show the results of The GÊANT dataset first and in more detail since it is very familiar to the research community. Also, although the accuracy is different, both datasets gave relatively similar model performances. In addition, showing graphs of all experiments is space demanding with little to offer.

4.5.1 Decomposition

There are aspects of time series data that should be removed before applying modeling techniques. Mainly, seasonality and trend can affect prediction negatively. After analysis we found no trend in the data. For seasonality, the data included four months of data which was not enough to include monthly or quarterly seasonality. What's interesting was removing seasonality of the week, resulted in removing the whole predictability of the data as in figure 4.4. The figure shows the effect of subtracting one data point from the same data point the following week. This cements the correlation between the same time of day/day of the week in the weekly cycle. In simpler terms, a Monday looks more similar to other Mondays than the next Tuesday. The figure also indicates that the variance and mean are somewhat constant.

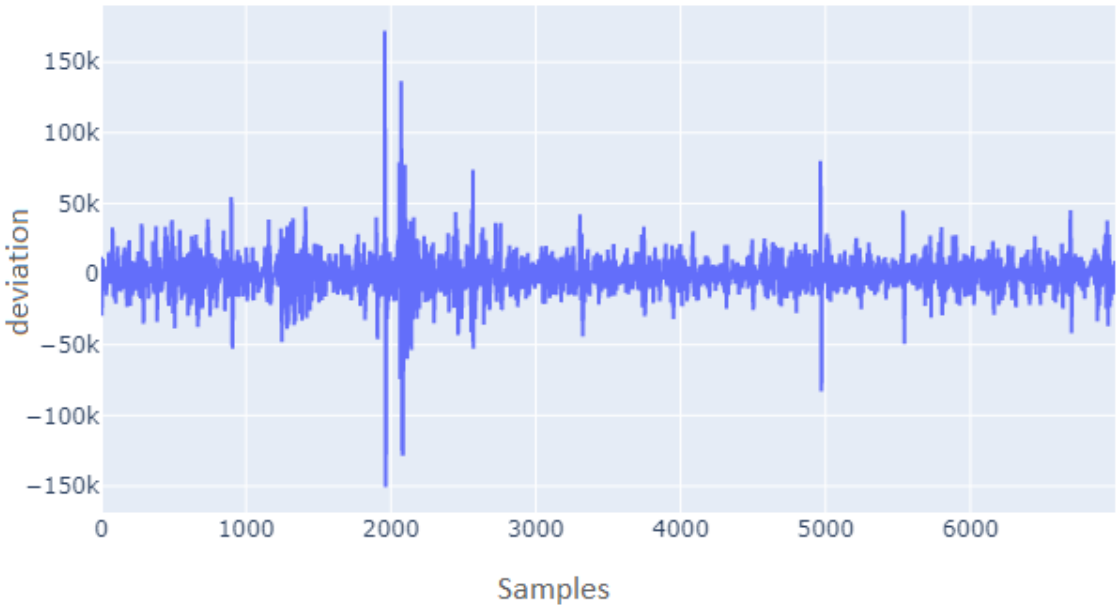
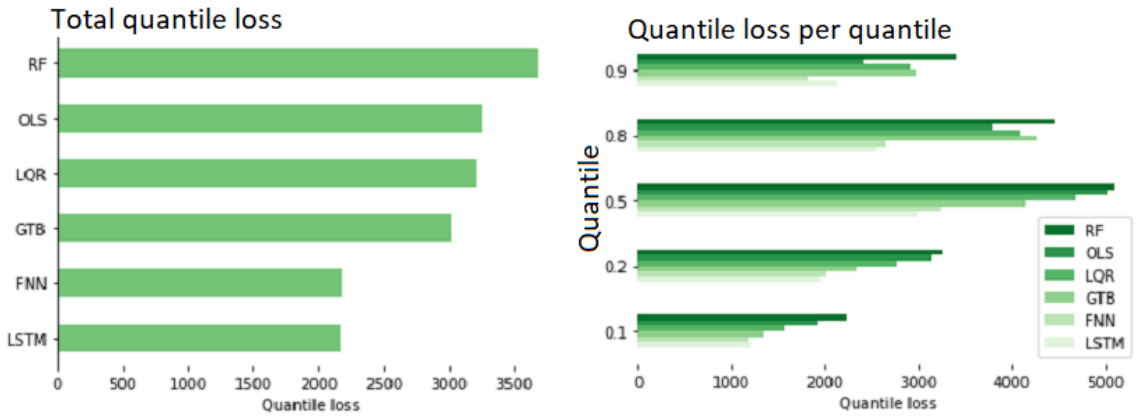


Figure 4.4: Weekly seasonality removed.

4.5.2 The Time series prediction (unsupervised)

We started experimenting by building the models without considering the influence of previous data (Lag as a feature) on the prediction of the quantiles. Although this is not the standard practice, it was interesting to see the results if the data was unsupervised. In figure 4.5a we can see the overall quantile loss of each model, while

figure 4.5b shows a detail look in to 5 quantile losses. The six graphs of figures 4.6, 4.7 and 4.8 show the same five day snippet upper and lower predicted percentiles (90% and 10%, respectively) and the value it is tested against. It is very interesting to see each model's adaptive behavior to the daily cycle and how they react to ouliers. Also, we can see how consistent are neural network approaches. On the other hand we can see the flexibility and greediness of Tree models.



(a) Average Quantile loss.

(b) Detailed Quantile loss.

Figure 4.5: Models performances.

The results in Table 4.1 show the quantile loss, the percentage captured inside the PI (from 10% to 90%), and the metric combining the loss and the captured percentage (arranged by results). Also, It shows that with unsupervised data the neural network approaches preform better. It is interesting to see that both neural networks and linear model captured 80%+ of the test values inside the chosen interval. Using the proposed metric we can see that NN models perform better and then both of the linear models. While the Tree models preformed poorly.

Table 4.1: Quantile loss and percentage captured (unsupervised)

GEANT				UVic			
Method	Quantile Loss (Kbps)	Percentage Captured	Result	Method	Quantile Loss (Kbps)	Percentage Captured	Result
FNN	2182	0.856	2550	LSTM	8839	0.867	10189
LSTM	2165	0.794	2727	OLS	10790	0.854	12630
OLS	3256	0.869	3746	LQR	12433	0.947	13128
LQR	3205	0.847	3786	RF	10578	0.748	14135
GTB	3016	0.558	5402	GTB	10187	0.695	14650
RF	3690	0.517	7136	FNN	14725	0.728	20214

Figure 4.6: Predicting PI for Linear models

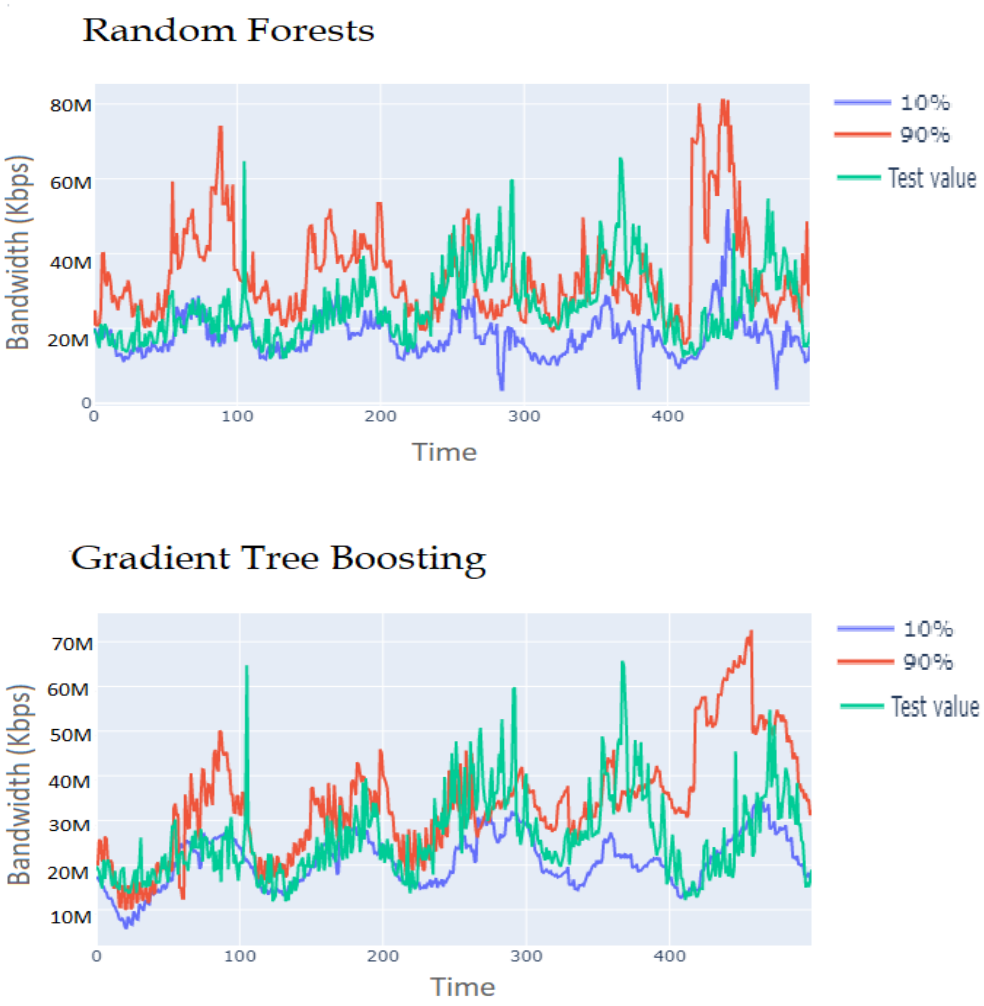


Figure 4.7: Predicting PI for Tree models

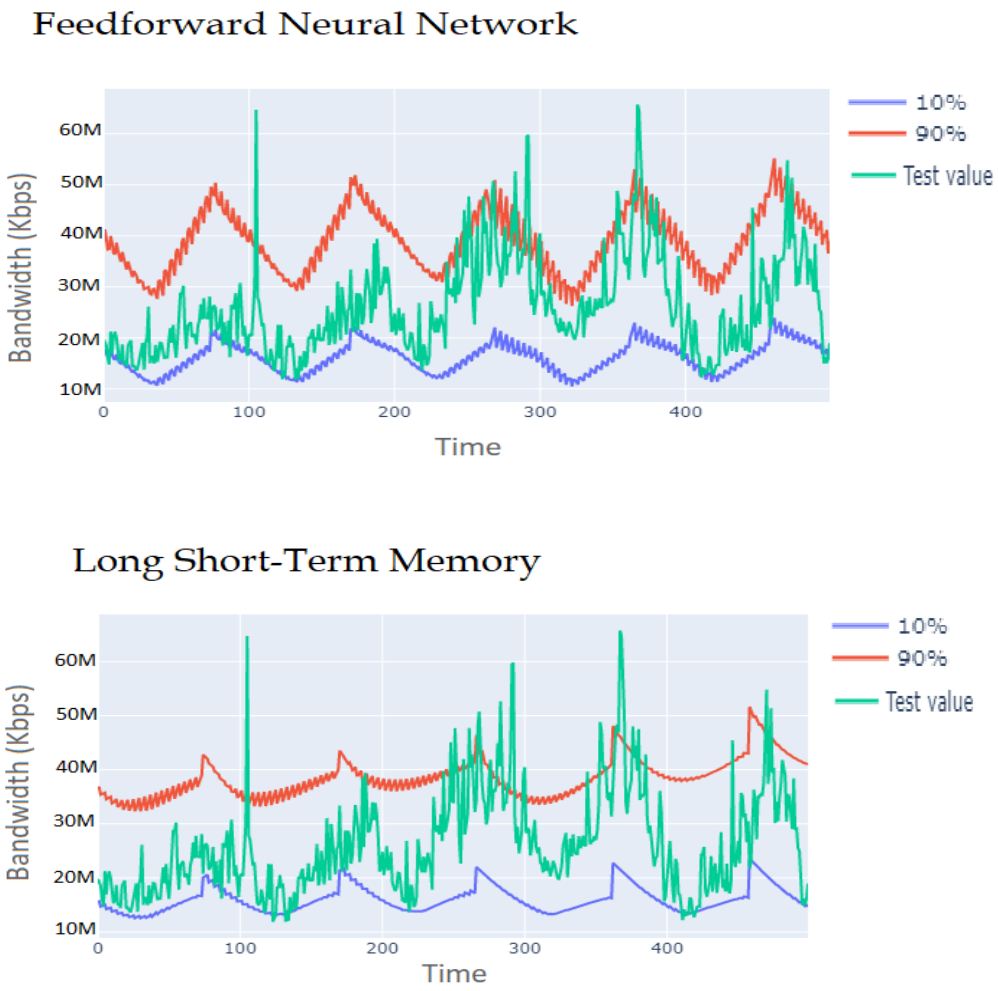
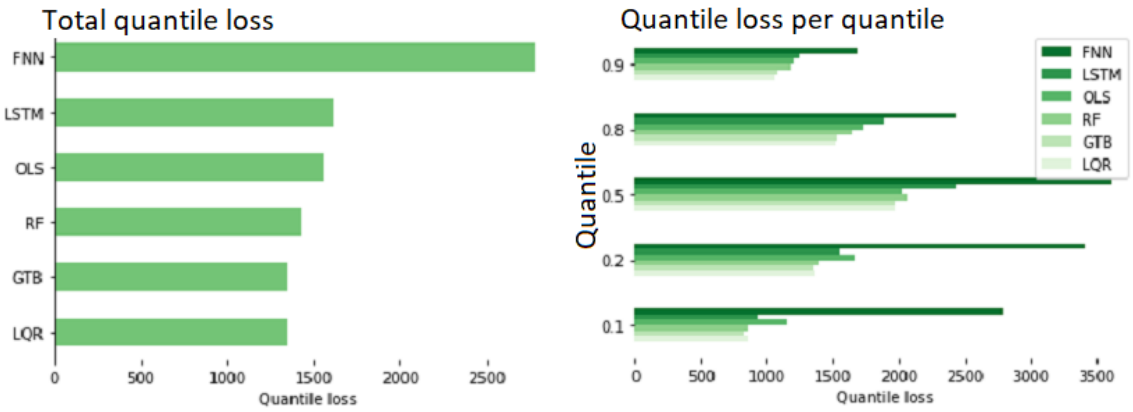


Figure 4.8: Predicting PI for Leaner models



(a) Average Quantile loss.

(b) Detailed Quantile loss.

Figure 4.9: Models performances.

4.5.3 The lag as a feature Case (supervised)

Introducing lagged data in the prediction, can be as easy as putting the previous data points as features in the learning process. Introducing more than one lag to the data doesn't improve the results much. This is in line with the work in [149]. Another similar conclusion we reached was that the convergence of the model to near its best result can be attained by much less training data (20% or less in some models). This is understandable since capturing the daily cycle with its variations only need that much of training. The supervised case is a 25% improvement on the unsupervised case.

The impact of introducing lag into the feature space has reduced the loss in all models. The prediction interval is much narrower and a bit affected by outliers (not more than one lag). All models show a capture percentage of 75% or better except FNN. For FNN the Zig-zag behavior which is normal for ML prediction, caused a fluctuation of prediction for each boundary for upto 25% (figure 4.12) (changing the batch size didn't help much).

We can see from figures 4.9a and 4.9b how the quantile loss has improved in all models. Also, The figures illustrates the difference in loss between the middle quantiles and the boundary quantiles. This in essence implies that the loss function produce better results on extreme quantiles compared to middle quantiles. From table 4.2, we can see how linear models have produced the best results both in captured percent and quantile loss. For decision trees, the improvement was very noticeable. Another interesting observation was that OLS captured percentage was the best in

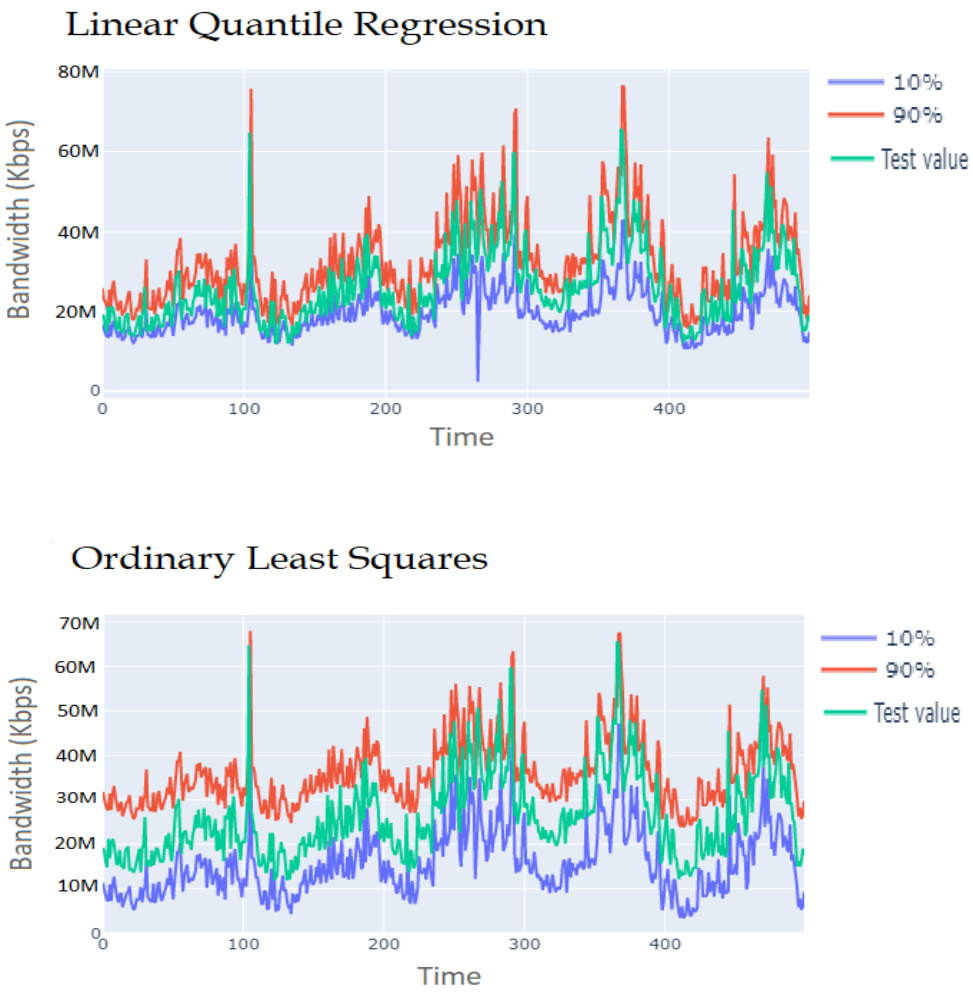


Figure 4.10: Predicting PI for Linear models

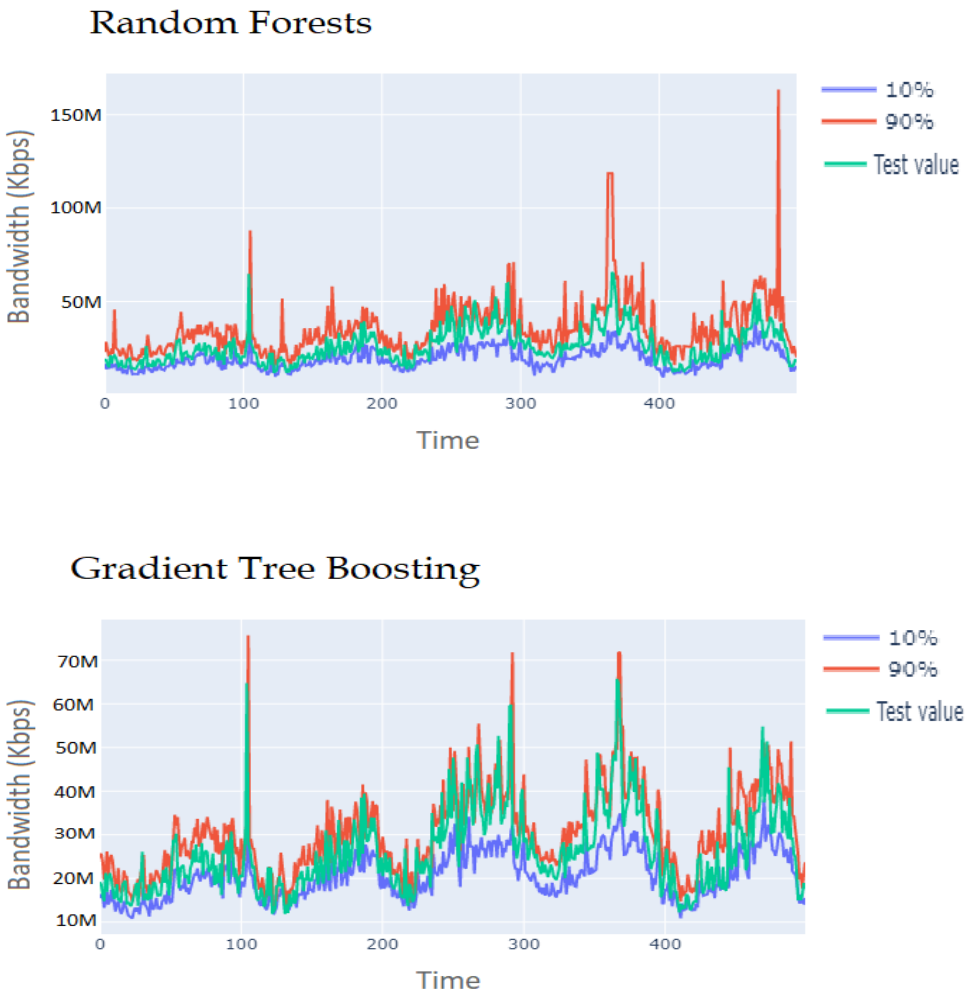


Figure 4.11: Predicting PI for Tree models

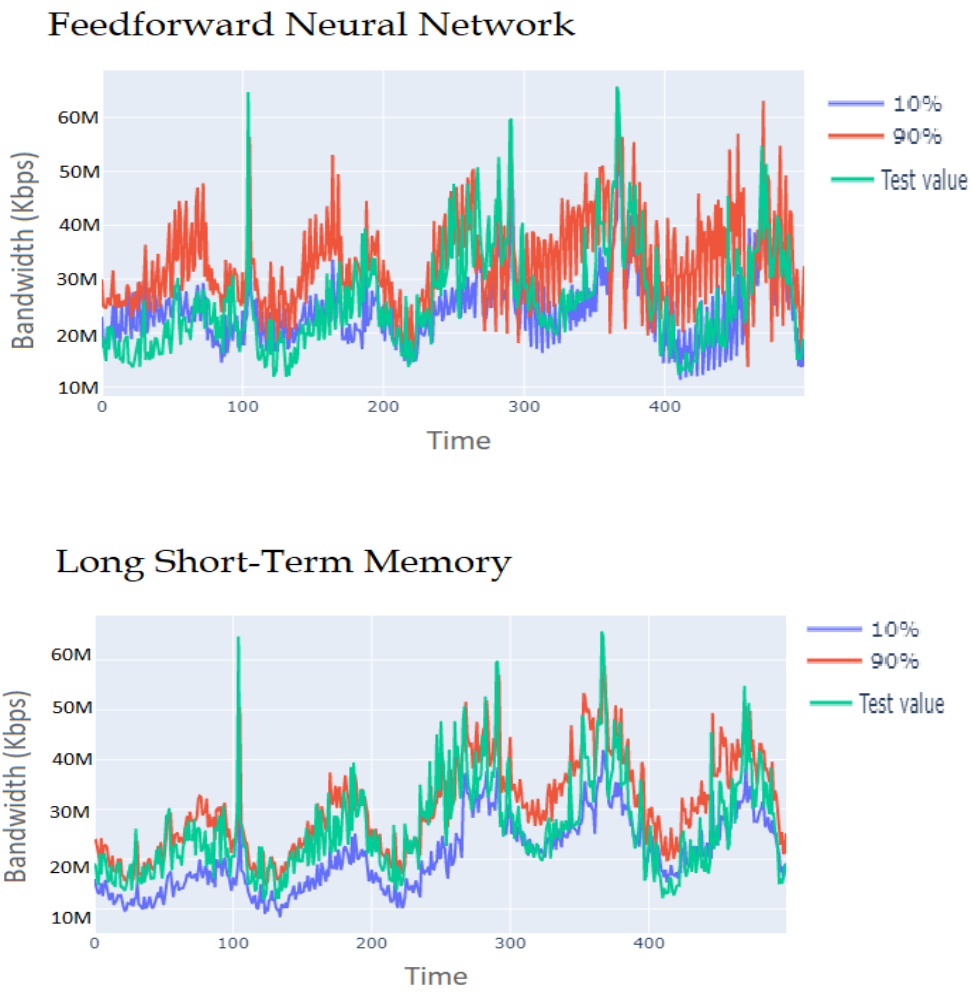


Figure 4.12: Predicting PI for Leaner models

Table 4.2: Quantile loss and percentage captured (Lagged data)

GEANT				UVic			
Method	Quantile Loss	Percentage Captured	Result	Method	Quantile Loss	Percentage Captured	Result
LQR	1351	0.80	1681	LSTM	6848	0.91	7498
OLS	1558	0.93	1673	OLS	6662	0.88	7571
RF	1429	0.825	1731	LQR	6848	0.83	8218
GTB	1353	0.75	1788	RF	7320	0.83	8784
LSTM	1611	0.74	2167	GTB	7412	0.74	10016
FNN	2787	0.38	7315	FNN	8206	0.64	12690

both datasets studied. This can be attributed to its property of minimizing the mean (quantile in our case). Also it is interesting to see that LSTM fared much better than FNN, and was best in one dataset and in line with other models in the other. Figure 4.10, 4.11 and 4.12 shows the same period shown in the previous section but using the supervised data. The figures illustrates how impactful the introduction of lag into the feature space and how close the loss function is keeping the prediction bound to the test value.

There is no doubt that there exists improvements and better parameters for each model. Also, for NN the loss function can be chosen in a way to best serve the properties of NN. But for our intended purpose this loss function was adopted to compare between these models. This also opens the door for future works to improve on various aspects of the models.

4.5.4 The UVic dataset

Although the result stated above still hold with the exception of the poor performance of the Feed forward Neural Networks, we wanted to discuss the differences separately. The nature of the UVic dataset is different where the data aggregation is shallower than that of the GEANT dataset. We can see from Figure 4.13 how chaotic the underlying data is. Also, the data is not homogeneous in the sense that there are multiple applications running in the network. A high demand for the network doesn't necessarily mean that it is a peak hour (a single user can influence the overall demand). Although such characteristics were not favorable, upon seeing the results we found that they are consistent with the other dataset conclusions and was interesting to discuss.

Figure 4.13 shows how the best preforming model (LSTM) construct the PI. It is trying to find the best boundaries around the test value. This in essence why we chose to combine two measurements. A lazy model can just produce a wide interval

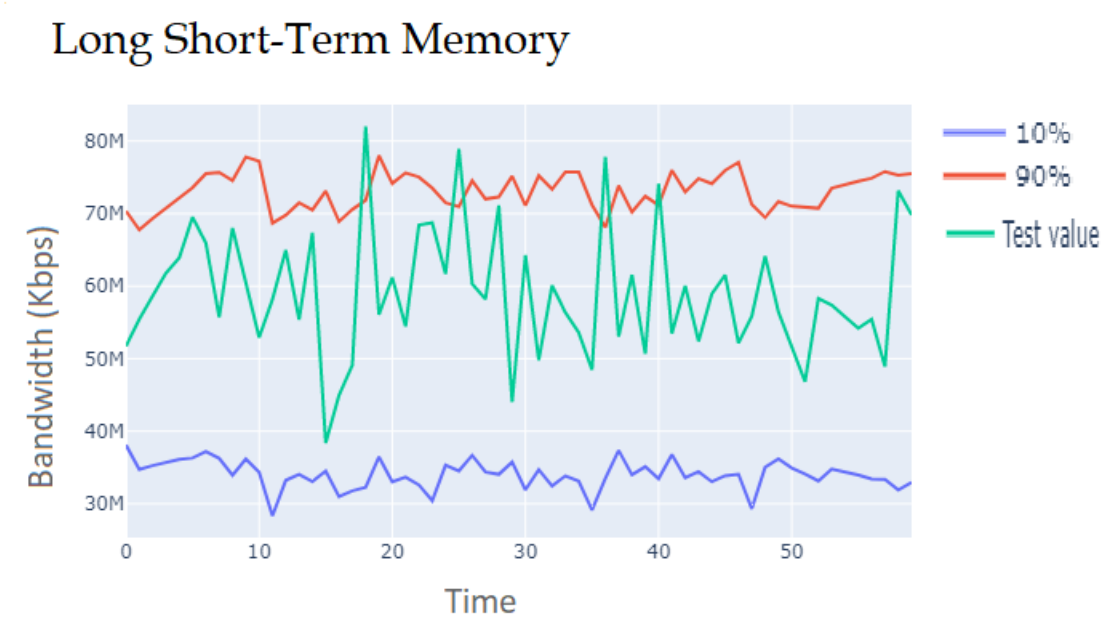


Figure 4.13: Predicting UVic dataset with LSTM.

area that can capture most of the test values but we can penalize such model with loss or vice versa. This can be seen clearly in Table 4.1 where the linear models had higher loss than tree models (3205 for LQR and 3016 for GTB) but they had greater capture rate (84.7% for LQR compared to 56% for GTB). The combined result show that linear models preformed better.

From Table.4.1 and Table.4.2, we can see that the UVic dataset have the same results as the GEANT dataset with exception of the Neural Networks models. The other models have results which are comparably the same in both datasets. FNN preformed the poorest in the UVic dataset in both cases (with supervised and unsupervised). This can be attributed to the lack of temporal dynamics in this dataset and the memoryless characteristics of the model. On the other hand, the LSTM model preformed well in all four tests especially in the UVic dataset. This solidifies the conclusion that LSTM is the best model and most diverse to use to predict time series data.

4.6 CONCLUSION

It is very clear that predicting an interval instead of exact values is a much better descriptor. We showed how we can calculate the quantile loss function and developed a combined metric consisting of loss and captured rate. This helped us optimize to the desired quantile. We then applied this loss function to six models, two linear, two decision trees, and two neural networks models. The results show the superiority of neural network models in the raw data case while linear models preformed much better when lag as a feature was introduced. The LSTM model gave good overall results across all datasets and feature variations.

Chapter 5

Predicting Data Center Resource Usage using Quantile Regression to Conserve Energy while fulfilling the Service Level Agreement

5.1 Introduction

If you ask any Data Center administrator, about the main characteristics of resource usage in their network, they will agree on two main points. First, That it is fluctuating and the second is that it is periodic. On a daily cycle, there are peak hours of the day and there are off-hours. These depend on what services the Data Center has and which hour of the day. Which part of the world does the services inside the Data Center serve, also affect the behavior and resource utilization. Figure 5.1 shows an eight day CPU average utilization (Alibaba dataset [150]). The figure shows that a daily cycle can reach upto 80% utilization and down to below 20%. Looking at this figure we can deduce that it is highly periodic and this implies that it is also predictable.

It has been suggested that 2% of the global electricity is consumed by Data Centers [151]. Although many techniques were adopted for more efficient use, it is estimated that 50% of the power used is wasted [152]. As shown below, resource utilization goes through a daily cycle where there are peak and off hours. For bigger Data Centers the behavior can be more complex since they cover a wider geographical area. But

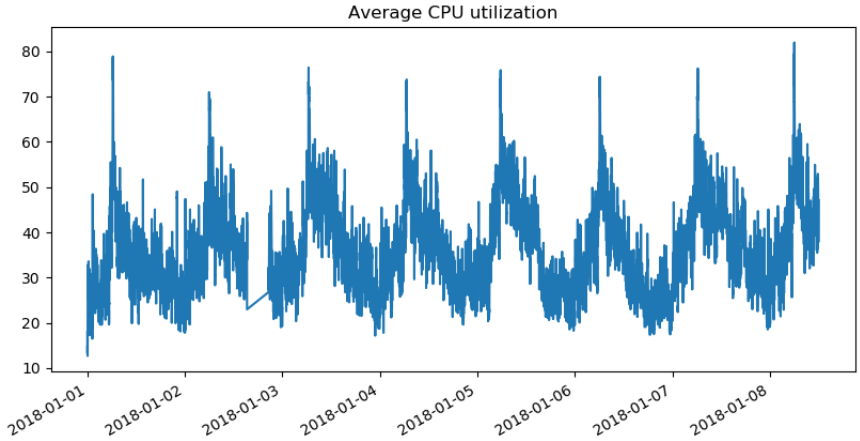


Figure 5.1: Average CPU utilization on an 8 day cycle.

we believe that the majority of Data Centers have a similar daily periodic behavior. Using stated facts, we can derive a mechanism that conserve the energy used by preemptively turning-on and off equipment.

Network functions Virtualization (NFV) has been introduced to allow for a dynamic behavior in Data Center and network deployments. The programmability of such frameworks allow for the concentration of resources in smaller number of physical server/containers. This enables better efficiency and power management. For example, at off-hours the services can be migrated to a small number of server racks which will allow non-operational racks to be shutdown through the management software. Doing that will also cut the energy cost needed to operate the servers and the cooling of that part of the Data Center. In addition, turning the machines off or putting them in less energy operational mode help increase the life cycle of hardware.

In the previous chapters we showed how we can collect information using the SDN infrastructure (*chapter*3) [153]. Then use such information to get an insight into the data and predict network traffic in (*chapter*4) [154]. In this work we will combine those ideas to study the applicability of prediction models on Data Center resources. . Here we are interested in allocating enough resources to meet the Service Level Agreement (SLA), while keeping the power consumption to a minimum. We will take lessons learned from the previous chapter and apply them to the data set of Data Center resource usage. On the contrary to the modeling setup of chapter 4, we will not use PI in the previous chapter, rather we are going to predict upper bounds on the predicted resource to fulfill the SLA.

5.2 Related Work

Managing power consumption of cloud servers is a fundamental problem in any modern cloud Data Centers. To address this problem, a broad range of techniques and approaches has been proposed through the years.

Server consolidation is an approach to enhance cloud Data Center power efficiency. This technique which occurs through virtualization, aims to increase the utilization of physical servers, hence decreasing their number and reducing the power consumption. As a matter of fact, servers even at a very low load, like 10% CPU utilization, consume around 50% of their peak power [155].Using virtual machine migration, VMs can be moved between physical machines and combine multiple workloads onto fewer servers. That will allow idle servers to be switched off.

Srikantaiah et al. [156] took the first step towards energy efficient consolidation, while providing required performance metrics. They addressed different challenges involved in the problem and applied a multi-dimensional bin-packing problem for allocating and migrating workloads to achieve energy optimality. They also studied the impact of multiple resources (like CPU and disk) utilization on the performance and energy consumption of consolidated workloads. Results showed that increase in disk utilization can be a degrading factor in server performance. Besides, there is always an optimal combination of resource utilization that minimizes energy per transaction of consolidated workload.

Chen et al.[157] further improved the server consolidation method by adding spatial awareness, in which the workload will be placed into most suitable machines by considering thermal distribution and cooling facilities. They proposed a reinforcement learning model to predict the thermal distribution resulting from different placement of incoming loads and then picking the optimal one. The results showed 13%-18% saving in cooling energy.

There is a group of studies on power management that rely on workload prediction and scheduling, in which the efficient placement of workload would reduce the power consumption in servers.

On an early work, Bradly et al. [158] developed a predictive technique for minimizing power consumption, server failure rate and client disconnection rate. The periodic behavior of e-commerce and web-servers workload make the prediction achievable. Their proposed algorithm decides to turn on or off the servers in near future considering the current and predicted demand. If the CPU utilization of any resource

exceeds a threshold, extra resources would be added. On the other hand, if the utilization of all servers are lower than the threshold and adding the load of one server won't make them over-utilized, then at least one server can be switched off. The implementation of the algorithm on the workload data from production servers showed 20% saving in energy with having less than 1% of unmet demand.

In another attempt, Goiri et al. [159] presented a dynamic job scheduling method to enable consolidation of workloads into smaller number of nodes, while maintaining quality of service. The power-aware scheduling algorithm gets different parameters like resource requirements of VMs, available resources offered by servers, servers energy consumption and SLA constraints as input. The algorithm then assigns different scores for each possibility of server-VM allocation and tries to find the best possible schedule. They achieved a power consumption reduction of 15% with compared to typical policies.

Machine learning methods have been excessively used for demand forecasting and energy management in many works [160, 161, 162]. Farahnakian et al. [160] proposed a CPU usage prediction method based on history of machine usage, since CPU is the main consumer of energy in servers. They made use of prediction model to identify the over and under-utilized hosts and minimize the energy consumption by migrating VMs and changing the servers to sleep mode from idle mode. In a similar study Hsieh et al. [161] did short-term future CPU prediction of hosts for the purpose of economizing energy in cloud Data Center. They used the combination of current and future utilization as a measure for servers workload. Gray-Markov prediction model have been used and applied on real-world workload trace.

Dabbagh et al. [162] developed a framework for prediction of future VM request and resource requirements using machine learning models. It consists of three components: data clustering with k-means, workload prediction using stochastic Wiener filter, and power management. They used Google cluster trace for evaluation of their approach, considering only two types of resources, CPU and memory. The results demonstrated that the energy management system achieved near-optimal energy efficiency.

Compared to previous works we have evaluated the prediction models using multivariate feature space instead of uni-variate feature space. The feature space we use for training consists of minutes, hour, day of week, and day of month. These features help the models capture the periodic behavior of the data. Also, we examine if the models fulfill the services SLAs.

5.3 Quantiles and Service Level Agreement

Machine learning has been advancing rapidly in the last decade due to the increase in computational power and the proliferation of data. Researchers and Data Scientists have created an ever increasing number of models. Each model can claim its superiority in fitting certain data or description of a problem. Some are very well suited for a class of problems. One thing that is certain in this field is that better models are always replacing older ones. This is in part because of the availability of new descriptive data, bigger feature space, better tuning of model hyper-parameter, etc. That's why in this paper we evaluate our approach on several prediction models.

From the previous chapter, we can see that producing a quantile out of the studied data, is an attempt to put a line through the data. This line is separating the data into two parts which are α and $\alpha - 1$. For example, a 90% quantile bound have 10% of the observed data above the quantile while 90% are under this quantile. The quantile is as good of a descriptor of the underlying data when the model is implemented.

Since Data Centers serve multiple clients with different service priorities and needs, Service Level Agreements (SLA) are established between the client and the service provider. Most SLAs have strict availability requirements. This with other factors cause service providers to over provision their resources to the point where only 10% of the resources are utilized.

Quantiles are by definition a percentage guarantee that a percent of the observed data lie below the quantile. With this we can be rest assured that it will satisfy the needed SLA (in a perfect model). It should be noted that the observed data can't represent all possible data space. Also, model accuracy can vary widely and hidden model features can affect the outcomes. Taking those factors into account, in addition to others that we will mention in next section, is very important when implementing resource allocation prediction.

5.4 Dynamic Allocation of Resources

An ideal server can waste up to 70% of the power consumed by a fully utilized server [163]. Using programmable NFVs to dynamically turn-on or turn-off servers depending on the predicted load, we can introduce better power management and efficiency. Consolidating servers to smaller number of containers/racks has been used widely to conserve power in Data Centers [164]. Using prediction models the controller

can migrating virtual servers and services to the needed number of resources. Then we can dynamically turn-off idle devices. The conservation of power process need not be a completely shut down of those devices. Some devices can be set to low power consuming modes.

One aspect we should consider is to guarantee that the SLAs are not violated during this operation. Introducing a buffer with the prediction will mitigate the bursts of usage demands.

5.5 Prediction Models

To cover a wide range of models we looked into the applicability of the most used linear, and non-linear prediction models. We did not consider ARIMA and its variations because those models are built for univariate data. We would like to take other features into consideration (time of day, day of the week).

Using the results from the previous chapter, we narrowed the scope of the prediction models to only three. We chose the best preforming ones. They are OLS, GTB, and LSTM. We chose those because they represent each category of prediction. Also, the differences between each model and the other model in the same category was negligible.

5.6 Data and Experiments

Data Center usage data sets are rare, especially those that show resource usage per unit time frame. In this work we used the Alibaba dataset [150] (we did not consider the 2020 Alibaba dataset because it is a GPU dataset. Also, the 2021 dataset which has only 12 hours of data). The dataset includes an eight day collection of various matrices on resources used. These include CPU, Memory, Disk utilization of machines (also, containers). We chose these three resources as our prediction target. The data was sanitized out of outliers. Since we would like to capture the daily cycle, we created three new features. These are minute of the hour, hour of the day, day of the week, day of the month, month. These have been extracted out of the none periodic absolute time given (Unix time). Each resource was predicted separately.

The applicability of a model has been typically measured with the Mean Square Error (MSE) or Mean Absolute Error (MAE). In our study we value the satisfaction of the SLA more. Also, the maximization of resource usage is an important metric.

But the most important metric is how much energy was saved using the prediction mechanism. This will introduce a delicate balance between trying to over allocate resources to guarantee SLA, and to under-allocated resources to save on energy.

Similar to what we've done in chapter 4, the model parameters were chosen according to the best practices in the field. For the models learning rate we chose 0.1. For the tree model, we chose 2000, 9, 3 as the number of estimators, the minimum sample leaf, and maximum depth, respectively. We chose the number of input nodes as 100.

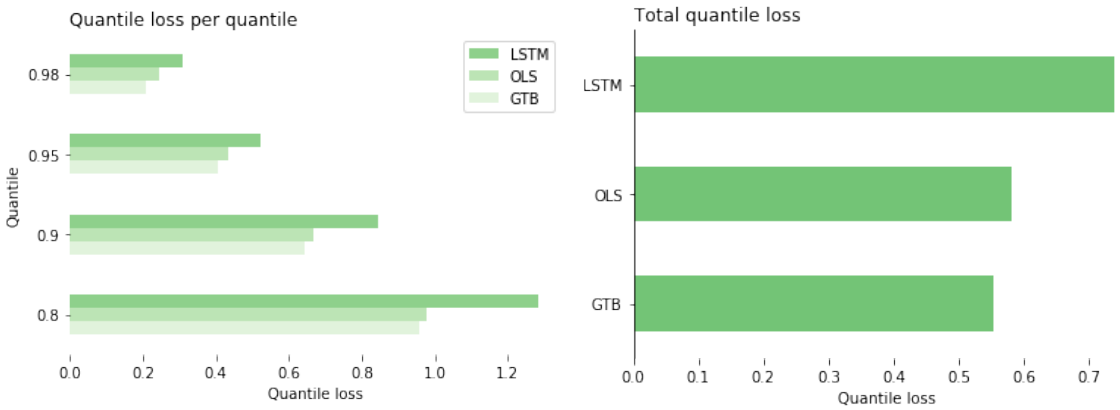
We are going to first show the prediction results of the CPU data. This will give us insight on the efficiency and the guarantee of SLAs. Then we will show the prediction results of the other resources. Finally, we will discuss improvements and suggestions for the employed techniques.

5.6.1 Predicting Upper Quantiles of the CPU Utilization

From figure 5.1 we can see that the CPU data is very periodic which implies that this cycle is predictable. In our experiments we tested a good number of models but only the results of the best performing ones are shown. We don't claim that these are the optimal prediction results. There exists better models and better parameters but using them will show how prediction of resources is possible and useful.

Since we are trying to fulfill the SLA we chose to predict four quantiles that are usually used. These are the 80%, 90%, 95%, and 98% quantiles. We trained the data on the first seven days then tested it on the eighth day. Although the data doesn't represent a wider range we believe that the results will be similar. The prediction is done for the next minute aggregate average utilization of the CPU. Figure 5.3 shows the quantiles predicted compared to the test value. The figure shows how each predicted quantile builds an upper bound that tries to capture a quantile percentage or more below it.

The quantile loss for each model is represented in figure 5.2a and figure 5.2b. These show the total quantile loss and the loss per quantile, respectivly. Since the models are predicting the utilization, the loss is a percentage loss out of 100. We see that the linear model has the least loss compared to the other models. Although there are differences in prediction between the implemented models, the loss in general is very small to a point were we can say that all are great predictors and are very similar.



(a) Detailed Quantile loss.

(b) Average Quantile loss

Figure 5.2: Models performances.

5.6.2 Energy predicted and SLA

Since resources in Data Centers are turned on/off as discrete units, it is important for us to treat the prediction as discrete as well. For simplicity, we imagined each one percent of the utilization (out of 100) to be one resource. This means that predictions should be rounded-up to represent resource prediction. This assumption is very representative of a medium size Data Center. Figure 5.4 shows the 90% quantile prediction of OLS rounded (in bars) to the test value. Since it is the prediction of the 90% quantile, we can expect that 90% of the test data is below this boundary and a 10% above. As a matter of fact, the number of test data that went above this boundary for this particular example were 7.5% (due to rounding up). This in essence satisfies the SLA demanded of the prediction. Table 5.1 shows SLA compliance for all prediction models with the quantiles for the CPU dataset.

Table 5.1: SLA compliance

	Quantiles			
	80%	90%	95%	98%
OLS	87.43	92.48	95.49	97.47
GTB	78.71	89.44	93.94	97.56
LSTM	84.22	89.35	94.49	97.30

Although there is a small deviation from the target value, we can see that the prediction is very close. We attribute this error to the small dataset and the inherent nature of prediction in general.

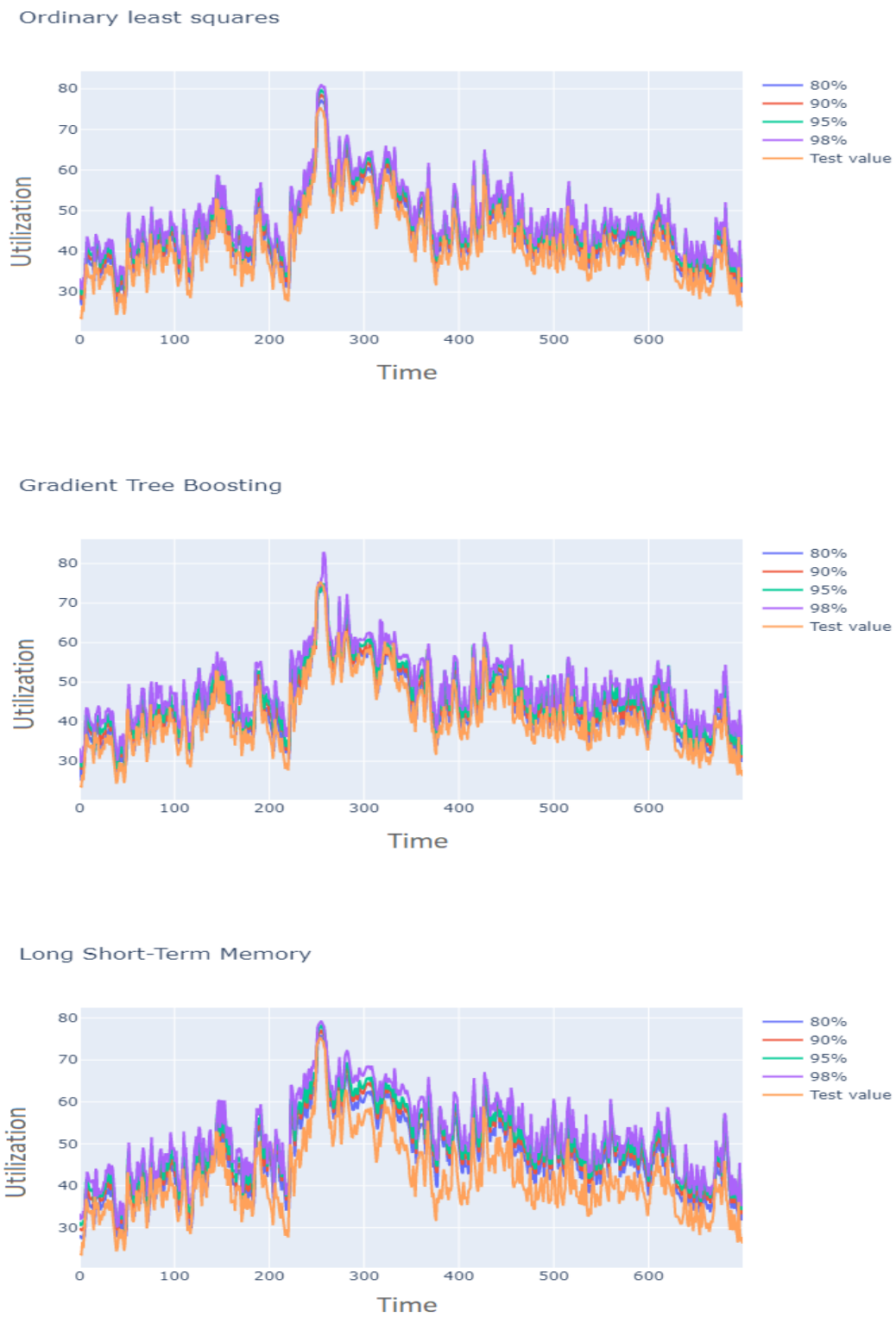


Figure 5.3: CPU quantile prediction

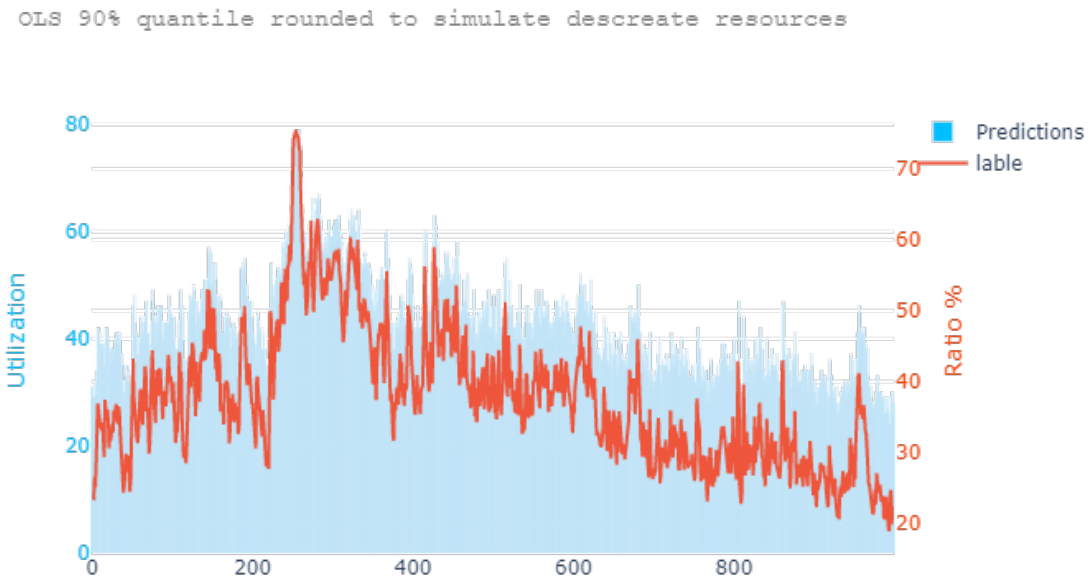


Figure 5.4: Prediction Rounded compared to tested.

5.6.3 Buffer and power saved

It has been shown [165] that CPU utilization is linearly correlated with power consumption. From that we can assume that any conserved resources (CPU) can be translated to a direct conservation of energy. Saying that our results and conclusions are a prove that dynamic allocation of resources is undermined.

From figure 5.4 we can see how much we can save on energy if we adopt the prediction scheme. The exact average predicted cut is 56% (average unutilized resources). The average extra predicted resources above the test data is 7.5% which means the resources are on average utilizing 92.5% of the predicted CPU resources. Since we are looking for compliance rather than accuracy, we can introduce a buffer. The buffer can act like an extra guarantee and to absorb bursts in demand [166] (an inherent Poisson characteristic of Data Center traffic and usage). If we assume a 20% buffer to CPU allocation we get a 36% more efficient energy use (save) and a compliance of 100% (we got no test values that went above the prediction plus the buffer in the CPU simulation).

5.6.4 Other resources

In table 5.2 we summarize the results of the memory and disk I/O model quantile prediction. We noticed that the memory data has less predictability (absence of periodic behavior) and thus the efficiency was the least among the three resources studied. We attribute this to the fact that operating systems can address more memory than that allocated physically. This and other techniques employed like paging can give a poor representation of the memory used. Also, processes can stay ideal in memory without being removed or used for arbitrary time periods.

Figure 5.5 shows the memory and disk prediction using the three models used. We can see tight bounds on all quantiles. Table 5.2 shows the average quantile loss of all models ran through the three resources studied. The table gives a good overlook on the performance of the models.

Table 5.2: Memory and Disk SLA compliance

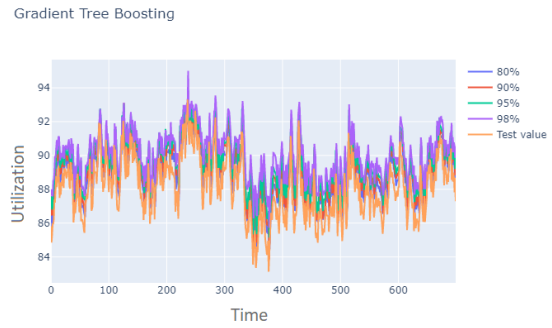
	Quantiles							
	Memory				Disk I/O			
	80%	90%	95%	98%	80%	90%	95%	98%
OLS	93	97.1	98.6	99.4	91.2	94.7	96.2	97.4
GTB	91.3	95.2	97.3	98.7	86.4	93.2	96.7	98.4
LSTM	94.7	97	98	98.3	57.1	77.3	86.1	98.3

5.7 Discussion

As we can see, the differences between the selected prediction models are very small. The results show that the predictability of the resources usage are high. As if choosing any prediction model didn't matter much (we have tried others and they gave nearly similar results). Consequently, using such techniques has an impact on cutting energy and cost. The next challenge to such an implementation is to consolidate used resources in fewer physical devices. This will enable resources to be concentrated for more efficient cooling and maintenance operations. Also, since on average 56% of the resources are off, we can assume that their lifetimes are doubled. Of course, in a real implementation other factors should be taken into consideration like, power-on lag times, buffer for sudden spikes in demand, resources allocation optimization, and fault tolerance of management of resources and cooling.

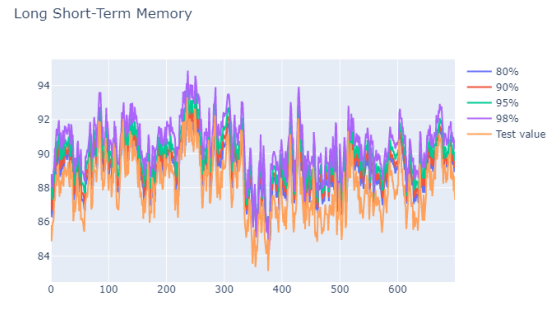
(a) Memory OLS

(b) Disk OLS



(c) Memory GTB

(d) Disk GTB



(e) Memory LSTM

(f) Disk LSTM

Figure 5.5: Memory and Disk quantile prediction

5.8 CONCLUSION

In this work, we showed that resources usage in Data Center are predictable. It was not meant to produce accurate prediction rather to investigate the impact of resource usage prediction on Data Center power saving. Using quantile regression and hand picked Machine Learning models, we were able to use enough resource to cut down on the energy needed up to 56%. In the mean time, our simulations show that with careful implementation SLAs can be easily fulfilled.

Chapter 6

Conclusion and Future work

In this work we went through the journey of exploring how we can translate the capabilities of central management in Data Centers into effective and beneficial actions. We began by describing SDN, its advantages over traditional networks, and how it is implemented with the Openflow protocol. Also, we discussed the research frontier in categories that was explored by researchers using SDN. The next part of the work dealt with using SDN to collect information from the network. We showed that SDN can be used to collect information and data on the network. Since SDN is a programmable infrastructure, there are many approaches to solve a certain task. The task we tackled was measuring delay between two devices and we implemented four methods for this purpose. The results show that using SDN to make accurate measurements is feasible. At the end of this part we shared the problems we faced with Openflow and our suggestions to fix them.

Since we established that SDN is a powerful tool of collecting data, the next logical step is to get meaning out of that data. We took data that has been collected previously to train machine learning models for the purpose of predicting the next time frame. Instead of predicting a certain number, we preferred the approach of predicting an interval. Since this approach takes the interval capture percent and the prediction loss into consideration, a new metric was created to evaluate and compare the models. We showed that bandwidth usage is highly predictable.

The main motivation of the work is quality of service. Even though this is a broad subject, the whole idea is to provide better services. Taking the prediction results and tailoring the infrastructure to better fit the services, will produce better network and Data Center designs. For example, after showing that the network bandwidth usage is predictable, we can allocate enough bandwidth to fulfill the services need. This will

cut cost and allow for better sharing of resources. From that we looked at the biggest consumer of cost and energy which is Data Centers. We employed the same prediction models to establish upper bounds on the prediction. Since services in the Data Center give priority to SLAs, a connection between quantiles and SLAs was established. The results show that the differences between the models are not significant. What is significant is the percentage of waste and consequently the potential save in energy and cost.

Looking back at the whole work we can extend the research in each of the three main topics. First, in SDN there are many efforts to research better ways to collect data from the network. Changes to the Openflow protocol can be implemented to make measurements much easier with lighter load on the network. Introducing logic into Openflow is a very interesting field that the community is looking deeper into. The second topic research area is more in the predictability of network traffic than it is in machine learning. This topic can be expanded by finding better prediction models, covering more data-sets, and including different time frames. The third topic is similar in how the research can be extended. The different modes of operation of devices can be looked at to introduce better performance and agility.

As we come to the conclusion of this work, we would like to stress the objectives of the work. We began with the idea of quality of service and how does the SDN infrastructure accommodate it. This led us to using SDN to collect the needed information upon which we gain insight on patterns in the data. Then we took it further to make changes to the network and Data Center. The end result is an efficient infrastructure which cuts on cost and consumes less energy.

Bibliography

[1] ITU. [Online]. http://www.itu.int/en/mediacentre/Pages/2016-PR35.aspx (2016, September).

[2] Cecilia kang. (2016, June) New York Times. https://www.nytimes.com/2016/06/15/technology/net-neutrality-fcc-appeals-court-ruling.html?_r=0.

[3] M. Fedor, M. Schoffstall, and J. Davin J. Case, A Simple Network Management Protocol (SNMP), 1990.

[4] Jennifer Rexford and Ellen Zegura Nick Feamster, The Road to SDN: An Intellectual History of Programmable networks, ACM Queue, vol. The Extensibility Working Group, December 2013.

[5] A. A. Lazar, J.-F. Huard, K. Lim, S. Mahjoub, L.-F. Pau, M. Suzuki, S. Torstensson, W. Wang, and S. Weinstein J. Biswas, "The ieee p1520 standards initiative for programmable network interfaces. ," Communications Magazine, IEEE, vol. 36(10), pp. 64–70, 1998.

[6] K. L. Calvert, S. L. Murphy, H. K. Orman, and L. L J. M. Smith, "Activating networks: A progress report.," Journal Computer, vol. 32, no. 4, pp. 32-41, Apr 1999.

[7] S. Rooney, L. Leslie, and S. Crosby. J. E. van der Merwe, "The Tempest: A practical framework for network programmability ," IEEE Network, vol. 12, no. 3, pp. 20–28, 1998.

[8] N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe M. Caesar, "Design and implementation of a routing control platform," in 2nd USENIX NSDI, Boston, MA, 2005.

[9] E. Kohler, A. Ghosh, O. Hodson, and P. Radoslavov M. Handley, "Designing extensible IP router software," in Networked Systems Design and Implementation, 2005.

[10] T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo T. V. Lakshman, "The SoftRouter Architecture. ," in 3nd ACM Workshop on Hot Topics in Networks (Hotnets-III), San Diego, CA, 2004.

[11] BIRD Internet routing daemon. [Online]. http://bird.network.cz/

[12] D. Pei, T. Scholl, A. Shaikh, A. Snoeren, and J. van der Merwe P. Verkaik, "Wresting control from BGP: Scalable fine-grained route control.," in USENIX Annual Technical Conference, 2007.

[13] T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner N. McKeown, "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Computer Communications Review, Apr 2008.

[14] T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker N. Gude, "NOX: Towards an operating system for networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 3, pp. 105–110, 2008.

[15] Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart and Amin Vahdat Sushant Jain, "B4: Experience with a Globally-Deployed," ACM SIGCOMM Computer Communication Review, vol. 43, no. 3, pp. 3-14, 2013.

[16] The Economist. [Online]. `http://www.economist.com/news/business/21568435-software-defined-networking-inspiring-hope-and-hype-network-effect`.

[17] R., M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Sherwood, "Carving research slices out of your production networks with OpenFlow," SIGCOMM Comput. Commun, vol. 40, no. 1, pp. 129-130, 2010.

[18] J. Reich, N. Foster, J. Rexford, and D. Walker C. Monsanto, "Composng Software Defined Networks," in Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, Berke- ley, CA, USA, 2013, pp. 1-14.

[19] N., R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Foster, "Frenetic: A network programming language," SIGPLAN, vol. 46, no. 9, pp. 279-291, 2011.

[20] H. Kim, and N. Feamster A. Voellmy, "Procera: A Language for High- Level Reactive Network Control," in in Proceedings of the First Workshop on Hot topics in Software Defined Networks, New York, NY, USA, 2012, pp. 43-48.

[21] S. Sundaresan, M. Chetty, N. Feamster, and W. K. Edwards H. Kim, "Communicating with caps: Managing Usage Caps in Home Networks," in in Proceedings of the ACM SIGCOMM, New York, NY, USA, 2011, pp. 470-471.

[22] J. Pettit, K. Amidon, and M. Casado B. Pfaff, "Extending Networking into the Virtualization Layer," in Proceedings of ACM SIGCOMM HotNets, ACM, 2009.

[23] (https://wiki.opendaylight.org/view/OpenStack_and_OpenDaylight

[24] (2017, May) Opendaylight. [Online]. http://www.opendaylight.org/

[25] (2017, May) OpenStack. [Online]. https://www.openstack.org/software/

[26] Neutron. [Online]. https://wiki.openstack.org/wiki/Neutron

[27] W. Zhou, M. Caesar, P. Godfrey A. Khurshid, "Veriflow: verifying network-wide invariants in real time," in Proceedings of the First Workshop on Hot Topics in Software Defined Networks, 2012, pp. 49-54.

[28] D. Levin, S. Seetharaman, A. Feldmann A. Wundsam, "Ofrewind: enabling record and replay troubleshooting for networks," in Proceedings of Usenix Annual Technical Conference, 2011.

[29] S. Al-Haj E. Al-Shaer, "Flowchecker: configuration analysis and verification of federated openflow infrastructures," in Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration, SafeConfig'10, 2010, pp. 37-44.

[30] W. Marrero, A. El-Atawy, K. ElBadawi E. Al-Shaer, "Network configuration in a box: towards end-to-end verification of network reachability and security," in 17th IEEE International Conference on Network Protocols, 2009, pp. 123-132.

[31] D. Venzano, P. Peresini, D. Kostic, J. Rexford M. Canini, "A nice way to test openflow applications," in Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, 2012.

[32] B. Heller, V. Jeyakumar, D. Mazières, N. McKeown N. Handigol, "Where is the debugger for my software-defined network?," in Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN'12, 2012.

[33] L. Jose, R. Miao M. Yu, "Software defined traffic measurement with opensketch," in opensketch, in: Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, 2013, pp. 29-42.

[34] T. Kwon, C. Dixon, W. Felter, and J. Carter J. Suh, "Opensample: A low-latency, sampling-based measurement platform for sdn," IBM, 2014.

[35] M. Ghobadi, Y. Ganjali A. Tootoonchian, "Opentm: traffic matrix estimator for openflow networks," in Proceedings of the 11th International Conference on Passive and Active Measurement, 2010, pp. 201-210.

[36] Y.Mundada,M. B. Tariq, and N. Feamster A. Ramachandran, "Securing Enterprise Networks Using Traffic Tainting," tech. rep, Aug 2009.

[37] P. Porras, V. Yegneswaran, and M. Fong S. Shin, "Fresco: Modular composable security services for software-defined networks," in Network and Distributed Systems Security Symposium, 2013.

[38] J. Khalid, and S. A. Khayam S. A. Mehdi, "Revisiting Traffic Anomaly Detection using Software Defined Networking," Recent Advances in Intrusion Detection (RAID), pp. 1-20, 2011.

[39] S. H. Yeganeh and Y. Ganjali, "Kandoo : A Framework for Efficient and Scalable Offloading of Control Applications," in Proceedings of the first workshop on Hot topics in software defined networks HotSDN' 12, 2012, pp. 19-24.

[40] V. Yegneswaran, P. Porras, and G. Gu S. Shin, "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks," in Proceedings of the 2013 ACM Conference on Computer and Communications Security, 2013.

[41] S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat M. AlFares, "Hedera: Dynamic Flow Scheduling for Data Center Networks," in Proc. 7th USENIX Symposium on Networks Systems Design and Implemen. NSDI'10, San Jose, 2010, pp. 19-19.

[42] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee A. R. Curtis, ""DevoFlow: scaling flow management for high- performance networks," ACM SIGCOMM Computer Communications Review, vol. 41, no. 4, pp. 254–265, 2011.

[43] W. Kim, and P. Yalagandula A. R. Curtis, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detec- tion," in Proc. 30th IEEE Intternational Confrence Computer Communications IEEE INFO- COM 2011, Shanghai, China, 2011, pp. 162-163.

[44] G.-M. Muntean, and K. Katrinis R. Trestian, "MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow," in IFIP/IEEE Intternational Sympimposium on Integrated Networks Management IM 2013, Ghent, Belgium, 2013, pp. 904-907.

[45] Z. A. Qazi et al, "Application-awareness in SDN," ACMSIGCOMM Computer Communication, vol. 43, no. 4, pp. 487–488, 2013.

[46] S. Kukliński, W. Kujawa, and M. Ulaski K. T. Dinh, "MSDN- TE: Multipath Based Traffic Engineering for SDN," in Intelligent Information and Database Systems. Asian Conference on Intelligent Information and Database Systems, 2016, pp. 630–639.

[47] Armitage, Grenville, et al. "Household Internet and the 'need for speed': Evaluating the impact of increasingly online lifestyles and the Internet of Things." Transport 500 (2017): B3.

[48] L. Skorin-Kapov, M. Matijasevic, "Analysis of QoS requirements for e-health services and mapping to evolved packet system QoS classes", Int. J. Telemed. Appl., vol. 2010, pp. 1-18, Jan. 2010.

[49] Yamaki, Hirofumi, Michael P. Wellman, and Toru Ishida. "A market-based approach to allocating QoS for multimedia applications." Proceedings of the Second International Conference on Multi-Agent Systems, ICMAS. Vol. 96. 1996.

[50] Oida, Kazumasa, and Masatoshi Sekido. "An agent-based routing system for QoS guarantees." IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028). Vol. 3. IEEE, 1999.

[51] de Meer, P., et al. "Programmable agents for flexible QoS management in IP networks." IEEE Journal on Selected Areas in Communications 18.2 (2000): 256-267

[52] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband Internet traffic," in Proc. ACM IMC, 2009.

[53] A. Mukherjee, "On the dynamics and significance of low frequency components of Internet load," Internetworking: Res. and Experience, vol. 5, pp. 163–205, Dec. 1994.

[54] V. Panson. End-t+cnd internet packet dynamics. 1EEUAC.W Tmnrocriom om Networking,7(3):277-292. June 1999.

[55] H. Jiang and C. Dovrolis. Passive estimation of TCP round-trip times. CCR, 32(3):75–88, 2002.

[56] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson. Design Principles for Accurate Passive Measurement. In Proceedings Passive and Active Measurements (PAM) workshop, Apr. 2000.

[57] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson. Design Principles for Accurate Passive Measurement. In Proceedings Passive and Active Measurements (PAM) workshop, Apr. 2000.

[58] Ding, Hao, and Michael Rabinovich. "TCP stretch acknowledgements and timestamps: findings and implications for passive RTT measurement." ACM SIGCOMM Computer Communication Review 45.3 (2015): 20-27.

[59] S. Kalidindi, M. Zekauskas: "Surveyor: An Infrastructure for Internet Performance Measurements", Proceedings of INET'99, San Jose, CA, USA, June 22-25, 1999.

[60] Paxson, Vern, Andrew K. Adams, and Matt Mathis. "Experiences with NIMI." Proceedings 2002 Symposium on Applications and the Internet (SAINT) Workshops. IEEE, 2002.

[61] Uijterwaal, Henk, and Olaf Kolkman. "Internet delay measurements using test traffic." Design Note", RIPE-158(1997).

[62] T. Zseby, "Evaluation of building blocks for passive one-way-delay measurements", Proc. Passive Active Meas. Workshop (PAM), 2001.

[63] S. Niccolini, M. Molina, F. Raspall, S. Tartarelli, "Design and implementation of a one way delay passive measurement system", Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS), vol. 1, pp. 469-482, Seoul, 2004.

[64] I. D. Graham, S. F. Donnelly, S. Martin, J. Martens, J. G. Cleary, "Nonintrusive and accurate measurement of unidirectional delay and delay variation on the Internet" 8th Internet Soc. Conf.(INET), 1998.

[65] Van Adrichem, Niels LM, Christian Doerr, and Fernando A. Kuipers. "Opennetmon: Network monitoring in openflow software-defined networks." 2014 IEEE Network Operations and Management Symposium (NOMS). IEEE, 2014.

[66] He, Qiang, and Shengbao Wang. "A low-cost measurement framework in software defined networks." International Journal of Communications, Network and System Sciences 10.5 (2017): 54-66.

[67] Peng, G.-Q.; Xue, G.; Chen, Y.-C. Network Measurement and Performance Analysis at Server Side. Future Internet 2018, 10, 67.

[68] Benson, Theophilus, Aditya Akella, and David A. Maltz. "Network traffic characteristics of data centers in the wild." Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010.

[69] Zander, Sebastian, Thuy Nguyen, and Grenville Armitage. "Automated traffic classification and application identification using machine learning." The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l. IEEE, 2005.

[70] Park, Junghun, Hsiao-Rong Tyan, and C-C. Jay Kuo. "Internet traffic classification for scalable qos provision." 2006 IEEE International Conference on Multimedia and Expo. IEEE, 2006.

[71] V. Misra, W. B. Gong, and D. Towsley. Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In Proceedings of ACM SIGCOMM, Sept. 2000.

[72] R. Morris. Scalable TCP Congestion Control. In Proceedings of IEEE INFOCOM, Apr. 2000.

[73] C. Villamizar and C.Song. High Performance TCP in ANSNET. ACM Computer Communication Review, Oct. 1994.

[74] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking, 7(4):458–473, Aug. 1999.

[75] R. Mahajan, S. Floyd, and D. Wetherall. Controlling High Bandwidth Flows at the Congested Routers. IEEE ICNP, Nov. 2001.

[76] B. Wong, A. Slivkins, E. G. Sirer, "Meridian: A lightweight network location service without virtual coordinates", ACM SIGCOMM Comput. Commun. Rev., vol. 35, no. 4, pp. 85-96, Oct. 2005.

[77] Brunekreef, Jacob, et al. "Design and analysis of dynamic leader election protocols in broadcast networks." Distributed Computing 9.4 (1996).

[78] Salonidis, Theodoros, et al. "Distributed topology construction of Bluetooth personal area networks." Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213). Vol. 3. IEEE, 2001.

[79] RFC 792 (1981 September) https://tools.ietf.org/html/rfc792

[80] Chapade, S. S., K. U. Pandey, and D. S. Bhade. "Securing cloud servers against flooding based ddos attacks." 2013 International Conference on Communication Systems and Network Technologies. IEEE, 2013.

[81] Gont, Fernando. ICMP attacks against TCP. No. RFC 5927. 2010.

[82] Cardwell, Neal, Stefan Savage, and Thomas Anderson. "Modeling TCP latency." Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064). Vol. 3. IEEE, 2000.

[83] Jacobson, Van, Robert Braden, and David Borman. TCP extensions for high performance. No. RFC 1323. 1992.

[84] Openflow 1.5.1 (march 2015) https://www.opennetworking.org/ wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[85] Mininet (2014), http://mininet.org/

[86] Open vSwitch (OVS) (2013). http://openvswitch.org/.

[87] Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." ACM SIGCOMM Computer Communication Review 44.3 (2014): 87-95.

[88] Bianchi, Giuseppe, et al. "OpenState: programming platform-independent stateful openflow applications inside the switch." ACM SIGCOMM Computer Communication Review 44.2 (2014): 44-51.

[89] Beba Project http://www.beba-project.eu/the-project (1 March 2020)

[90] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system."

[91] W. Leland, M. S. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of ethernet traffic (extended version)," IEEE/ACMTrans. Networking, vol. 2, pp. 1–15, 1994.

[92] K. Park, G. Kim, and M. Crovella, "On the effect of traffic self-similarity on network performance," In Proc. SPIE Int. Conf. Performance and Control ofNetwork Systems, Nov. 1997.

[93] V. Paxson and S. Floyd, "Wide-area traffic: The failure of Poisson modeling," IEEE/ACM Trans. Networking, vol. 3, pp. 226–244, 1995.

[94] Beran, J., Sherman, R., Taqqu, M. S., Willinger, W. (1995). Variable bitrate video traffic and long range dependence. IEEE Transactions on Communications, 43, 1566–1579.

[95] R. Cáceres, N.G. Duffield, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B. Krishnamurthy, D. Lavelle, P.P. Mishra, K.K. Ramakrishnan, J. Rexford, F. True, and J.E. van der Merwe, "Measurement and Analysis of IP Network Usage and Behavior", IEEE Communications Magazine, vol. 38, no. 5, pp. 144–151, May.

[96] K.C. Claffy, H.-W. Braun, and G.C. Polyzos. "Parameterizable methodology for internet traffic flow profiling", IEEE Journal on Selected Areas in Communications, vol. 13, no. 8, pp. 1481–1494, October 1995.

[97] Duffield, Nick, Carsten Lund, and Mikkel Thorup. "Properties and prediction of flow statistics from sampled packet streams." Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment. 2002.

[98] Duffield, Nick, Carsten Lund, and Mikkel Thorup. "Estimating flow distributions from sampled flow statistics." IEEE/ACM Transactions on Networking 13.5 (2005): 933-946.

[99] Duffield, Nick, Carsten Lund, and Mikkel Thorup. "Estimating flow distributions from sampled flow statistics." Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications. 2003.

[100] S. Bhattacharyya, C. Diot, J. Jetcheva, and N Taft. POP-level and Access-Link-Level Traffic Dynamics in a Tier-1 POP. ACM SIGCOMM Internet Measurement Workshop, November 2001.

[101] K. Papagiannaki, N. Taft, S. Bhattacharya, P. Thiran, K. Salamatian, and C. Diot, "On the feasibility of identifying elephants in internet backbone traffic. Sprint ATL Technical Report TR01-ATL-110918," Sprint Labs, November 2001.

[102] Mori, Tatsuya, et al. "Identifying elephant flows through periodically sampled packets." Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. 2004.

[103] Li, Yi, et al. "Predicting inter-data-center network traffic using elephant flow and sublink information." IEEE Transactions on Network and Service Management 13.4 (2016): 782-792.

[104] Moore, A.W., Papagiannaki, K.: Toward the accurate identification of network applications. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431, pp. 41–54. Springer, Heidelberg (2005)

[105] Zander, Sebastian, Thuy Nguyen, and Grenville Armitage. "Automated traffic classification and application identification using machine learning." The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l. IEEE, 2005.

[106] Moore, A.: Discrete content-based classification — a data set. Technical Report, Intel Research, Cambridge (2005)

[107] Moore, A., Zuev, D.: Discriminators for use in flow-based classification. Technical Report, Intel Research, Cambridge (2005)

[108] Callado, Arthur, et al. "Better network traffic identification through the independent combination of techniques." Journal of Network and Computer Applications 33.4 (2010): 433-446.

[109] https://www.geant.org/Projects/GEANT Project GN4

[110] Takahashi, Y., Aida, H., and Saito, T. (2000). ARIMA model's superiority over f-ARIMA model. In International conference on communication technology proceedings, WCC – ICCT 2000 (Vol. 1, pp. 66–69).

[111] J. Dai, J. Li, VBR MPEG Video Traffic Dynamic Prediction Based on the Modeling and Forecast of Time Series, Fifth International Joint Conference on INC, IMS and IDC, pp. 17521757. Seoul, Korea, 2009.

[112] Won, Y., Ahn, S. (2005). GOP ARIMA: Modeling the nonstationarity of VBR processes. Multimedia Systems, 10(5), 359–378.

[113] P. Cortez, M. Rio, M. Rocha, P. Sousa, Internet Traffic Forecasting using Neural Networks, International Joint Conference on Neural Networks, pp. 26352642. Vancouver, Canada, 2006.

[114] H. Feng, Y. Shu, Study on Network Traffic Prediction Techniques, International Conference on Wireless Communications, Networking and Mobile Computing, pp. 10411044. Wuhan, China, 2005.

[115] Rutka, G. (2009). Some aspects of traffic analysis used for internet traffic prediction. Electronics and Electrical Engineering Kaunas: Technologija, 5(93), 7–10.

[116] Mandelbrot, B. B., Van Ness, J. W. (1968). Fractional Brownian motions, fractional noises and applications. SIAM Review, 10, 422–437.

[117] Corradi, M., Garroppo, R.G., Giordano, S., Pagano, M. (2001). Analysis of f-ARIMA processes in the modeling of broadband traffic. ICC'01 (Vol. 3, pp. 964–968).

[118] Katris, Christos, and Sophia Daskalaki. "Comparing forecasting approaches for Internet traffic." Expert Systems with Applications 42.21 (2015): 8172-8183.

[119] Balkin, S. D., Ord, J. K. (2000). Automatic neural network modeling for univariate time series. International Journal of Forecasting, 16, 509–515.

[120] Dorffner, G. (1996). Neural networks for time series processing. Neural Network World, 4, 447–468.

[121] Frank, R. J., Davey, N., Hunt, S. P. (2001). Time series predictions and neural networks. Journal of Intelligent and Robotic Systems, 31(1–3), 91–103.

[122] Haviluddin, Haviluddin, and Rayner Alfred. "Daily network traffic prediction based on backpropagation neural network." (2014).

[123] Rutka, G., and G. Lauks. "Study on internet traffic prediction models." Elektronika ir Elektrotechnika 78.6 (2015): 47-50.

[124] Oliveira, Tiago Prado, Jamil Salem Barbar, and Alexsandro Santos Soares. "Computer network traffic prediction: a comparison between traditional and deep learning neural networks." International Journal of Big Data Intelligence 3.1 (2016): 28-37.

[125] Yi, Hongsuk, HeeJin Jung, and Sanghoon Bae. "Deep neural networks for traffic flow prediction." 2017 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE, 2017.

[126] Ma, Xiaolei, et al. "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data." Transportation Research Part C: Emerging Technologies 54 (2015): 187-197.

[127] Oliveira, Tiago Prado, Jamil Salem Barbar, and Alexsandro Santos Soares. "Computer network traffic prediction: a comparison between traditional and deep learning neural networks." International Journal of Big Data Intelligence 3.1 (2016): 28-37.

[128] Azzouni, Abdelhadi, and Guy Pujolle. "NeuTM: A neural network-based framework for traffic matrix prediction in SDN." NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2018.

[129] https://www.daxx.com/blog/development-trends/python-developer-salary-usa

[130] Shrestha, Durga L., and Dimitri P. Solomatine. "Machine learning approaches for estimation of prediction interval for the model output." Neural Networks 19.2 (2006): 225-235.

[131] Roger Koenker and Kevin F. Hallock. "Quantile Regression." Journal of Economic Perspectives—Volume 15, Number 4—Fall 2001—Pages 143–156

[132] Koenker, Roger, and Gilbert Bassett Jr. "Regression quantiles." Econometrica: journal of the Econometric Society (1978): 33-50.

[133] Choi, Seung-Whan. "The effect of outliers on regression analysis: regime type and foreign direct investment." Quarterly Journal of Political Science 4.2 (2009): 153-165.

[134] Computational Complexity Learning Algorithms https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/

[135] Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.

[136] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5.

[137] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. No. 10. New York: Springer series in statistics, 2001.

[138] Meinshausen, Nicolai. "Quantile regression forests." Journal of Machine Learning Research 7.Jun (2006): 983-999.

[139] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[140] Ogutu, Joseph O., Hans-Peter Piepho, and Torben Schulz-Streeck. "A comparison of random forests, boosting and support vector machines for genomic selection." BMC proceedings. Vol. 5. No. S3. BioMed Central, 2011.

[141] Gu, Jiuxiang, et al. "Recent advances in convolutional neural networks." Pattern Recognition 77 (2018): 354-377.

[142] Mikolov, Tomáš, et al. "Recurrent neural network based language model." Eleventh annual conference of the international speech communication association. 2010.

[143] Graves, Alex, et al. "A novel connectionist system for unconstrained handwriting recognition." IEEE transactions on pattern analysis and machine intelligence 31.5 (2008): 855-868.

[144] Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." (1999): 850-855.

[145] Felix A. Gers and Jurgen Schmidhuber, LSTM recurrent networks learn simple context free and context sensitive lan- guages, IEEE Transactions on Neural Networks , vol. 12, no. 6, pp. 13331340, 2001

[146] Lister, R and Stone J "An Empirical Study of the Time Complexity of Various Error Functions with Conjugate Gradient Back Propagation" , IEEE International Conference on Artificial Neural Networks (ICNN95), Perth, Australia, Nov 27-Dec 1, 1995.

[147] Sang, Aimin, and San-qi Li. "A predictability analysis of network traffic." Computer networks 39.4 (2002): 329-345.

[148] https://towardsdatascience.com/how-to-generate-prediction-intervals-with-scikit-learn-and-python-ab3899f992ed

[149] Zhani, Mohamed Faten, Halima Elbiaze, and Farouk Kamoun. "Analysis and Prediction of Real Network Traffic." JNW 4.9 (2009): 855-865.

[150] Alibaba production cluster data v2017, 2017, [online] Available: https://github.com/alibaba/clusterdata.

[151] Beyond PUE: Tackling IT's Wasted Terawatts Bashroush and Lawrence, 2020.

[152] United States Data Center Energy Usage Report, Arman Shehabi, Sarah Josephine Smith, Dale A Sartor, Richard E Brown, 2016.

[153] Ahmed Alutaibi, Sudhakar Ganti, "ICCNCS 2020: International Conference on Computer Networks and Communication Systems" Aug 13-14, 2020 in Venice, Italy.

[154] A. Alutaibi and S. Ganti, "Network Traffic Prediction using Quantile Regression with linear, Tree, and Deep Learning Models," 2020 IEEE 45th Conference on Local Computer Networks (LCN), 2020, pp. 421-424, doi: 10.1109/LCN48667.2020.9314779.

[155] United States Data Center Energy Usage Report, Arman Shehabi, Sarah Josephine Smith, Dale A Sartor, Richard E Brown, 2016.

[156] Srikantaiah, Shekhar, Aman Kansal, and Feng Zhao. "Energy aware consolidation for cloud computing." (2008).

[157] Chen, Hui, et al. "Spatially-aware optimization of energy consumption in consolidated data center systems." International Electronic Packaging Technical Conference and Exhibition. Vol. 44625. 2011.

[158] Bradley, David J., Richard E. Harper, and Steven W. Hunter. "Workload-based power management for parallel computer systems." IBM Journal of Research and Development 47.5.6 (2003): 703-718.

[159] Goiri, Inigo, et al. "Energy-aware scheduling in virtualized Data Centers." 2010 IEEE International Conference on Cluster Computing. IEEE, 2010.

[160] Farahnakian, Fahimeh, Pasi Liljeberg, and Juha Plosila. "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers." 2013 39th Euromicro conference on software engineering and advanced applications. IEEE, 2013.

[161] Hsieh, Sun-Yuan, et al. "Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers." Journal of Parallel and Distributed Computing 139 (2020): 99-109.

[162] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Energy-efficient cloud resource management," in Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on. IEEE, 2014, pp. 386–391

[163] C. Mastroianni, M. Meo, G. Papuzzo, Probabilistic consolidation of virtual machines in self-organizing cloud data centers, IEEE Trans. Cloud Comput. 1 (2) (2013) 215–228

[164] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, "Power management of Data Center workloads using per-core power gating," Computer Architecture Letters, vol. 8, no. 2, pp. 48–51, 2009.

[165] Fan, Xiaobo, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power provisioning for a warehouse-sized computer." ACM SIGARCH computer architecture news 35.2 (2007): 13-23.

[166] S. Mittal and Z. Zhang, "EnCache: Improving cache energy efficiency using a software-controlled profiling cache," in IEEE International Conference On Electro/Information Technology, Indianapo- lis, USA, May 2012.