2021-04-16

Deadline-aware Bulk Transfer Scheduling in Best-effort SD-WANs

Hosseini Bidi, Seyed Arshia

UNIVERSITY OF CALGARY

Deadline-aware Bulk Transfer Scheduling in Best-effort SD-WANs

by

Seyed Arshia Hosseini Bidi

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

APRIL, 2021

Abstract

Wide area networks (WANs) that connect geo-distributed datacenters enable online applications to provide a diversity of services to their users in various locations throughout the world. Inter-datacenter (inter-DC) traffic constitutes a significant portion of today's worldwide traffic while utilizing dedicated lines that are in different networks than the Internet, making it a very expensive communication. Consequently, inter-DC network providers are keen to minimize their expenses while guaranteeing the quality of service to their customers. As a result, scheduling available resources is of paramount importance to increase the efficacy of these networks for both their providers and customers. In this regard, software-defined wide area networks (SD-WAN) seem to be a promising solution to mitigate legacy WAN's restrictions such as lack of a global view. While conventional multi-protocol label switching (MPLS) tunnelling has proven to be a practical approach to guarantee performance, its significant service price can be a drawback. Utilizing Internet best-effort paths is a cheap and viable alternative. However, to utilize these paths, we have to take their capacity fluctuations into account to avoid over-allocation. In this thesis, we first characterize and estimate the fluctuations in short and long periods using statistical analysis and machine learning. Next, we take the estimated capacities into account and consider the problem of scheduling bulk transfer requests over best-effort SD-WANs to maximize the gained profit from successful transmissions. Furthermore, we propose an approximate algorithm with a significant computational advantage over our exact algorithm with an approximation ratio that only depends on the number of overlapping requests with the same profit to bandwidth ratio. Finally, we provide a thorough mathematical analysis of the approximate algorithm, as well as simulation and experimental results to evaluate our proposed algorithm's performance. The results show that our algorithm can improve the inter-DC provider's profit by an average of 60% while reducing ISP service costs by an average of 15%.

Preface

This thesis is an original work by the author and parts of this research will be published [A. Hosseini, M. Dolati and M. Ghaderi, "Bulk Transfer Scheduling with Deadline in Best-Effort SD-WANs", in Proc. IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, May 2021].

Acknowledgements

I would like to express my gratitude to Prof. M. Ghaderi for the guidance and advice he has provided throughout my time as his student. I would also like to thank my family and friends who supported me along the way.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Today, cloud service providers establish multiple datacenters (DCs) in a geographically distributed manner to improve their performance, scalability, and robustness, and provide services of high quality to their customers [41, 36, 45, 77]. These geo-distributed DCs are inter-connected through a Wide Area Network (WAN) that constitutes a significant portion of cloud service providers' yearly budgets. On the one hand, legacy WANs maintain a low average utilization to account for sudden high traffic bursts and mitigate packet loss. On the other hand, measurement studies show rapid growth in the inter-DC traffic. Thus, inter-DC WANs are assets whose costs sometimes exceed intra-DC networks [77, 69]. Furthermore, legacy WAN technologies lack a global view of the network; therefore, they perform resource allocation in a distributed manner, which leads to a system that performs globally sub-optimal.

Shortcomings of legacy WANs, such as lack of a global view and poor efficiency, motivated companies such as Google, Microsoft, and Facebook to migrate to Software Defined WANs (SD-WANS). SDN decouples the control logic (*i.e.*, control plane) from the forwarding logic (*i.e.*, data plane) and moves it to a centralized entity named the controller. This

controller enables a global view of the network and simpler management using application programmable interfaces (APIs). Centralized resource allocation and the global view allow us to schedule network traffic more efficiently and avoid under-provisioning.

Cloud service providers are interested in the characteristics of the SD-WAN traffic in order to be able to manage their resources more efficiently and maximize their revenue. Bulk transfer requests, which constitute a significant portion of the SD-WAN traffic, are usually associated with a deadline. Cloud service providers can make a profit by successfully submitting bulk transfers based on a service level agreement (SLA). The SLA generally entails the complete submission of a transfer before a specified deadline, which obligates the cloud providers to schedule the bulk transfers on time and space (*i.e.*, tunnels or paths) due to the resource limitations to meet the SLA requirements.

SD-WANs support a variety of connection types such as MPLS tunnels, leased lines, and the Internet. Even though cloud providers were able to mitigate some of the shortcomings of legacy WANs by leveraging SDN, using MPLS tunnels and dedicated lines is quite costly [36]. A cost-efficient alternative to MPLS tunneling is utilizing Internet best-effort tunnels. These tunnels, however, suffer from capacity fluctuations due to various reasons such as random cross-traffic, ISP traffic engineering, and link failure. As a result, Internet tunnels are not a desirable option for traffic that demands guaranteed service (*e.g.*, bulk transfers). To be able to utilize these best-effort tunnels, we have to take into account the capacity fluctuations to avoid overloading the tunnels and losing data.

Scheduling bulk transfers on a network with capacity fluctuations is an optimization problem with uncertainty. State-of-the-art works have used stochastic programming and robust optimization to address programming problems that involve uncertainty. In such problems, some or all parameters are uncertain. Stochastic programming requires that we know the random variable's distribution, which is hardly practical due to the extensive number of factors that affect the bandwidth values. In contrast, robust optimization relies solely on a range of estimations, namely uncertainty sets. This work argues that as opposed to the

complete distribution of the capacities, the range of their fluctuations (*i.e.*, the uncertainty sets) is predictable in short periods using statistical analysis and machine learning.

Many works in the literature have focused on traffic scheduling and resource allocation in SD-WANs [61, 60, 36, 41, 42, 52, 23, 45]. These works have studied different aspects of SD-WANs and have addressed various pertaining problems such as point-to-multipoint scheduling [23, 60, 61], store-and-forward scheduling [29, 48, 71], cost reduction [30, 51], deadline guarantee [52, 77], and fairness [36, 41]. However, state-of-the-art works have not considered utilizing best-effort tunnels to achieve MPLS-like performance while reducing operational costs in an SD-WAN. In this thesis, we first show that the bandwidth of the best-effort tunnels has a predictable behaviour. Second, we propose to use the Internet best-effort tunnels to emulate MPLS behaviour, taking into account the capacity fluctuations that may cause overloading and loss of data.

1.2 Thesis Objective

In this thesis, the first objective is to utilize statistical analysis and machine learning to demonstrate that the best-effort Internet tunnels' capacity shows predictable characteristics. Predicting Internet tunnels' capacity is of significant importance when scheduling transfers because unlike dedicated lines, their capacities are not fixed and fluctuate over time. Our second objective is to formulate the problem of scheduling bulk transfers over best-effort SD-WANs as a robust optimization problem and devise a scheduling scheme that provides a deadline guarantee for bulk transfer requests by considering the capacity predictions.

1.2.1 Bandwidth Prediction for Internet Tunnels

Bandwidth prediction is crucial to scheduling bulk transfers on a network whose capacity fluctuates. Numerous works focused on predicting computer networks' bandwidth using methods such as statistical analysis and neural networks. Even though statistical analysis

methods provide somewhat useful predictions, they lack the capability to remember values over arbitrary intervals, which makes them unable to deal with systematic deviations and sudden changes caused by random cross traffic, link failure and ISP traffic engineering [47]. Consequently, machine learning seems to be a promising approach to overcome the shortcomings of conventional prediction methods. State-of-the-art works demonstrate that a shallow neural network is sufficient to produce accurate predictions on networks' capacity [47]. With respect to bandwidth prediction, the following questions arise:"How variable is the bandwidth capacity of the Internet best-effort tunnels?" and "Are we able to predict the capacity of these tunnels?". In this thesis, we utilize neural networks to study the behaviour of Internet tunnels, determine their variability, and design a model to predict the capacity of these paths in short periods of time. These predictions allow us to make decisions some time into the future.

1.2.2 Scheduling Bulk Transfers over Best-effort SD-WANs

There have been numerous works in the literature concerning the scheduling of bulk transfers over WANs. Although some of these works have considered demand uncertainty, none of them have considered bandwidth capacity uncertainty and using Internet best-effort tunnels as the underlying network. In order to provide guaranteed delivery for bulk transfers with a deadline, the main question is: "How can we provide guaranteed delivery for bulk transfers with deadlines over best-effort tunnels?". In this thesis, we propose to use multiple best-effort Internet tunnels to schedule bulk transfer requests over an SD-WAN, and we formulate the problem as a robust optimization problem. In this way, we can avoid saving a headroom bandwidth to account for fluctuations, and we can guarantee an acceptable performance without paying a substantial cost for the underlying network.

1.3 Thesis Contributions

In this section we briefly summarize the contributions of this research thesis including the formulation, solution, and analysis of the bulk transfer scheduling over best-effort tunnels problem, and our bandwidth prediction model using neural networks.

- **Problem Formulation.** We formulate the problem of scheduling bulk transfer requests on best-effort Internet tunnels considering deadlines and the bandwidth capacity constraints to maximize the profit gained by successfully transmitting each request.

- **Bandwidth Prediction Model.** We set up a *long short-term memory* (LSTM) neural network which is used to learn order dependence in sequence prediction problems. We then train the network with data gathered from our own measurements. Finally, we utilize it to design our bandwidth prediction model. Our test results demonstrate that the capacity of best-effort tunnels is predictable in short periods.

- **Uncertainty Modeling.** By utilizing the bandwidth prediction data and considering the minimum and the maximum variations, We model the bandwidth capacity fluctuations using *cardinality-constraint sets* [9]. Furthermore, we rewrite our problem formulation as a *robust optimization problem* to consider capacity fluctuations.

- **Bulk Transfer Scheduling with deadlines Over Best-effort Tunnels.** We calculate the *dual* of our robust formulation in order to preserve the linearity of the problem and make it solvable by off-the-shelf solvers. However, it is still infeasible to solve large instances of our problem using this formulation due to time and computational complexity. Consequently, we design an approximate algorithm using the *iterative-rounding technique*, which solves a relaxed version of the problem.

- **Simulations and Mininet Experiments.** We present extensive model-driven simulations to demonstrate our algorithm's performance under different conditions against two

baseline algorithms, namely *Average Algorithm* and *Effective Bandwidth Algorithm* [46]. Furthermore, we conduct a number of experiments in the Mininet network emulator to verify our algorithm's performance results in a realistic network environment.

Parts of the research conducted in this thesis will be published in the proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM 2021). In this respect, the findings and the results regarding the scheduling of bulk transfers over best-effort tunnels will be published in the conference proceedings [38] and presented in the conference venue in May 2021.

1.4 Thesis Organization

This thesis is organized as follows:

- Chapter 1 discusses the motivation for our research, this thesis's objective, and a summary of the contributions.
- Chapter 2 provides the required background in *Bandwidth Prediction* and *Resource Scheduling in WANs*, and mathematical and software tools utilized in this study. Furthermore, it provides a brief review of *Bandwidth Prediction* using machine learning and statistical approaches. Then, it provides a thorough review of the *Resource Scheduling in WANs* state-of-the-art, including legacy WANs and SD-WANs.
- Chapter 3 defines our bandwidth prediction problem. Then it describes the methods with which we collected our data, created the neural network, and made predictions for bandwidth capacities. Finally, it demonstrates the obtained results.
- Chapter 4 defines our resource scheduling problem in best-effort SD-WANs, and proposes a robust formulation of the problem. Then, it proposes an approximate algorithm that has a computational and run-time advantage over our exact method.

- Chapter 5, first, provides a theoretical analysis of our approximate algorithm. Then, it provides the experimental analysis, which includes simulations and Mininet experiments.

- Chapter 6 concludes the thesis by providing a summary of this research as well as its possible future directions.

Chapter 2

Background and Related Work

2.1 Bandwidth Prediction

Bandwidth prediction is widely used for applications such as video streaming and voice over IP to improve user experiences. The available bandwidth is predicted some time into the future to allow the application to proactively decide the best level of service it is going to provide to the end-users. The bandwidth prediction problem has been traditionally addressed using time series forecasting (TSF), which entails creating a model to find correlations between the past data and the future data. We can divide the bandwidth prediction methods based on TSF into two main categories: 1) Statistical Analysis 2) Machine learning

2.1.1 Statistical Analysis

In this group of methods, algorithms such as moving average and simple linear regression is utilized to predict future bandwidth based on the previous data.

Simple Linear Regression

Simple Linear Regression is a statistical method that allows us to study the relationship between two continuous variables: *the predictor* and *the response*. Consider the model

function (2.1):

$$y = \alpha + \beta x, \tag{2.1}$$

which describes a line with slope β and y-intercept α. In this model, x is the predictor, and y is the response. Linear regression is the relationship between the optimal α and β pair that allows the prediction of y based on x with the lowest possible error. Error is defined as the deviation of the predicted value of y from its real value. Various methods such as the least-squares approach, can be applied to achieve a near-optimal pair [34, 19].

Moving Average

Another method based on statistical analysis for bandwidth prediction is the Auto Regressive Integrated Moving Average (ARIMA). An ARIMA model is a form of regression analysis that predicts future values by studying the previous values' differences. An ARIMA model constitutes the three following characteristics:

- Autoregression: In an autoregressive model, the output variable is determined by its previous values as well as a stochastic term.
- Integrated: The model uses the differences between the data values rather than the data itself to predict the future values.
- Moving Average: In a moving average model, the output variable is determined by a stochastic term's current and previous values.

Early models based on ARIMA were able to recognize the short-range dependency of time series data (*i.e.*, network traffic) [63, 4]. However, later studies demonstrated that network traffic, in addition to short-term dependency, exhibits long-range dependency, which makes ARIMA-based models unsuitable for network traffic and bandwidth prediction [49]. To account for short-range and long-range dependency of traffic data, the Fractional Auto Regressive Integrated Moving Average (FARIMA) model was utilized by later works [66, 17].

Even though these simple algorithms work for the average case, their lack of adaptability to sudden changes, anomalies, and systematic deviations make them a less preferable method in environments with significant variability.

2.1.2 Machine Learning

Machine Learning (ML) means utilizing algorithms that improve automatically over time. It consists in using statistics to find patterns in substantial amounts of data [57]. Today, ML empowers many platforms such as Facebook, Amazon, and Google to provide a better experience to their users by learning their habits and preferences. ML-based bandwidth prediction methods have come to prominence due to their flexible ability to address the issues of traditional TSF methods. Bandwidth prediction with ML is achieved by utilizing Neural Networks (NNs).

Neural Networks

In Machine Learning, NNs are a set of algorithms modeling the human brain, which posses the ability to be trained to find patterns in raw data [37]. Neural Networks are able to cluster or classify data and make correlations between previous and future events, and eventually, make predictions about the output of a system. A number of works in the literature have utilized neural networks for bandwidth prediction [47, 53, 13, 27, 80, 18] whose results are shown to be promising.

An NN comprises simple computational units called *neurons* (*i.e.*, nodes). A neuron receives the inputs from multiple edges, performs a weighted sum according to the edge weights, and applies a non-linear function called *the activation function*, which is depicted in Fig. 2.1. Also, equation (2.2) shows the mathematical representation of a neuron, where x, w, b, and f represent the input vector, weight vector, neuron bias, and activation function,

respectively.

$$f(b + \sum_{i=1}^{n} x_i w_i) \tag{2.2}$$

The activation function can be one of the many non-linear functions such as *sigmoid*, *tanh*, *rectified linear units (ReLU)*, to name a few, which are defined by equations (2.3),(2.4), and (2.5), respectively.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.4}$$

$$ReLU(z) = \max(0, z) \tag{2.5}$$

$$a(z) = z \tag{2.6}$$

Furthermore, we apply the identity function shown in equation (2.6) for regression problems that deal with continuous values.

Long Short-Term Memory

In a neural network, all inputs and outputs are independent. Consequently, NNs are not suitable to make predictions for data in which a dependency exists within a sequence. For instance, predicting the next word in a sentence requires information regarding the previous words. Recurrent Neural Networks (RNNs), however, utilize sequential information to make predictions. In other words, in an RNN, the output is dependant on the previous predictions. Due to architectural restrictions, recurrent neural networks have trouble maintaining a short-term memory to make predictions for longer sequences of data. As a result, traditional RNNs are rarely used for real-world problems. Long Short-Term Memory (LSTM) networks were proposed to resolve the memory problem in traditional RNNs. They are a type of RNN that is used to learn the order dependence in sequential prediction problems with longer sequences, such as the word prediction for long paragraphs or the bandwidth prediction problem. The architecture of an LSTM allows it to maintain its short-term memory for

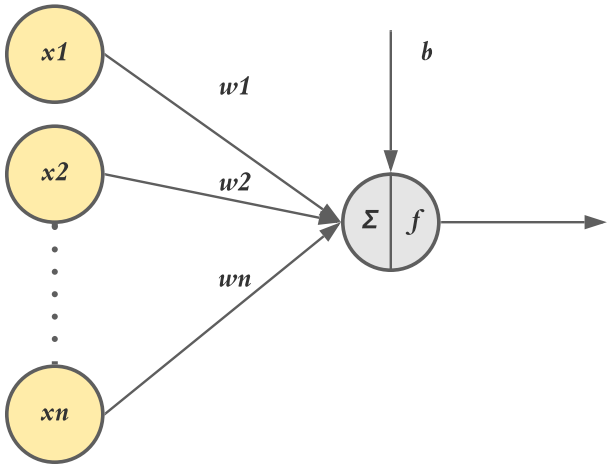


Figure 2.1: Structure of a neuron.

longer periods, making it very efficient in detecting temporal patters in sequential data [55]. As it is shown in Fig. 2.2, an LSTM network essentially consists of an input layer, several hidden layers, and an output layer [33].

An LSTM unit is a combination of inter-connected simpler nodes. Work [35] introduces the main components of an LSTM unit as follows:

- Constant error carousel (CEC): This component acts as the memory for past information.
- Input Gate: This component protects the unit's memory from irrelevant input.
- Output Gate: This component protects other units from interference by the information stored in the memory of the unit.

Each of these components can represent a conventional artificial neuron since they apply an activation function to the weighted sum of their inputs. These components also regulate the flow of data between different units. However, conventional LSTM units with CEC gates faced an issue under specific circumstances, which caused the memory unit to reach saturation and perform as a memory-less unit. To address this issue, later works utilized a more efficient model proposed by [33], which replaced CEC gates with forget gates. Forget gates can learn to gradually decay the memory to zero once its contents are out of date.

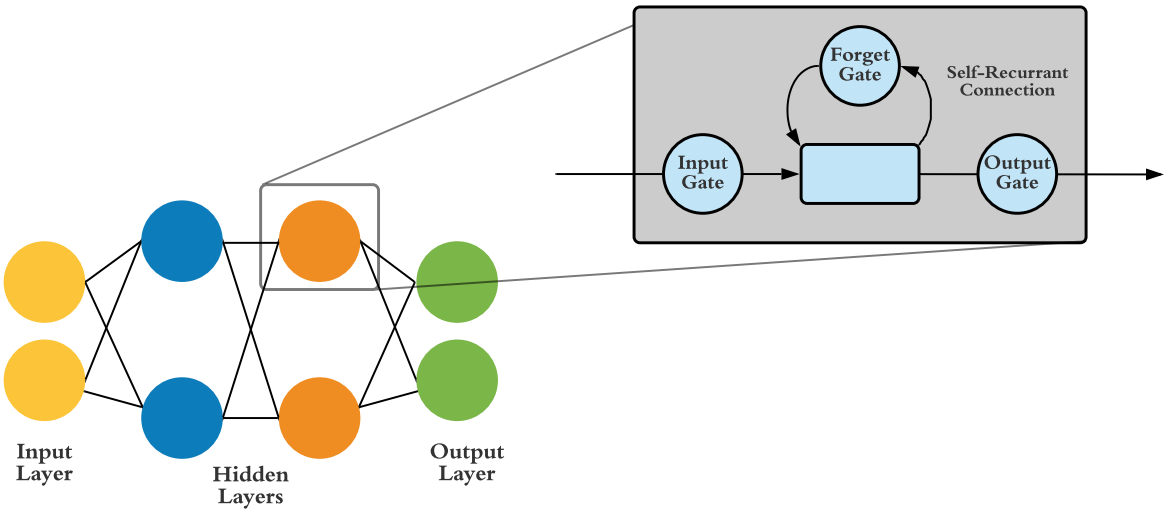


Figure 2.2: An LSTM RNN for time series prediction.

Accuracy

The accuracy of a neural network model is the precision on which the neural network performs concerning its objective. In the case of a time series prediction, such as the bandwidth prediction problem, the accuracy shows how close the predicted bandwidth values are to their real values some time into the future. In this thesis, we use the *Root Mean Squared Error (RMSE)* as a metric for our model's performance. In statistics, the MSE of a predictor is equal to average of the squares of the errors (*i.e.*, difference between the real values and the predicted ones). Furthermore, the RMSE of a predictor is equal to the square root of its MSE. In order to calculate the RMSE of a predictor for a vector of n samples, we use the following equation:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}, \tag{2.7}$$

where Y is the vector of observed values and $\hat{Y}$ is the vector of predicted values.

2.2 Resource Scheduling in WANs

Today, with a rapid increase in the number of Internet-based applications and their users, cloud, Internet, and content providers have established multiple geographically distributed DCs (datacenters) worldwide to provide their customers with reliability, performance, and scalability. The wide area network (WAN) that interconnects these DCs carries a significant fraction of Internet traffic [36, 41]. The inter-DC WANs (Inter-dc WAN) are often a dedicated network separate from the Internet, which, along with the high traffic volume, makes it an expensive resource. Consequently, it is crucial for the providers of this network and its customers to reduce its annual costs and fully utilize its potential [36, 41]. Inter-DC traffic has certain characteristics such as a wide range of deadlines, transfer sizes, and resource requirements. These characteristics enable operators to properly schedule the transfers to improve the networks' efficiency. Essentially, Inter-DC traffic falls into three groups based on its characteristics [36]:

- **Interactive:** This type of traffic is high priority traffic that is bursty and small. An example of such traffic is a user request transferred to a DC from another DC because the information does not exist in the requester DC.

- **Elastic:** Elastic traffic consists of bulk transfers that demand a timely delivery that ranges between a few minutes to a few hours. For instance, traffic of distributed computing applications such as MapReduce is considered as elastic traffic.

- **Background:** Background traffic is the group of huge transfers with extended deadlines or no deadline. Data replication between DCs for redundancy purposes is an example of background traffic.

2.3 Software Defined Networks

Software Defined Networks (SDN) is a recent paradigm in computer networks that decouples the control logic (*i.e.*, control plane) from the forwarding logic (*i.e.*, data plane) and moves it to a centralized entity called a controller. The switches in this paradigm are simple boxes that merely forward the traffic according to the rules set on them by the controller. In case a new flow of traffic arrives, the switches forward the flow to the controller, and the controller sets the relevant rules on the switches.

The decoupling of the control plane from the data plane provides a global view of the network and allows simple network management using programmable APIs. The end-user connects to the northbound APIs and applies configurations through the control plane. Furthermore, the switches' simplification enables the controller to program them using a simple southbound API, regardless of their vendors. OpenFlow is the most prominent southbound API and is widely used in the industry and academia.

In general, SDN seems to be a promising alternative to traditional decentralized networks due to its simpler management and reduced costs. Over the past few years, in addition to service providers such as Google and Microsoft, network equipment vendors such as Cisco and Juniper have started to produce OpenFlow-compatible equipment.

Fig. 2.3 demonstrates the architecture of an SDN network with the OpenFlow protocol, wherein the controller receives the network's configuration from the end-user through the northbound APIs. Furthermore, the controller applies the relevant rules on the switches through the southbound APIs (*e.g.*, OpenFlow).

2.4 Mathematical Preliminaries

Constrained Optimization. Constrained optimization is the process of finding the maximum or the minimum of an objective with respect to several variables that are subject to at least one constraint. Equation 2.8 demonstrate the general form of a constrained

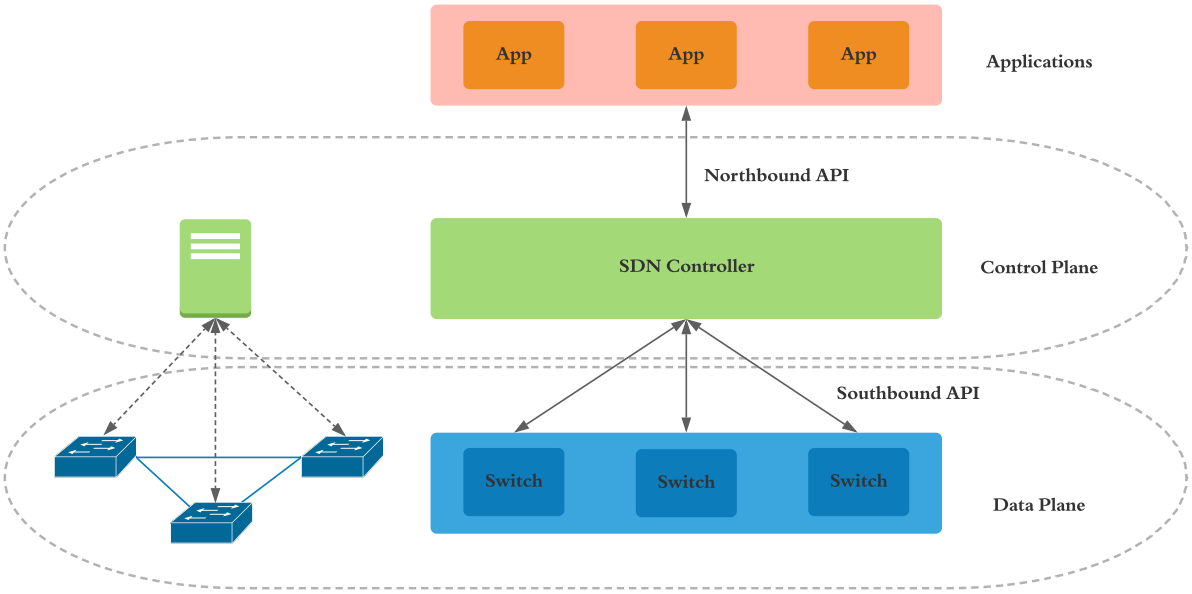


Figure 2.3: Architecture of an SDN network [54].

optimization problem:

$$\begin{aligned} \min \quad & f(x), \\ \text{subject to} \quad & g_i(x) = c_i, \quad \text{for} \quad i = 1, ..., n \quad \text{Equality constraints} \\ & h_j(x) \geq d_j, \quad \text{for} \quad j = 1, ..., m \quad \text{Inequality constraints} \end{aligned} \tag{2.8}$$

where the first line is the objective and the second and third lines are the constraints.

In this problem, variable x can be of various types such as a real number ($x \in R$), an integer ($x \in Z$), a natural number ($x \in N$) or a binary number ($x \in \{0, 1\}$). Moreover, while the above example's constraints are simple linear equations, in an optimization problem, the constraints are often non-linear. Based on the variables and the constraints, optimization problems are divided into *Linear Integer*, *Non-linear Integer*, *Mixed-Integer Linear programs*, and *Mixed-Integer Non-Linear programs*.

Mixed-Integer Non-linear Program (MINLP). There is a mix of integer and real variables in this type of optimization problem, and there is at least one non-linear constraint. Our deadline-aware bulk transfer scheduling on best-effort paths is an example of an MINLP.

This type of optimization problem is generally NP-Hard, due to the integrality of some of its variables. Furthermore, non-linearity of the constraints makes some instances of this problem computationally intractable and nearly impossible to solve with typical solvers.

Robust Optimization. General optimization deals with deterministic input, while stochastic and robust optimization deal with uncertainties. Stochastic optimization requires that the distribution of the variables be known. On the other hand, robust optimization is only concerned with uncertainty sets. In other words, having partial information regarding the variables would suffice to find a robust solution. In our problem of deadline-aware bulk transfer scheduling on best-effort paths, we use our neural network prediction model to predict a maximal and a minimal value for bandwidth capacities, namely the cardinality-constrained uncertainty set of the bandwidth capacity. Consequently, we reformulate the problem as a Γ-robust optimization problem considering the uncertainty set for the bandwidth capacities [9].

Consider the optimization problem (2.9), in which variable a_i is subject to uncertainty and its value fluctuates within the range $[\bar{a}_i - \hat{a}_i, \bar{a}_i + \hat{a}_i]$ where $\bar{a}_i$ and $\hat{a}_i$ are the average and the maximum deviation of variable a_i, respectively.

$$\begin{aligned} \min \quad & f(x), \\ \text{subject to} \quad & \sum_{i=1}^{n} a_i x \leq b, \quad \forall a_i \in [\bar{a}_i - \hat{a}_i, \bar{a}_i + \hat{a}_i] \end{aligned} \tag{2.9}$$

In this problem, we have to assign the value of x by taking into account the deviation of a_i. To ensure that the constraint is not violated, considering all the possible values of a_i is not feasible. Instead, we may consider the worst-case scenario to ensure that the constraint is never violated. In the worst-case, a_i takes value $\bar{a}_i + \hat{a}_i]$. Consequently, we can rewrite the constraint as follows:

$$\sum_{i=1}^{n} (\bar{a}_i + \hat{a}_i) x \leq b. \tag{2.10}$$

Even though this scenario will never lead to a situation in which the constraint is violated,

assuming that all the variables a_i are always deviating maximally from their average, is a rather conservative approach. Instead, we can define a subset Q of the variables with size Γ to take maximum deviations. Therefore, we can rewrite the constraint as follows:

$$\sum_{i=1}^{n} \bar{a}_i x + \max_{\substack{Q \subseteq A, \\ |Q| \leq \Gamma}} \sum_{i \in Q} \hat{a}_i x \leq b. \tag{2.11}$$

If we replace the constraint (2.11) with the one in (2.9), a Γ-robust optimization problem is formed. Since the max operator is non-convex, we can utilize the duality theorem to linearize the problem.

Duality. In optimization theory, the duality theory indicates that a problem can be viewed from two perspectives: the primal and the dual problem. The solution to the dual problem provides a lower bound for the primal problem. In the case of a convex problem, the solution to the dual is the same as the solution to the primal problem.

Consider the constraint (2.11) of the Γ-robust optimization problem described before. In order to linearize this constraint, first, we extract the inner part, including the max operator, and rewrite it as a different optimization problem as follows:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \hat{a}_i x z_i \\ & \sum_{i=1}^{n} z_i \leq \Gamma, \\ & 0 \leq z_i \leq 1. \end{aligned} \tag{2.12}$$

where z_i is a binary variable that indicates whether variable a_i is deviating maximally from its average (*i.e.*, it is subject to worst-case scenario). However, the term $x_i z_i$ is still keeping the non-linearity of the problem. To address this, we take the dual of this problem and rewrite it as a linear program:

$$\begin{aligned} \min \quad & \Gamma x + \sum_{i=1}^{n} \hat{\beta}_i \\ \text{subject to} \quad & \alpha + \beta_i \geq \hat{a}_i x, \\ & \alpha, \beta_i \geq 0. \end{aligned} \tag{2.13}$$

Finally, to achieve the complete Γ-robust formulation, we can replace the inner optimization problem in constraint (2.11) with equation (2.13), and the result with the constraint of the original optimization problem (2.9).

2.5 Software Tools

Gurobi. Gurobi optimizer is a very fast, user-friendly, widely-used off-the-shelf solver capable of solving a wide range of optimization problems [1]. Gurobi supports a variety of programming languages such as Python, Java, and C/C++. Furthermore, Gurobi supports multi-processor operations, which makes it a powerful distributed solver. Also, academic users can obtain a free academic license. We have used Gurobi in Python 3 environment to solve the instance of our robust optimization problem.

Mininet. Mininet is an open-source network emulator mainly focused on SDN [3]. Mininet utilizes Linux kernels essential features such as namespaces and process virtualization to create configurable virtual networks that emulate a real one. The default Mininet switches are software defined switches with OpenFlow support. Furthermore, Mininet provides an extensive Python API that allows users to create sophisticated custom networks. Mininet was designed to target personal computers, making it a popular network emulator in academia. We have used Mininet to conduct experiments in a realistic environment.

Iperf. Iperf is a network tool for active measurements of bandwidth on IP networks [2]. It supports TCP and UDP protocols and various parameters as input as well as different operational modes and packet types, which makes it a powerful tool. Iperf can measure

the throughput between the two ends in one or both directions. It is open-source software written in C that runs on multiple platforms, including Linux, Windows, and Unix. In this work, we have used iPerf for bandwidth measurement.

2.6 Related Work

2.6.1 Bandwidth Prediction

In this section, we study the state-of-the-art works in bandwidth estimation and prediction. These works can be grouped into statistical (*i.e.*, conventional) and machine learning methods.

Statistical Analysis

Traditional methods for network traffic and bandwidth prediction are based on statistical analysis. In some works, probing is combined with statistical analysis to produce more realistic predictions and estimations under intense scenarios.

Work [67] introduces Spruce which works based on the Probe Gap Model (PGM). In this model, multiple consecutive probes are sent over the network, whose arrival gap is then measured in the receiver. Utilizing this information, Spruce estimates the available bandwidth. In [43], DietTopp is proposed, which uses probe trains along with simple linear regression to estimate the bandwidth. Estimating the bandwidth using probe trains involves sending a train of probes with a specific rate and comparing it to the rate with which they are received at the sender. The sending rate is increased gradually to trigger the bottleneck spacing effect and make more accurate measurements, which means that the congestion on the network's tight link directly affects the arrival rate of probe trains [39]. Pathload [40] is based on the idea that the delay of a stream of periodic probes will have an increasing trend provided that the stream rate is higher than the available bandwidth. Work [24] estimates the available bandwidth by inducing congestion and repeatedly sending probe pairs with random

rates. Moreover, it uses Kalman filtering to improve the estimation over time. However, the Kalman filter only works efficiently when the measurement noises are Gaussian. Similarly, PathChirp [62] utilizes self-induced congestion as well as chirps to estimate the available bandwidth. Chirps are a sequence of probes, and they differ from probe trains and pairs in that they are more bandwidth efficient, and they can capture critical delay correlations.

Work [63] proposes an upper bound for network traffic prediction based on ARIMA models. Furthermore, using analysis, it argues that traffic prediction is highly affected by various factors and traffic properties such as the traffic measurement intervals, the network control time-scale, and the utilization target of network resources. Corradi *et al.* [17] study the applications of FARIMA processes for modeling of traffic with long-range and short-range dependencies. To this end, they first provide a theoretical analysis. Then they apply FARIMA to simulated LAN and Video conferencing traffic to evaluate long-range and short-range modeling capabilities of this approach. Similarly, work [66] proposes a procedure for traffic prediction using FARIMA models, and shows that FARIMA models, as opposed to ARIMA models, are capable of capturing both short-range and long-range dependencies of network traffic.

Conventional methods which, are based on active probing and statistical preprocessing can barely make any viable predictions for heterogeneous traffic of today's networks. Furthermore, they suffer from probe packet interference, packet loss, the randomness of cross traffic, multiple tight links, and clock synchronization.

Machine Learning

As mentioned in the previous sections, ML can be used for regression problems such as TSF for bandwidth and traffic. Recent works that utilized ML have achieved better results than the works that were based on traditional statistical analysis. While the majority of the TSF works in networks are based on NNs, there has been a number of works that used Reinforcement Learning (RL) or Support Vector Regression (SVR).

Neural Networks.

Recently, there have been a great number of works employing NNs for bandwidth and traffic prediction. Khangura *et al.* [47] argue that it is reasonable to use machine learning in order to estimate bandwidth more efficiently. Their measurements are performed by utilizing the vectors of packet dispersion. To this end, they used the packet trains approach in which every probe packet adapts its rate based on the feedback from the previous packet. Similar to [47], in [26], Eswaradass *et al.* propose a bandwidth prediction system based on NNs for grid environments. The input of the NN is the minimum, maximum, and average bandwidth in the previous iteration. The results show that this system outperforms previous statistical analysis models in the same environment. Work [18] proposes an NN model to predict network traffic targeting ISP networks. The model is trained using ISP backbone traffic as well as inter-ISP traffic. Tests for real-time forecasting (online forecasting on a few-minute sample), short-term (one-hour to several-hours sample), and mid-term forecasting (one-day to several-days sample) demonstrate better performance and results, as well as lower time and computational complexity than the traditional statistical analysis methods. Work [55] argues that it is crucial that we account for the variations in the mobile networks and predict them to some extent in order to adjust the transmission rate and improve the user experience in applications that require high bandwidth and low delay. These variations can be due to several reasons, such as mobility and fading. The writers have designed a neural network using LSTMs and a Bayesian fusion to estimate mobile networks' bandwidth in scenarios with high mobility such as in the subway or a bus ride. In work [72], the authors argue that utilizing neural networks on its own does not necessarily improve the accuracy of predictions due to the biases that happen in the training phase of the model, due to most solutions being only tested in the simulation or emulation environment rather than in real Internet environment. In this work, which was deployed and tested using a real video streamer with thousands of monthly users, supervised learning is combined with a conventional buffer-based method to make more accurate predictions that can be checked immediately.

Other works have studied the effects of training algorithms for network traffic prediction [13, 80]. In work [13], Chabaa *et al.* investigate the effects of using different back-propagation algorithms for training of NNs for Internet traffic prediction. While back-propagation algorithm involves local optimization, revolutionary algorithms such as Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC), which involve global optimization, can be utilized for NN training. Work [80] proposes a hybrid training algorithm based on PSO and ABC. The NNs that were trained using this technique produced more accurate predictions. This method also improves the training speed.

Reinforcement Learning. Reinforcement Learning (RL) has been mostly used for Adaptive Bitrate applications. In [53], Mao *et al.* propose Pensieve, which utilizes reinforcement learning in order to employ an ABR scheme in the client-side of a video player to increase the Quality of Experience (QoE) for the end-user. Pensieve, which works with neural networks, can adapt to a diversity of environments and QoE parameters as opposed to the state-of-the-arts that rely on static models and fixed assumptions about the environment. Chiariotti *et al.* [14] also use RL for an online Dynamic Adaptive Streaming over HTTP (DASH) to improve the long-term QoE for end-users. This work proposes a parallel learning technique that achieves a faster and more accurate learning process than previous works.

Support Vector Regression. While other works mainly use NNs for bandwidth and traffic prediction, work [8] has studied Support Vector Regression (SVR) for TSF of link load in ISP networks. The model proposed by this work can provide significant improvement over traditional methods on heterogenous traffic of an ISP network. While SVR has lower computation overhead than NNs, NNs outperform SVR when extensive and continuous training is required. Similarly, work [56] proposes a TCP throughput prediction method based on Vector Regression modeling that combines prior transfer and current measurements. In other words, it consists partly of active measurements to ensure accuracy.

Machine learning has been successfully utilized by numerous works in the literature for traffic and bandwidth prediction. While NNs and SVRs were applied to TSF of traffic and

bandwidth in various environments [26, 18, 47, 55, 72], RL has been mostly utilized for optimal adaptive bitrate in real-time video applications. Machine learning approaches can accurately predict short-term and long-term dependencies at low complexity with simple models.

2.6.2 Resource Scheduling in WANs

Legacy WANs

This section briefly introduces works based on MPLS traffic engineering (MPLS-TE) and decentralized traffic engineering and resource scheduling in Inter-DC WANs.

Work [5], introduces RATE, an MPLS-based traffic engineering system that receives user inputs from an interface and provides bandwidth-guaranteed LSPs (Label Switched Paths) to online requests that arrive one by one. RATE operates in a centralized fashion wherein it obtains the topology and link-state information either through standard protocols such as SNMP (Simple Network Management Protocol) or link state peering. However, QoS-related static link and nodal attributes are maintained by tracking the user's allocation through a graphical interface. Furthermore, LSPs are established from the source by signaling means.

In [11], writers introduce an RCP (Routing Control Platform), which is logically centralized and separate from the IP forwarding platforms. The RCP performs routing on behalf of iBGP (interior Border Gateway Protocol) routers; then the routes are communicated to the routers using the protocol of the same name.

Ethane [12] proposes a network architecture for enterprise networks whose primary enabler is a controller with a global view of the network and enforces network-wide fine-grain policies for the arriving flows. This work is a building block of today's SDNs and the OpenFlow protocol; however, it works in compliance with legacy switches.

Work [15] proposes WISE, a traffic engineering server for MPLS-based IP networks to address some of the MPLS TE issues such as globally sub-optimal allocations and lack of a

sophisticated traffic statistics and analysis. WISE is a centralized solution that works in an offline fashion.

In order to achieve fairness between multiple commodities over a set of fixed resources, max-min fairness is a well-known approach. However, it suffers from scalability issues when the number of commodities and resources increases. Work [20] proposes the Upward max-min fairness, a relaxed solution that performs close to max-min fairness in terms of utilization and fairness. Furthermore, work [21] extends the previous work and achieves a tradeoff between fairness and throughput, as well as it speeds up the algorithm by reducing the number of steps required to solve the problem.

MATE [25] proposes a multipath traffic engineering solution for MPLS networks that balances the traffic based on the measurements and analysis in order to prevent path congestion. This decentralized work, however, does not account for the different service requirements of various applications. Also, it lacks an admission control, which leads to gradual congestion of the network when the number of arriving flows is high.

Work [31] refers to traditional IP routing protocols for traffic engineering by utilizing link weights based on a global view of the network and traffic. This view, however, is achieved through means such as router configuration and SNMP. This process is not flexible and is sluggish, as well as the maintenance, and the implementation are cumbersome.

Kandula *et al.* introduce in [44] TeXCP, a network-based adaptive multipath routing in which the load is adaptively balanced between multipaths by an administrative domain. In this traffic engineering work, as opposed to prior works that utilize static load balancers or offline multipath route optimizers, the load balancing happens in realtime to protect the network from congestion, link failure, and traffic bursts.

TEAM [64] proposes an automated manager for networks that use a combination of MPLS and DiffServ (differentiated services) to provide quality of service and to reduce congestion. TEAM is implemented in a centralized fashion and utilizes online measurements to perform resource and route management.

All the works introduced in this section fail to account for transfer requests with a deadline which, according to studies [36] constitute a major portion of today's Inter-DC traffic. Moreover, the expensive nature of the MPLS solutions, low utilization, and lack of a unified solution for network monitoring and analysis of the heterogeneous networks make some of these works unsuitable for today's inter-dc WANs.

SD-WANs

In this section, we first discuss SD-WAN resource scheduling. We then cover the proposed works in multiple groups based on their objectives, and finally, we present a classification for them. Legacy Inter-DC WANs that are typically implemented using MPLS TE (Multi-protocol Label Switching Traffic Engineering) suffer from low utilization. The first culprit behind this problem is over-provisioning the resources to account for peak traffic hours, and the second one is the lack of a global view in the decision-making components. In order to solve the problems mentioned above, SD-WAN (Software-defined WAN) has separated the control-plane from the data-plane, moving it to a central entity called the controller, which has a global view of the network, including information about the switches, links between them, and attached hosts. The global view enables the operators to make globally optimal decisions by properly scheduling transfers and efficiently allocating resources.

Throughput Fairness. Microsoft and Google have proposed their SD-WAN solutions, namely, SWAN and B4 [41, 36]. In both works, the objective is to maximize the network utilization for their private networks since they are costly assets and constitute a notable portion of the companies expenses, as well as to and provide fairness to the flows. To this end, they use max-min fairness in a discreet time environment where scheduling happens at the beginning of every timeslot for the flows that have arrived earlier. However, they only schedule flows over paths and do not take future timeslots into consideration. As a result, their approaches do not apply to requests with deadlines as they are likely to miss their deadlines.

Deadline Guarantee. Other works have been proposed that are deadline-aware [45, 52]. Tempus [45] focused on flow fairness by maximizing the minimum transfer rate between all the flows. However, Tempus lacks admission control and simply admits every flow. Consequently, some admitted flows might lose their associated deadlines due to the limitation of resources. There are other works, however, that provide a guaranteed admission before the deadlines. PGA [52] admits only flows whose deadlines can be met with the available resources. As opposed to the majority of state-of-the-art that focus either on hard or soft deadlines, this work simultaneously considers soft and hard deadlines. In all the works mentioned above, capacity fluctuations have not been taken into account. In this respect, the capacities are either fixed and stable or have predictable fluctuations due to interactive traffic. Authors in [73] consider a fixed prediction error for realtime traffic. Transmission of bulk transfers are conditional upon the status of the realtime traffic. If the realtime traffic goes beyond the expected volume, bulk transfers are delayed in descending order of their deadlines. Amoeba [77] is another work that considers capacity fluctuations but does not provide a proper formulation and merely addresses uncertainty by reserving a proportion of the resources according to the predicted cross or realtime traffic. These works cannot characterize the uncertainty to provide a deadline guarantee and achieve a desirable resource utilization. Works such as NetSticher [48] and ElasticTEN [71] address store-and-forward mechanism in the inter-DC WAN. This mechanism is beneficial when the source and the destination of a request are in different time-zones, and therefore, have different peak-hours. ElasticTEN proposes a dynamic time-expansion graph for temporal-spatial scheduling of requests to reduce the complexity of the graph, while other methods copy the network graph for every timeslot and solve the scheduling problem. However, it lacks an admission control mechanism which can cause some admitted requests to miss their deadlines. Both works assume that all bandwidth capacities are known in advance.

Work [23] creates a Robust formulation of a Point-to-Multipoint transfer with deadlines considering demand uncertainty for realtime transfers. Similar to other P2MP approaches,

they use Steiner trees to route the transfers. This work requires demand estimates of all realtime flows in the network, limiting the scalability of their approach.

Cost Reduction. The store-and-forward (SnF) mechanism is a mechanism used in systems where the source and destination are in different time zones, having different peak hours. In this approach, the intermediate nodes store the traffic in peak hours and send them to the destination in non-peak hours. Postcard [30] is an example of a work based on SNF that only considers fixed capacities without fluctuations. Utilizing a usage-based billing scheme provides users with much flexibility in their transmission schedule. For instance, TrafficShaper [51] makes use of the free burstable timeslots in the q-percentile billing scheme and maximizes the transmission volume during those timeslots. Utilizing the free burstable timeslots allows them to transmit more transfers and gain more profit without paying more for the traffic. Similarly, in [42] the Pretium framework provides a dynamic pricing scheme. However, its objective is social welfare which is the total profit gained from successful transfers minus the cost of transferring them. The authors claim that customers are encouraged to report truthful characteristics rather than expand or tighten them to receive a better service. The prices are dynamically chosen at the beginning of a time window based on the statistics of previous time windows. For every bulk transfer, an initial route and schedule are decided upon arrival. Then at the beginning of every timeslot, routes and schedules are rescheduled in order to guarantee the promised service. Also, [50] introduces a similar solution for deadline-agnostic flows with dynamic provisioning, which works in a distributed fashion. However, it fails to consider capacity uncertainty. Jetway [29] is an inter-DC solution for deadline-agnostic video flows. This work only takes into account the historical data about the system in order to make decisions. Jetway admits the maximum number of flows that the available resources can support. Subsequently, it acquires the minimum amount of extra resources to accommodate the remaining flows. Work [59] uses a similar approach for deadline-aware bulk transfers. Considering only historical data, although making the system simpler, has the disadvantage of not handling uncertainty in any form.

Completion Time. Yang *et al.* in [74] propose a scheduling mechanism that minimizes the bandwidth costs and guarantees transfer deadlines. Even though the bandwidth capacity is not fixed in this work (as opposed to state-of-the-art), it does not characterize the capacity fluctuations, thus failing to address scheduling in environments with prevalent fluctuations (such as the environment in our problem). Although most works in the literature propose solutions for point-to-point (P2P) transfer requests, there are works proposed for point-to-multipoint (P2MP) ones. Noormohammadpour *et al.* [61, 60] propose solutions based on Steiner trees for P2MP transfers. Both works focus on reducing completion time and bandwidth usage at the same time. QuickCast [60] is proposed to overcome the challenge of shared bottleneck links among different destinations in a single P2MP transfer. They divide destination nodes into subgroups and P2MP transfers into multiple transfers to overcome this problem. They both only address a network with a fixed, predictable bandwidth capacity.

While works [77, 45, 52, 23, 42] consider demand fluctuations for high-priority traffic (i.e: interactive traffic), none of them consider using unreliable resources - such as that of the Internet - with a fluctuating capacity to provide performance similar to that of MPLS-TE while reducing operational costs.

Quality of Service. In BDS [79], Zhang *et al.* designed a centralized solution for inter-datacenter networks. In order to speed up the system, they decoupled the control algorithm into the scheduling of data transfers and routing into overlay paths. The overlay paths are chosen to be bottleneck-disjoint, meaning that they do not share any bottleneck paths. The scheduling is done using the rarest-first manner wherein each cycle, the block with the fewest duplicates is selected. This approach has been shown to balance the availability of blocks and avoid starvation. The objective is to maximize the throughput in each cycle.

Work [76] claims that existing solutions for DCN and WAN do not work in compliance because of the discrepancies between the nature of these two environments regarding buffer depths and delays. These discrepancies make solutions such as ECN or delay signal ineffective. For instance, the RTTs of inter-DC and intra-DC can vary by three orders of

magnitude. Therefore, they require a very different ECN threshold. Furthermore, the delay signal cannot distinguish between the congestion in inter-DC and intra-DC as well, making it inefficient on its own. Consequently, they propose using both ECN and delay signal in conjunction to achieve reasonable congestion control for cross-DC networks.

Gao *et al.* in [32] propose a model for an inter-DC SD-WAN taking into account the traffic priorities. In their model, the centralized controller is decomposed into a distributed two-layer control structure. In this structure, the network is divided into several domains, and each domain is controlled by a local controller. Furthermore, there is a central controller to control the local controllers. While they took two traffic types into consideration including background traffic and real-time user traffic, the optimization problem's goal is to maximize the background traffic throughput. The main problem is decomposed into multiple sub-problems using the dual decomposition method. Subsequently, each sub-problem is solved, and the results are used to solve the main problem. The experiment results demonstrate that this architecture can perform as well as a centralized method with increased scalability.

Paper [70] proposes QTE to maximize background throughput and provide Quality of Experience (QoE) for client-triggered traffic. They claim that relying solely on traffic priorities is not sufficient to get QoE due to the complicated relationship between QoE, delay, and bandwidth. First, they propose an optimization framework for QoE-aware SD-WAN. They then formulate the QoE problem as two optimization problems and propose a heuristic to solve it. The first problem intends to maximize the end-user QoE by routing client-triggered flows over proper tunnels, while the second problem uses the remaining bandwidth to maximize the utilization by routing the background flows based on their weights.

Works [79, 76, 32, 70] provide consistency to the system and provide QoE to the users. However, in all works, capacity uncertainty has not been considered, and capacities are fixed and known a priori, which is not the case in real-world networks, especially for the best-effort Internet tunnels.

In [7], the authors propose a platform of software-defined internet exchange points (IXPs)

in which a logically centralized controller enforces rules onto IXPs to provision inter-ISP routes with a better quality of experience (QoE) to end-users. To this end, they take into account the bandwidth and delay demand of each origin and destination pair. They have used the Branch and Reduce method so as to decrease the complexity of the resulted mixed integer non-linear programming problem.

Work [78] suggests bringing SDN to information centric networks (ICN) in order to utilize both of their advantages for future networks. Furthermore, they use deep learning (DL) to predict bandwidth demand based on content name and deep reinforcement learning (DRL) to perform traffic engineering (TE). In this system, cache and bandwidth information is sent by the switches to the controller. The controller then decides the TE state and updates its neural networks accordingly. Static mapping of edge to datacenter is faced with challenges such as capacity limitations, fault intolerance, and failure to keep up with new services and hardware.

Taiji [16] proposes a dynamic connection-aware routing between edge and data centers for Facebook. It routes user requests for dynamic content to the most appropriate data center in order to reduce the query load and network latency. This work is based on the assumption that similar service providers share numerous communities. Therefore, users of the same community are preferably routed to the same data center. Edge to data center routing is conducted with regards to parameters such as edge capacity, utilization, traffic volumes, and edge to data center latency. In order to maintain connection-awareness, social hashing is used, which divides users to buckets of roughly the same size.

Dragon [58] argues that conventional SDN-based TE solutions barely meet diverse and large-scale ISP networks' requirements. Furthermore, the lack of interactions between inter-domain and intra-domain policies leads to sub-optimal performance. Dragon proposes a scalable framework that jointly optimizes intra-domain and inter-domain routes and divides the whole problem into sub-problem, and then attempts to solve them separately to merge the solutions as a whole. This work can also incorporate various objectives such as service

chaining, maximizing the throughput, and guaranteeing fault tolerance.

Work [28] proposes an SD-WAN platform to provide better QoS and load balancing than MPLS-TE and CSPF algorithms. They have utilized the ONOS controller, Segment routing, PCEP, and BGP-LS in this framework. In order to maximize the profit gained by providing services, they propose to maximize the residue bandwidth on the link with the minimum value. In this work, all flows have a constant bit rate and last for a uniform random time, and all the links have a constant capacity.

The above works [7, 78, 16, 58, 28] take the scheduling to the edge of the network, bringing scalability to the system. However, edge scheduling eliminates the possibility of having a centralized control and a global view of the network, and consequently, the uncertainty handling.

Table 2.1 provides an overview and a classification of the seminal works in SD-WANs. Even though these works addressed several issues in the inter-DC networks such as deadlines, multiple destinations, and demand uncertainty, they lack a mechanism to deal with bandwidth uncertainty; therefore, they are not suitable to be utilized for best-effort SD-WANs.

Work	**Objective**	**Deadline-aware**	**SnF**	**P2MP**	**Provisioning**	**Maximum Capacity**	**Uncertainty-aware**	**Admission Control**
[41, 36]	Throughput Fairness	N	N	N	Fixed	Fixed	N	N
[45]	Completed Fraction	Y	N	N	Fixed	Fixed	N	N
[52]	Deadline Guarantee	Y	N	N	Fixed	Fixed	N	Y
[77]	Deadline Guarantee	Y	N	N	Fixed	Fixed	Static	Y
[30]	Cost	Y	Y	N	Dynamic	Fixed	N	N
[51, 29, 59, 50]	Cost	Y	N	N	Dynamic	Fixed	N	N
[48]	Completion Time	Y	Y	N	Fixed	Fixed	N	N
[71]	Completed Fraction	Y	Y	N	Fixed	Fixed	N	N
[61, 60]	Completed Time	Y	N	Y	Fixed	Fixed	N	N
[42]	Welfare	Y	N	N	Dynamic	Fixed	Demand	Y
[74]	Completed Time	Y	N	N	Dynamic	Dynamic	N	Y
[79, 76, 32, 70, 7, 78, 16, 58, 28]	Quality of Service	Y	N	N	Fixed	Fixed	N	Y
Our Work	Profit	Y	N	N	Dynamic	Fixed	Bandwidth	Y

Table 2.1: Works in SD-WANs.

Chapter 3

Bandwidth Prediction

In this chapter, we describe our bandwidth prediction model based on Long Short Term Memory (LSTM) [33] Recurrent Neural Networks (RNN) to find the answer to our first two research questions: "How variable is the bandwidth capacity of the Internet best-effort tunnels?" and "Are we able to predict the capacity of these tunnels?". To this end, we train an LSTM RNN model with offline data and utilize it to make online predictions.

Recent studies show that RNNs can learn temporal patterns so as to make precise predictions for applications with sequential data such as Natural Language Processing (NLP), speech recognition, and time series processing, to name a few [55]. Behaviour of bandwidth in the Internet tunnels demonstrates specific characteristics due to ISP traffic engineering, cross-traffic, and link failure, which brings about an opportunity to LSTM-based bandwidth estimations.

3.1 Bandwidth Prediction Problem

In this section, we formulate the bandwidth prediction problem. The problem consists in estimating the instantaneous bandwidth $b(t)$, n timesteps into the future by observing the

Figure 3.1: The sliding window method to predict the bandwidth.

past m timesteps at timestep τ, which is shown in equation (3.1):

$$\{b(t),\ t = \tau + 1, \cdots, \tau + (n-1), \tau + n\} = f(\{\hat{b}(t),\ t = \tau - m, \cdots, \tau - 1, \tau\}), \qquad (3.1)$$

where $f(\cdot)$ is the estimation function. We have used a Simple Linear Regression (SLR) as a baseline, to compare the results obtained by our LSTM RNN model.

3.2 RNN Model

As mentioned in previous chapters, neural networks are getting more attention in bandwidth prediction due to their ability to adapt to sudden change and making an overall better prediction. This advantage, is in fact, due to the ability of the LSTMs to save the memory of older events as opposed to other simpler methods, which only use a window of data to make predictions. Following a similar architecture as the state-of-the-art [47, 55], we design an LSTM RNN architecture which consists of an input layer, two hidden layers (with 200 and 100 nodes), and an output layer. We follow a sliding window approach wherein every consecutive sequence of multiple past inputs is used to make predictions for a sequence of multiple future outputs. Fig. 3.1 shows an example of how the sequence of inputs is associated with the sequence of outputs in the sliding window approach. It depicts a bandwidth trace from our best-effort tunnel from the campus to a datacenter in the EU. There are five consecutive sequences of 180 seconds input and output windows where the green windows are the inputs and the red ones are the outputs.

3.3 Data Collection

In order to collect data for our bandwidth prediction model, we have established a best-effort tunnel between the University of Calgary ICT building and a Virtual Private Server (VPS) on Microsoft Azure cloud in Amsterdam using the SSH protocol. We used the iPerf3 tool to measure the tunnel's bandwidth using UDP packets with a frequency of 1 second. For parameter tuning of the model, we used a 24-hour-long trace on this tunnel. After the initial results, we found an appropriate model to which we added additional training and testing using a 72-hour-long trace on the same path without any overlap in the data.

3.4 Evaluation

3.4.1 Short-term Predictability

In this section, we study the predictability of bandwidth capacity in best-effort Internet tunnels for short periods (*e.g.*, around 180 seconds) to address our first research question: "How variable is the bandwidth capacity of the Internet best-effort tunnels?". Using statistical analysis, we demonstrate that in short periods, best-effort tunnels' bandwidth capacity does not suffer from high variations and remains within a predictable range. To this end, we have conducted the following studies:

- The effects of window size on variability sequence: We define variability sequence as a sequence of bandwidth measurements in which the value does not deviate more than 20% from the first value in the sequence.
- The effect of window size on confidence band: Confidence band is used in statistical analysis to demonstrate the deviation from an estimated value.

Effects of Window Size on Variability Sequence

In this experiment, we study the effects of moving average window size on variability sequence in short-term predictions. In Fig. 3.2 we show the CDF of the length of variability sequence for our measurements. Having a longer variability sequence is a result of less severe fluctuations. It is evident from the results that an increase in the window size leads to a decrease in variability. For instance, according to Fig. 3.2a, 90% of the sequences have a length of less than 24 when the window size is 1 (*i.e.*, there is no averaging), while approximately 40% of the sequences have a length between 50 and 250 in Fig. 3.2f when the window size is 59. In other words, we are more likely to make an accurate prediction using statistical analysis when the moving average window size is bigger than 59. However, we should note that having a bigger window results in more smooth values and data loss.

Moving Average with Confidence Band

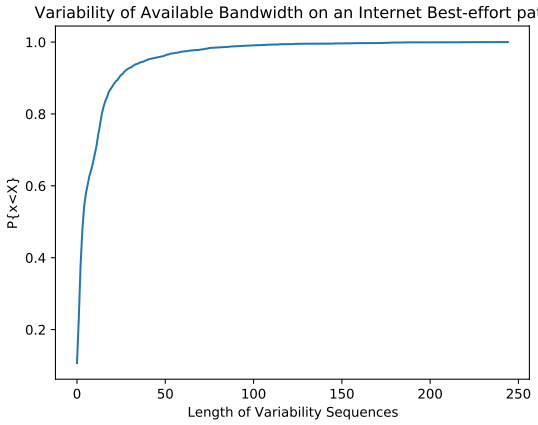
In this section, we calculate the moving average along with a 95% confidence band for our bandwidth measurements in a short period of 15 minutes. The results in Fig. 3.3 show that the confidence band can be moderately tight for various moving average window sizes, which indicates that the bandwidth fluctuates within a predictable range.

3.4.2 Long-term Predictability

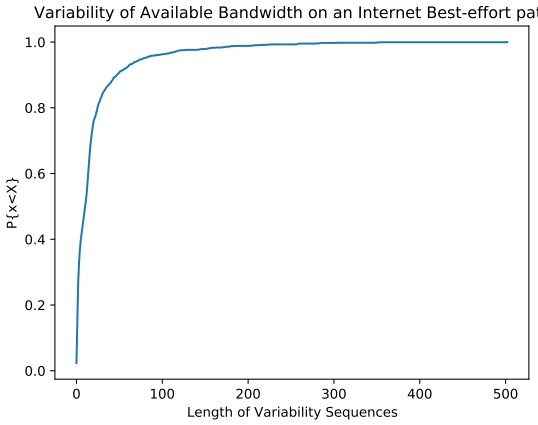
We utilize the LSTM neural network model described in 3.2 to study the long-term predictability of bandwidth in best-effort tunnels, and address our second research question: "Are we able to predict the capacity of these tunnels?".

Parameter Tuning

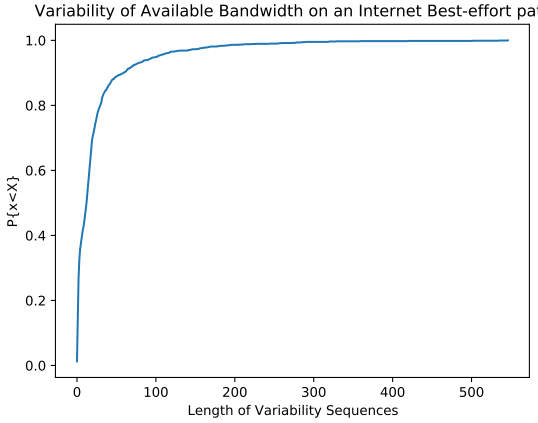
In order to find a suitable set of parameters for our model, we conducted comprehensive experiments on our 24 hour long bandwidth data collected from our best-effort tunnel. The



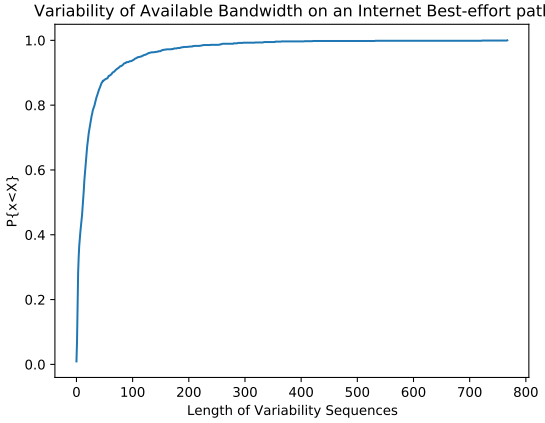
(a) Window size 1.



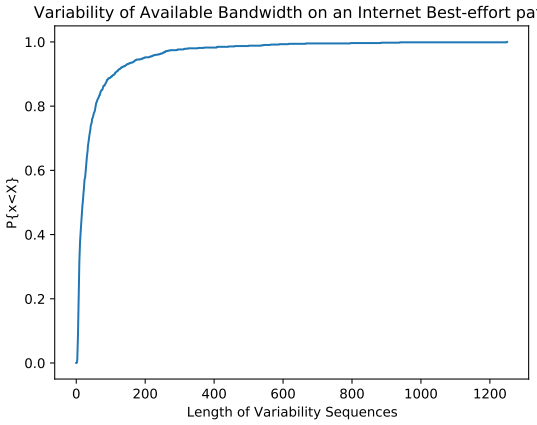
(b) Window size 3.



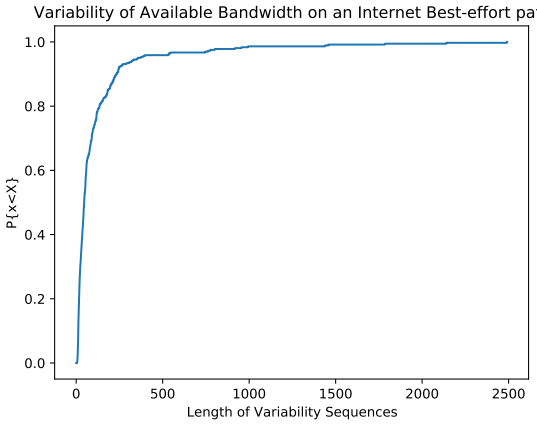
(c) Window size 5.



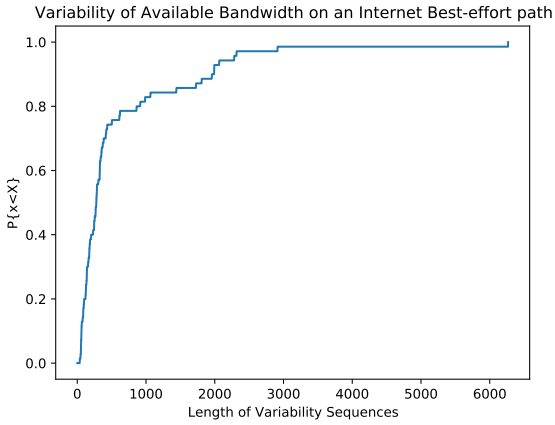
(d) Window size 9.



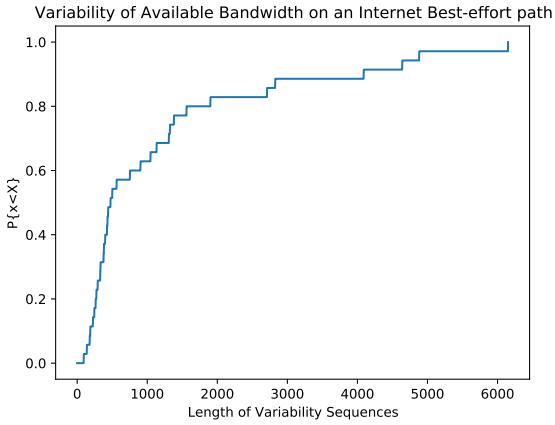
(e) Window size 29.



(f) Window size 59.

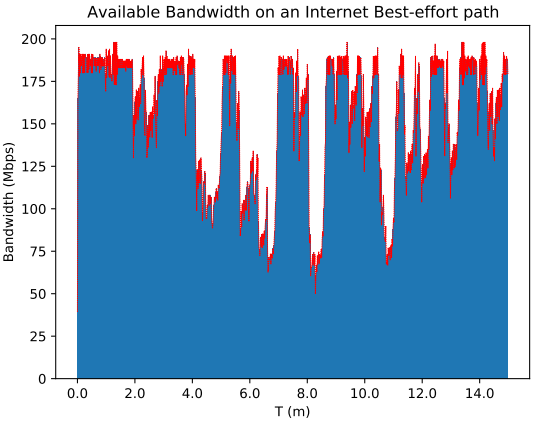


(g) Window size 179.

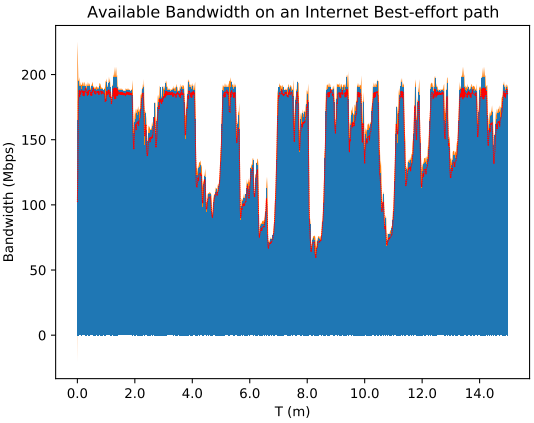


(h) Window size 299.

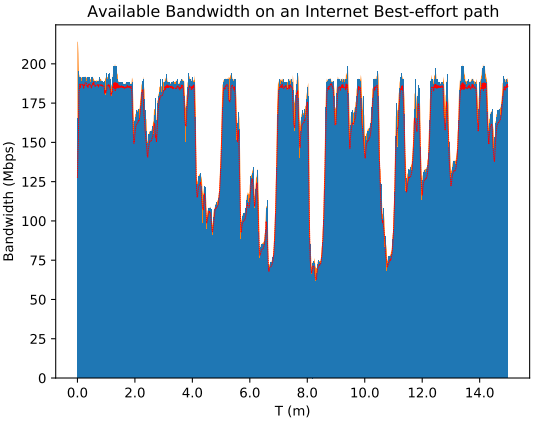
Figure 3.2: CDF of length of variability sequence.



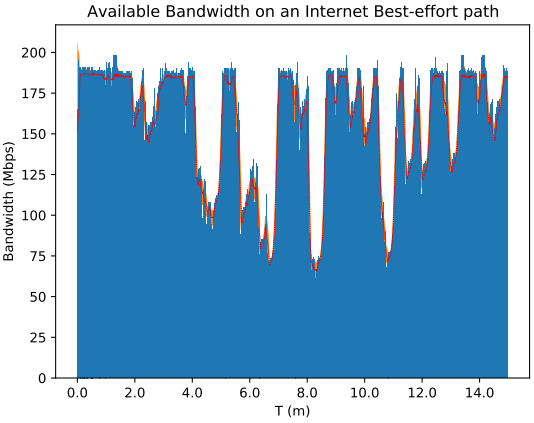
(a) Window size 1.



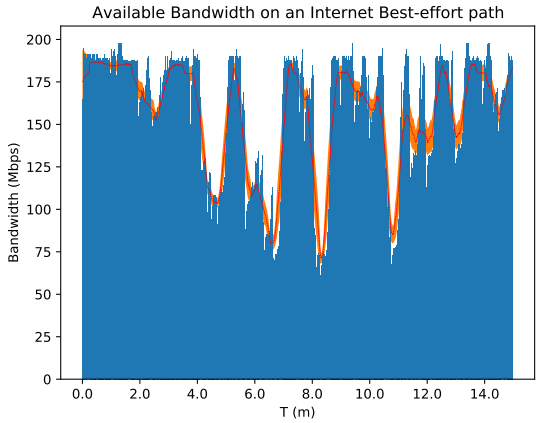
(b) Window size 3.



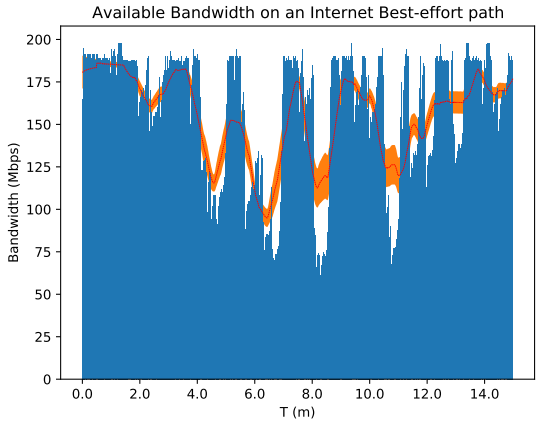
(c) Window size 5.



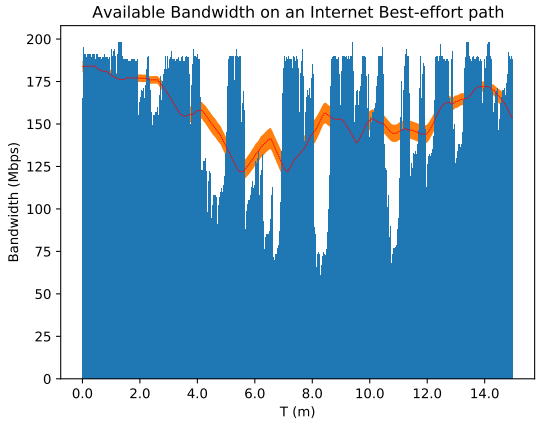
(d) Window size 9.



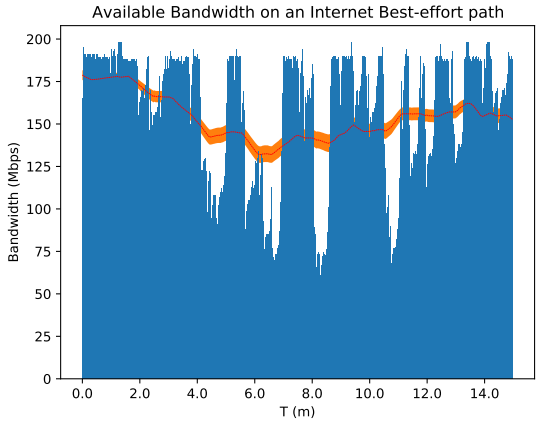
(e) Window size 29.



(f) Window size 59.



(g) Window size 179.



(h) Window size 299.

Figure 3.3: Moving average with confidence band.

data is divided to two subsets of 67% and 33% of the total data for training and testing, respectively. The data is divided in such way to cover peak and non-peak hours in both subsets. In this way, we made sure that both subsets possess the same fluctuation range and have roughly the same average value.

Furthermore, we have studied the effects of a moving average on the bandwidth values to find an acceptable tradeoff between smoother, more predictable values and loss of data. A moving average can mitigate the effects of random, short-term fluctuations. The input dataset has a maximum, minimum, mean, and median of 199.0, 0.0, 154.26, and 169.0, respectively. We have measured the training score and the test score (*i.e.*, RMSE of the

Steps In	Steps Out	Train Score	Test Score
10	10	19.67	20.74
30	10	19.21	20.51
60	10	19.52	20.79
120	10	19.34	20.66
180	10	19.27	20.64
300	10	19.33	20.55
30	30	30.48	32.26
60	30	30.62	31.78
120	30	30.17	32.28
180	30	30.54	31.76
300	30	30.63	31.70
60	60	35.46	36.75
120	60	35.17	35.98
180	60	35.17	36.10
300	60	35.36	36.63
120	120	37.66	38.79
180	120	36.70	38.45
300	120	35.18	37.45
180	180	37.06	39.04
300	180	35.95	39.10
300	300	35.79	39.12

Table 3.1: Scores with a window size of 1.

predicted values and the actual values) for different values of the following parameters:

- Steps-in: The number of values in the sequence of inputs for each prediction.
- Steps-out: The number of values in the sequence of outputs for each prediction.
- Moving Average Window Size: The number of values that are averaged in each subset of the full data set in a series of averages.

As it is evident from the results shown in tables 3.4a to 3.4f, there is a trade-off between the window size of the moving average and the accuracy of the predictions. Table 3.6 also shows the mean absolute percentage error for the moving average dataset as compared to the original dataset. According to the results, a window size of more than 9 leads to an extra 5% error which is not desirable. Furthermore, a window size lower than 9 does not provide a significant advantage over not applying a moving average at all, as shown in tables 3.1 to 3.5.

The results show that having a lower steps-out value yields more accurate predictions, which expected since making predictions too far into the future has a higher probability of

Steps In	Steps Out	Train Score	Test Score
10	10	16.67	17.59
30	10	16.60	17.63
60	10	16.60	17.46
120	10	16.87	17.76
180	10	16.80	17.98
300	10	16.74	17.53
30	30	28.93	30.06
60	30	29.11	30.00
120	30	42.76	40.99
180	30	39.56	39.67
300	30	29.02	29.92
60	60	34.73	35.18
120	60	34.49	35.00
180	60	34.68	35.15
300	60	33.30	34.44
120	120	35.19	38.10
180	120	36.44	38.60
300	120	33.90	36.33
180	180	35.12	38.12
300	180	35.07	38.15
300	300	34.78	38.27

Table 3.2: Scores with a window size of 3.

Steps In	Steps Out	Train Score	Test Score
10	10	14.54	15.37
30	10	14.25	15.03
60	10	14.31	15.06
120	10	14.41	15.21
180	10	14.43	15.35
300	10	26.62	27.04
30	30	27.52	28.51
60	30	27.86	29.01
120	30	27.80	29.20
180	30	27.46	28.58
300	30	27.88	28.79
60	60	33.44	34.10
120	60	33.88	34.31
180	60	35.16	35.69
300	60	33.40	34.60
120	120	35.21	36.72
180	120	34.14	35.78
300	120	35.91	36.54
180	180	34.57	37.64
300	180	34.51	37.67
300	300	34.28	37.75

Table 3.3: Scores with a window size of 5.

Steps In	Steps Out	Train Score	Test Score
10	10	11.52	11.98
30	10	10.75	11.30
60	10	11.00	11.43
120	10	11.629	12.094
180	10	11.03	11.52
300	10	11.12	11.61
30	30	25.40	26.31
60	30	25.36	26.40
120	30	25.64	26.56
180	30	36.20	37.23
300	30	26.48	27.53
60	60	31.94	32.50
120	60	32.53	32.89
180	60	32.59	32.98
300	60	32.29	32.66
120	120	34.17	36.67
180	120	33.74	36.67
300	120	32.19	34.57
180	180	33.57	36.82
300	180	33.56	36.81
300	300	33.32	36.93

Table 3.4: Scores with a window size of 9.

Steps In	Steps Out	Train Score	Test Score
10	10	1.17	1.26
30	10	1.08	1.23
60	10	1.25	1.30
120	10	1.08	1.20
180	10	4.85	5.66
300	10	1.35	1.48
30	30	3.27	3.52
60	30	3.43	3.70
120	30	3.28	3.65
180	30	3.45	3.67
300	30	3.33	3.60
60	60	7.60	7.58
120	60	23.08	21.99
180	60	20.82	21.17
300	60	36.25	32.95
120	120	11.94	12.17
180	120	28.84	25.62
300	120	11.80	12.23
180	180	23.53	22.30
300	180	29.21	25.97
300	300	29.588	26.28

Table 3.5: Scores with a window size of 179.

Window Size	MAPE
1	0.0
3	0.043
5	0.042
9	0.052
29	0.086
59	0.129
179	0.169
299	0.175

Table 3.6: Mean squared error for different window sizes in tuning experiments.

error. Also, having a higher steps-in either has a negative effect on the error or does not provide significant improvements. Consequently, considering the points mentioned above, we have conducted a test on a longer dataset with the previous model using a window size of 1 (*i.e.*, no moving average), and both a steps-in and steps-out value of 180. This value is compatible with the timeslot size of our bulk transfer scheduler and does not apply high computational burden on the system due to high frequency of prediction and scheduling. With regards to epoch count, we observed there were not any significant improvements for values higher than 30; therefore, we choose this number for our prediction experiment.

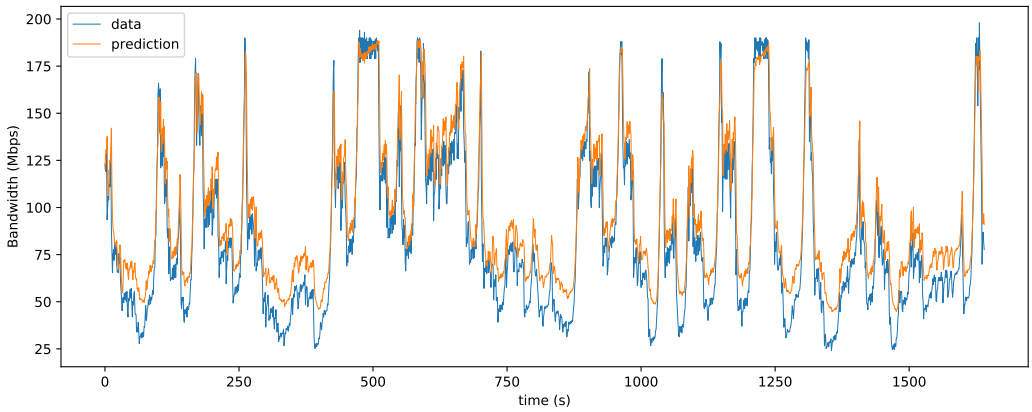
Prediction

In this section, we study the results of our prediction experiments. To this end, we have utilized the model from previous sections to make predictions on multiple 1-hour datasets gathered from our Internet tunnel from the University of Calgary to the Microsoft Azure VPS in Amsterdam at different times. Table 3.7 shows the achieved performance in all experiments in terms of RMSE. Figs. 3.4a to 3.4f also demonstrate the first 2000 predicted samples compared to the data at different times. As it is shown, the model can make a plausible prediction regarding the bandwidth trend with an average 20% MAPE and 18 RMSE, 180 seconds into the future. These results are achieved only with training on a small dataset consisting of 96 hours of data. Throughout time, with more training, the model will be able to make more accurate predictions. However, creating a model to make more

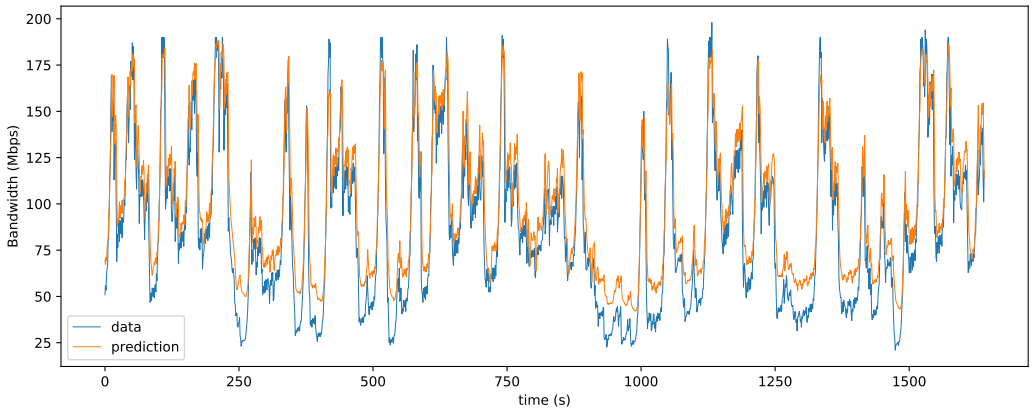
Experiment	**RMSE**
1	61.93
2	63.60
3	18.42
4	12.45
5	15.07
6	14.39

Table 3.7: Mean squared error in the prediction experiments.

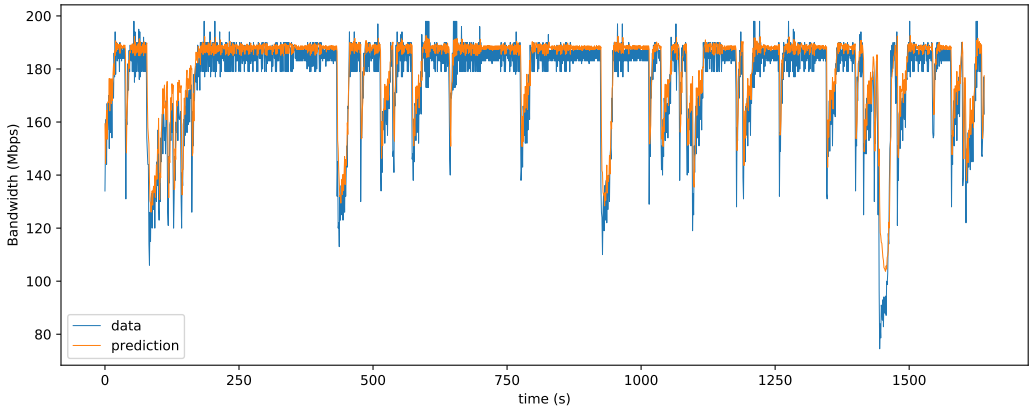
accurate predictions is out of the scope of this thesis.



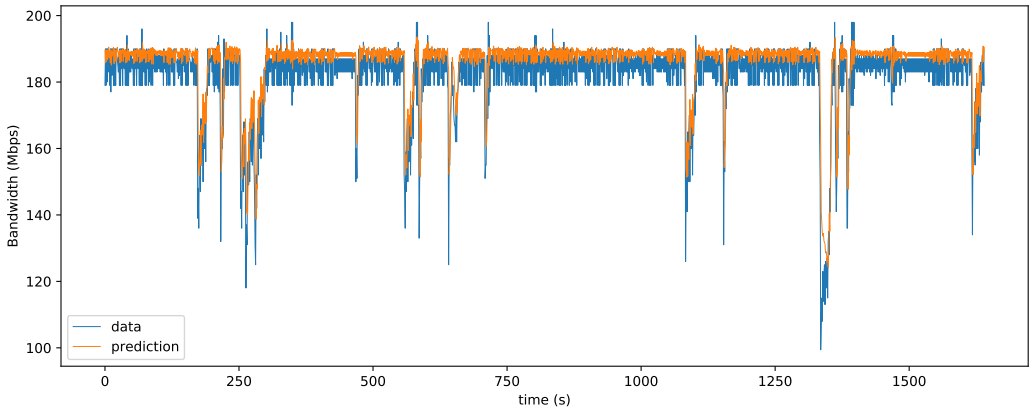
(a) Experiment at 3 P.M.



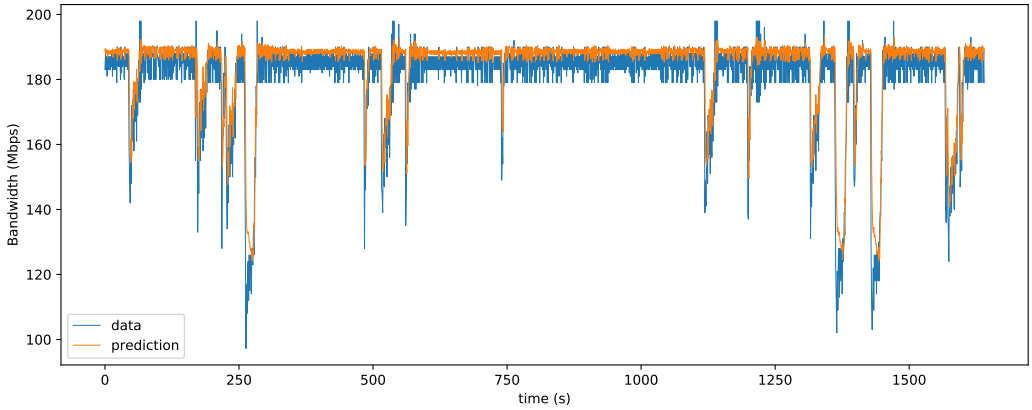
(b) Experiment at 6 P.M.



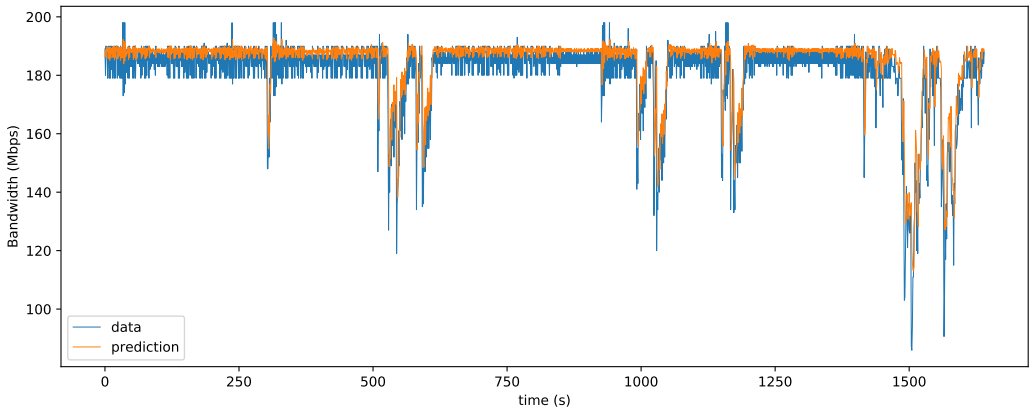
(c) Experiment at 9 P.M.



(d) Experiment at 12 A.M.



(e) Experiment at 3 A.M.



(f) Experiment at 6 A.M.

Figure 3.4: Predictions using the previously trained model.

Chapter 4

Resource Scheduling Problem Formulation

In this chapter, we introduce the system model for the deadline-aware bulk transfer scheduling problem and show how we can formulate this problem using the robust optimization theory.

4.1 System Model

In this section, we describe our system model including the demand model, network model, and capacity fluctuation model.

4.1.1 Demand Model

This thesis focuses on the bulk transfers that transmit large volumes of data and are associated with pertinent delivery deadlines. An example of such transfer is database backup. Previous studies found out that these transfers constitute a significant portion of data transmission on WANs between geographically distributed datacenters. We have adopted the batch model in which bulk transfer requests arrive in batches at every timeslot [45, 52, 23].

In this model, a batch of bulk transfer requests, denoted by R, becomes available in the origin. Furthermore, we have utilized an admission control scheme in which a transfer is either admitted for transmission or it is rejected. Admitted transfer $r \in R$ should transmit B_r bytes of data between timeslots τ_r^1 and τ_r^2 to the destination. Specifically, τ_r^1 is the earliest timeslot that transfer r can start its transmission and τ_r^2 is its delivery deadline. The WAN provider gains the associated profit of U_r units for each transfer upon completely transmitting it before its deadline, otherwise, it does not gain any profit from that transfer. Moreover, $\mathcal{P}_r$ denotes the set of tunnels that request r is allowed to use among all the available tunnels (for security or accounting reasons, a request may not use some of the available tunnels). The sender should only admit from the batch, the requests that can be fully transmitted before their deadlines. Each bulk transfer request i is in the form of a tuple $(B_i, \tau_i^1, \tau_i^2)$ where B_i, τ_i^1, and τ_i^2 denote the request demand, arrival time, and deadline, respectively.

4.1.2 Network Model

Similar to the state-of-the art, in this work time is divided to discreet timeslots and the scheduling is done at the beginning of every timeslot for the batch of requests that arrived during the previous timeslot. We set the duration of each timeslot to 3 minutes in order to leverage the predictability of interactive traffic [36] and reduce the complexity of the optimization problem that we are going to solve at every timeslot. Furthermore, we assume that a set of distinct tunnels, denoted by $\mathcal{P}$, are available between the source and the destination. However, you should note that the *exact* amount of tunnel capacities are not known a priori since they are best-effort tunnels that undergo fluctuations. Thus, we assume that according to the output of our bandwidth predictor, the capacity of tunnel $p \in \mathcal{P}$ in timeslot t fluctuates in an interval given by,

$$[\overline{C}_p(t) - \widetilde{C}_p(t), \overline{C}_p(t) + \widetilde{C}_p(t)], \tag{4.1}$$

where, $\overline{C}_p(t)$ is the expected capacity and $\widetilde{C}_p(t) = \delta\,\overline{C}_p(t)$, for $0 \leq \delta \leq 1$, is the maximum *capacity fluctuation*. Our measurements and state-of-the-art studies demonstrate that it is feasible to obtain these intervals with prediction methods [47]. Since the probability of all the tunnels hitting their lowest capacity (*i.e.*, $\overline{C}_p(t) - \widetilde{C}_p(t)$) in the same timeslot is low, we assume that in each timeslot at most $\Gamma \leq P = |P|$ tunnels will fluctuate maximally from their expected capacity.

4.1.3 Capacity Fluctuation Model

Depending on the bandwidth prediction model, we can define two variations of the uncertain capacity problem.

Path Deviation Model. In the first model, a maximum of Γ paths deviate maximally from their average capacity in each timeslot.

Time Deviation Model. In the second model, each path deviates a maximum of Υ times maximally from its average capacity in a time window (i.e: several timeslots).

4.2 Formulation

In this section, we formulate our problem of deadline-aware admission and scheduling of transfer requests over best-effort SD-WANs. To this end, we admit a set of requests and specify their transmission rates on the available tunnels such that their deadlines are guaranteed, and their sum of profits is maximized. Furthermore, we use linear functions throughout the problem formulation in order to control the complexity of the constructed model. For the sake of space, we only bring the steps to formulate the path deviation model. The time deviation model can be formulated in a similar approach. Table 4.1 provides a list of employed notations and their definitions.

System Profit. The objective of our problem is to maximize the profit that is gained from successfully transmitting the accepted requests on a set of permitted tunnels:

Inputs	**Definition**
$\mathcal{R}$	Set of all requests
$\mathcal{P}$	Set of all tunnels
B_r	Demand (volume) of request r
U_r	Profit gained by successfully transmitting request r
$\overline{C}_p$	Average capacity of tunnel p at timeslot t
$\widetilde{C}_p(t)$	Deviation of tunnel p's capacity from its average
Γ	Maximum number of tunnels that deviate from their maximum capacity
δ	Ratio of maximum capacity to average capacity
τ_r^1	Arrival time of request r
τ_r^2	Deadline of request r
Variable	**Definition**
$x_p^r(t)$	Transmission volume of request r at timeslot t over tunnel p
a_r	Admission status of request r

Table 4.1: Important Mathematical Notations

$$\text{Max.} \sum_{r \in \mathcal{R}} a_r \times U_r. \tag{4.2}$$

Scheduling Bulk Transfers. The decision variable $x_p^r(t)$ is defined to compute the *transmission rate* of request r over tunnel p in timeslot t. The following constraints ensure that each request is allowed only to use a set of permitted tunnels:

$$x_p^r(t) \geq 0, \quad \forall p \in \mathcal{P}_r, t, r \tag{4.3}$$

$$x_p^r(t) = 0. \quad \forall p \notin \mathcal{P}_r, t, r \tag{4.4}$$

Path Capacity Constraints. In each timeslot t, the total transmission rate on each tunnel

p should be less than or equal to its expected average capacity to avoid overloading of the tunnels,

$$\sum_{r \in \mathcal{R}} x_p^r(t) \leq \overline{C}_p(t). \quad \forall t, p \tag{4.5}$$

Deadline Guarantee. The Binary decision variable a_r indicates whether request r is accepted or not. For each accepted request r, the accumulated data transmission between its arrival time τ_r^1 and its deadline τ_r^2 should be equal to B_r, the demand of request r. In this way, we can guarantee that each admitted transfer request is transmitted fully before its associated deadline. This constraint is formulated as,

$$a_r \times B_r \leq \sum_{t \in [\tau_r^1, \tau_r^2]} \sum_{p \in \mathcal{P}_r} x_p^r(t) \times \Theta, \quad \forall r \tag{4.6}$$

where, Θ is the length of a timeslot that is utilized to compute transfer volumes from transfer rates.

Robust Formulation. In order to account for capacity uncertainty, we propose that the total transmission on all tunnels should be less than or equal to the available bandwidth in each timeslot, regardless of which paths experience a capacity fluctuation compared to the estimated values. To this end, we define the following constraint,

$$\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} x_p^r(t) \leq \sum_{p \in \mathcal{P}} \overline{C}_p(t) - \max_{\substack{\pi(t) \subseteq \mathcal{P}, \\ \pi(t) \leq \Gamma}} \sum_{p \in \pi(t)} \widetilde{C}_p(t) \quad \forall t. \tag{4.7}$$

In this equation, $\pi(t)$ is the set of paths that deviate maximally, or in other words, hit their lowest capacity in timeslot t. In this notation, $\Gamma = 0$ is the perfect scenario where all the estimations are accurate and $\Gamma = P$ is the worst-case scenario where all capacities deviate maximally from their expected value at every time-slot. Note that we are not aware of the value of $\pi(t)$ at the moment of scheduling. Instead, we initialize it with the tunnels that lead to the worst-case scenario using the optimization process (*i.e.*, the employed max

operator). Therefore, constraint (4.7) allows us to be prepared for the worst-case scenario that might prevail within the boundary of our assumptions about the accuracy of estimated values.

However, the max operator in this problem is non-linear and leads to complications in the optimization process. In order to linearize constraint (4.7), we utilize a similar approach to [10] and extract the non-linear term and write it as a separate program. Consequently, we have:

$$\text{Max.} \sum_{p \in \mathcal{P}} z_p(t) \times \widetilde{C}_p(t), \tag{4.8}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}} z_p(t) \leq \Gamma, \tag{4.8a}$$

$$0 \leq z_p(t) \leq 1, \tag{4.8b}$$

where, $z_p(t)$ is a decision variable that indicates whether path p deviates maximally in timeslot t and Γ restricts the number of fluctuating tunnels. Then, by defining two dual variables λ_t and ν_t^p, respectively, associated with constraints (4.8a) and (4.8b), we calculate the dual of the linear program (4.8), which is another linear program itself. The dual linear program is as follows:

$$\text{Min.} \ \lambda_p \times \Gamma + \sum_{p \in \mathcal{P}} \nu_p^t, \tag{4.9}$$

$$\text{s.t.} \ \widetilde{C}_p(t) \leq \lambda_p + \nu_p^t, \tag{4.9a}$$

$$0 \leq \lambda_p, 0 \leq \nu_p^t. \tag{4.9b}$$

Next, we can replace the non-linear term of constraint (4.7) with the objective of (4.9) and include constraints (4.9a) and (4.9b) in the original problem formulation. Specifically,

we replace constraint (4.7) with the following constraints:

$$\sum_{r\in\mathcal{R}}\sum_{p\in\mathcal{P}_r} x_p^r(t) \leq \sum_{p\in\mathcal{P}} \overline{C}_p(t) - \lambda_p \times \Gamma - \sum_{p} \nu_p^t \tag{4.10}$$

$$\widetilde{C}_p(t) \leq \lambda_p + \nu_p^t, \tag{4.11}$$

$$0 \leq \lambda_p, 0 \leq \nu_p^t. \tag{4.12}$$

Objective (4.2) along with constraints (4.3)-(4.5) and (4.10)-(4.6) define the problem of deadline-aware scheduling of bulk transfers over best-effort Internet tunnels, which is shown in Algorithm 1.

The time deviation model can be formulated in a similar approach. We replace constraint (4.7) with (4.13) as follows:

$$\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}} x_r(p) \times h_r(t) \leq \sum_{t} \overline{C}_p(t) - max \sum_{\substack{t\in\tau(t),\\ \pi(t)\leq\Gamma_2(p)}} \widetilde{C}_p(t), \forall p \tag{4.13}$$

Then, we utilize the same approach we used to linearize constraint (4.7) in order to derive

constraint (4.10).

Algorithm 1: Bulk Transfer Scheduling with Deadline on Best-effort SD-WANs

$$\max \sum_{r \in \mathcal{R}} U_r \times a_r$$

subject to

$$\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} x_p^r(t) \leq \sum_{p \in \mathcal{P}} \overline{C}_p(t) - \lambda_p \times \Gamma - \sum_p \nu_p^t$$

$$a_r \times B_r \leq \sum_{p \in \mathcal{P}} \sum_{t \in [\tau_r^1, \tau_r^2]} x_p^r(t), \quad \forall r$$

$$\widetilde{C}_p(t) \leq \lambda_p + \nu_p^t$$

$$0 \leq \lambda_p,$$

$$0 \leq \nu_p$$

4.2.1 Exact Algorithm

In this section, we present our exact algorithm for bulk transfer scheduling with deadline over best-effort SD-WANs.

Algorithm 2: Exact Algorithm

Input:

$B(t) = b_1, b_2, ..., b_n$: A batch request with n bulk transfer requests at timeslot t;

$P(t) = p_1, p_2, ..., p_m$: Set of m available tunnels between source and destination;

$C(t) = c_{p_1}(t_1), c_{p_2}(t_1), \cdots, c_{p_m}(t_1), \cdots, c_{p_m}(t_\tau)$: Matrix of expected capacities over the next τ timeslots for all tunnels;

Γ : Expected number of tunnels that experience worst-case deviation;

δ : Maximum deviation from expected value

Output:

$A = a_1, a_2, ..., a_n$: Set of admission status of all transfer requests $B(t)$

$X_p^r(t) = x_{p_1}^{r_1}(t_1), c_{p_2} r_1(t_1), \cdots, c_{p_m} r_1(t_1), \cdots, c_{p_m} r_{m-1}(t_\tau), c_{p_m} r_n(t_\tau)$: Matrix of transmission rate of each transfer request over τ timeslots for all tunnels;

$$\max \sum_{r \in \mathcal{R}} U_r \times a_r$$

subject to

$$\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} x_p^r(t) \leq \sum_{p \in \mathcal{P}} \overline{C}_p(t) - \lambda_p \times \Gamma - \sum_p \nu_p^t$$

(4.9a), (4.9b), (4.6)

Algorithm 2, however, is computationally demanding and time-consuming due to the integrality of some of its variables, which makes it impractical for big problems.

4.2.2 Approximate Algorithm

In order to mitigate the complexity issue of our exact algorithm, we develop an approximate algorithm based on the iterative rounding technique. To this end, we remove the integrality of a_r variables, which yields an optimal fractional solution in polynomial time. Then, we run a greedy algorithm that iteratively chooses a request with the highest profit to demand ratio (*i.e.*, $\rho_r = \frac{U_r}{B_r}$)), round a_r to 1, and determine whether the network capacity allows this admission. Please note that we choose the request with the highest demand among the requests with the same profit to volume ratio (B_r). If rounding to 1 leads to an infeasible

solution, we round it to 0. Algorithm 3 shows our approximate solution.

Algorithm 3: Approximate Algorithm

```
1  procedure xBESD()
2      M ← MIP()                                              /* create integer model */
3      M̃ ← relax(M)
4      A ← {}                                                 /* accepted requests */
5      J ← {}                                                 /* rejected requests */
6      status, {ã_r}, {x_p^r(t)} ← solve(M̃)
7      while True do
8          foreach r ∈ R − A ∪ J and ã_r = 1 do
9              ã_r.lower_bound ← 1                            /* fix the decision variable to 1 */
10             A.append(r)
11         foreach r ∈ R − A ∪ J and ã_r = 0 do
12             ã_r.upper_bound ← 0                            /* fix the decision variable to 0 */
13             J.append(r)
14         if A ∪ J ≠ R then
15             r* ←_{r∈R−A∪J} {ρ_r, B_r}
16             ã_{r*}.lower_bound ← 1
17             status, {ã_r}, {x_p^r(t)} ← solve(M̃)
18             if status = INFEASIBLE then
19                 ã_{r*}.lower_bound ← ã_{r*}.upper_bound ← 0
20                 J.append(r*)
21                 status, {ã_r}, {x_p^r(t)} ← solve(M̃)
22             else
23                 A.append(r*)
24         else
25             return {ã_r}, {x_p^r(t)}
```

In the following, we explain the approximate algorithm in detail. Please note that we have used tilde to indicate relaxed variables. First, the algorithm constructs the problem; then it starts by relaxing the integrality constraint of a_r variables, which is shown in line 3. Furthermore, lists A and J are created to store the accepted and rejected requests, respectively. Then the relaxed model is solved in line 6, and an optimal fractional solution is obtained. Subsequently, a while loop computes a feasible integral solution iteratively in lines 7 through 25. In each iteration, first, all requests whose values are already 1 and 0 are directly added to the accepted and rejected sets in lines 8 and 11, respectively. Then, in lines 15 through 23, each variable is first rounded to 1 by setting its lower bound to 1, then the problem is solved, and if the solution is feasible, the relevant request is added to the accepted list. Otherwise, if the solution is infeasible, the variable is rounded to 0 by setting its upper bound to 0 and the request is added to the rejected list. Then the problem is solved again, and the solution is stored. Finally, when no requests remain that are not in either of the accepted or rejected sets, the solution is returned in line 25.

Chapter 5

Analysis

In this chapter we first provide theoretical analysis to evaluate the run-time complexity and approximation ratio of our approximate algorithm. Then we provide our simulation and experiment results.

5.1 Theoretical Analysis

In theorem 5.2, we prove that our approximate method, specifically selecting requests, has an approximate ratio that depends on the number of requests with an equal profit to demand ratio.

Theorem 5.1. *The approximate algorithm runs in* $O(|R| \times (|R||P|T)^{3.5})$.

Proof. Since each iteration, we subtract at least one request from the remaining requests and add it either to the accepted requests or to the rejected ones, the while loop terminates after at most $|R|$ iterations. This is shown in lines 23 through 20) respectively. Moreover, we solve at most two linear programs with $O(|R| \times |P| \times T)$ decision variables in each iteration, where T is the length of the data transmission period from the smallest τ_r^1 to the largest τ_r^2. In the worst-case scenario, the interior point method solves a linear program in $O(n^{3.5})$, where n is the number of decision variables. □

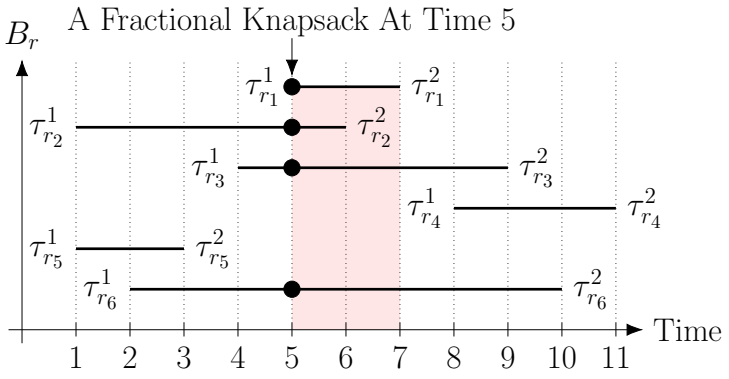


Figure 5.1: Effect of requests on bandwidth allocation of each other.

Theorem 5.2. *The approximation ratio of the algorithm is at most* $\Psi - 1$*, where* Ψ *is the maximum number of requests that overlap in a timeslot and their* ρ_r *(i.e., profit to demand ratio,* $\rho_r = \frac{U_r}{B_r}$ *) is equal.*

Proof. In order to accept r_1 with $\widetilde{a}_{r_1} \in (0, 1)$, we have to reject a set of requests $\overline{R}_{r_1}$, whose size is at most $\Psi - 1$. Fig. 5.1 shows that the total network consumption of set $\overline{R}_{r_1}$ in the $[\tau^1_{r_1}, \tau^2_{r_1}]$ period is at least $(1 - a_{r_1})B_{r_1}$. If $\overline{R}_{r_1}$ is empty, it is not possible to accept the request r_1; therefore, we set $a_{r_1} = 0$ and solve the fractional solution again. Also, we can take bandwidth from one request and give it to another one since none of the requests are complete yet. In each timeslot the problem is similar to a fractional knapsack, where more bandwidth is allocated to requests with a higher profit to volume ratios (*i.e.*, $\rho_r = U_r/B_r$). Furthermore, all admitted incomplete requests in a timeslot have an equal profit to volume ratio since the solution is optimal, and there is no possibility for increasing the objective value by admitting a request with higher profit to volume ratio. Our problem, however, differs from a fractional knapsack in that the allocated bandwidth to each request is determined by the solution and is not known a priori. As a result, we cannot apply a relevant solution to our problem. Moreover, by accepting r_1, the rounded solution gains U_{r_1}, while the optimal solution can gain, at most, $\sum_{r' \in \overline{R}_{r_1}} U_{r'}$. In order to bound the ratio $\frac{\sum_{r' \in \overline{R}_{r_1}} U_{r'}}{U_{r_1}}$, we define r'' to be the request in $\overline{R}_{r_1}$ with the maximum $B_{r''}$. An upper bound for the approximation ratio is

$$\frac{\sum_{r' \in \overline{R}_{r_1}} \rho_{r'} B_{r'}}{\rho_r B_{r_1}} \leq \frac{\sum_{r' \in \overline{R}_{r_1}} B_{r''}}{B_{r_1}} = (\Psi - 1)\frac{B_{r''}}{B_{r_1}} \tag{5.1}$$

We have $B_{r''} \leq B_{r_1}$, which leads to $\frac{B_{r''}}{B_{r_1}} \leq 1$. Therefore, the rounding technique with the descending order of transfer volume has an approximation ratio of at most $\Psi - 1$. □

5.2 Experimental Analysis

Methodology

We used simulations and Mininet experiments to evaluate our proposed algorithm. The simulations are focused on evaluating the performance of our algorithm compared to baseline algorithms under different circumstances. Furthermore, the Mininet experiments intend to evaluate our algorithm's performance in a realistic network environment. We use **the acceptance rate of the transfers**, **the total profit gained**, and **run-time in various scenarios** for our performance metric.

Algorithms

State-of-the-art algorithms have a reactive approach with regards to capacity fluctuations; therefore, they fail to leverage the extra profit gained by utilizing the cheaper best-effort tunnels. On the other hand, our algorithm considers the fluctuations and proactively schedules bulk transfers on the best-effort tunnels in such a way as to maximize the profit. Consequently, we chose two baseline algorithms to which we compared our result.

1. **Average Algorithm:** The average algorithm, denoted by AVG, takes into account only the predicted average capacities and schedules requests based on those values. It goes without saying that AVG does not consider capacity fluctuations.

2. **Effective Bandwidth Algorithm:** The **E**ffective **B**andwidth algorithm, denoted by

EB, calculates an effective bandwidth value for each tunnel somewhere between the average capacity and the maximum capacity fluctuation. Depending on how conservative we are on calculating the effective bandwidth, it can be either closer to the average bandwidth or to the maximum fluctuation. We used the algorithm in [46] to implement EB with three percentiles of 90, 95, and 99, with 99 being the most conservative (*i.e.*, closer to the maximum fluctuation) and 90 being the least conservative (*i.e.*, closer to the average capacity).

5.2.1 Simulation Experiments

Setup

In our simulation experiments, we used a topology that consists merely of a source and a destination to focus on the performance of point-to-point scheduling. However, our algorithm can also be utilized for more complicated topologies. In different experiments, we evaluate the performance of all algorithms for different values of several parameters, including the number of paths, the duration of the deadline, the number of requests, maximum capacity deviation, and frequency of fluctuations (*i.e.*, number of fluctuating paths). We have repeated each experiment 33 times and used a 95% confidence interval to present the results.

The default values for all other parameters are as follows. The number of end-to-end tunnels is set to 10. These tunnels have an average capacity in the range $[50, 200]$ Mbps with a maximum fluctuation of 40% from the average. Our measurements demonstrate a fluctuation range of 15% to 20%; however, it may reach up to 50% due to factors such as traffic shaping and link outages [47, 68, 75]. We set Γ to 7 and Υ to 35 for the tunnel deviation model and the time deviation model, respectively. The requests' start time is modeled as a Poisson process with an arrival rate of $\lambda = 4$ request per second. Also, the requests' deadline is modeled as an exponentially distributed random variable with a mean of 10 timeslots from their starting time. We generated the utility associated with each request

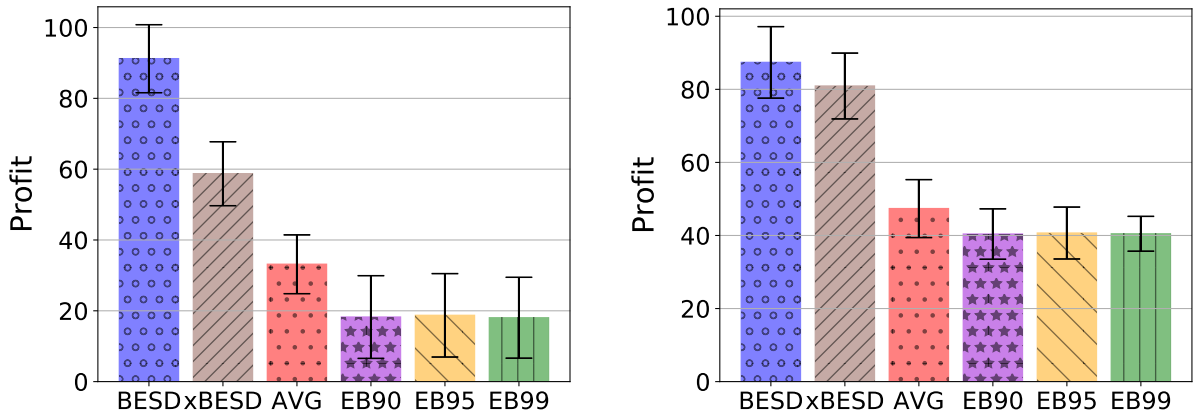
as a random variable with uniform distribution in range $[1, 10]$. We set each timeslot duration to be 3 minutes and the total transmission period to be 50 timeslots. We implemented all the simulations in Python 3 environment on a machine with an Intel® Core™ i7-8700 processor at 3.20 GHz and 8 GB of RAM.

Results

First, we provide an overview of the performance of our algorithm under the aforementioned general setup. In this overview, we demonstrate using Fig. 5.2 that our algorithm achieves better profit than its counterparts. At the same time, Fig. 5.2b shows that the difference between **B**ulk transfer **S**cheduling with **DE**adline over best-effort SD-WANs (BESD) and appro**x**imate-BESD (xBESD) is more subtle in the time deviation model. Nevertheless, xBESD still manages to outperform baseline algorithms by a significant margin in both models. Concerning the run-time, all algorithms perform similarly within a range of $[7, 13]$ seconds, while our approximate algorithm has a slight advantage over the baseline algorithms, as shown in 5.3. This run-time is reasonable for our case of scheduling bulk transfers where each transfer takes approximately 30 minutes to finish, and we run the algorithm once every 150 minutes for a large number of transfers. Afterwards, we provide micro-benchmarks, each of which focuses on a different aspect of the algorithm or the environment in order to provide a thorough evaluation of the performance. Since our exact algorithm is computationally hard and has a long run-time, we focus on our approximate algorithm, which has a similar performance with a considerable computational and run-time advantage.

Effect of Number of Paths

Figs. 5.4 and 5.5 demonstrate the benefits of our proactive algorithm for a different number of paths with a fixed total network capacity. We set Γ to be 1, 2, 3, 7, and 10 for 1, 3, 5, 10, 15 number of paths, in the path deviation experiment, as well as we set Υ to be 35 for the time deviation experiment. For one path, under both models, our algorithm performs



(a) Path deviation model.

(b) Time deviation model.

Figure 5.2: Profit of different algorithms.

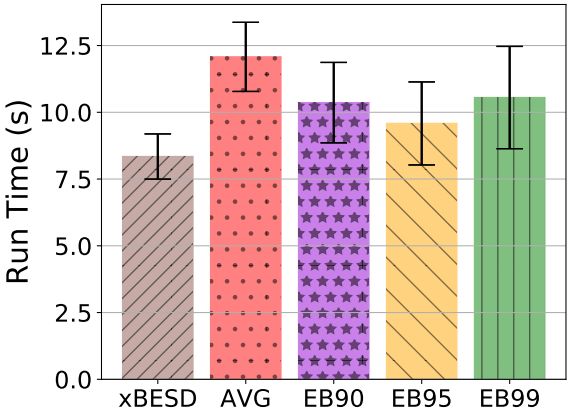
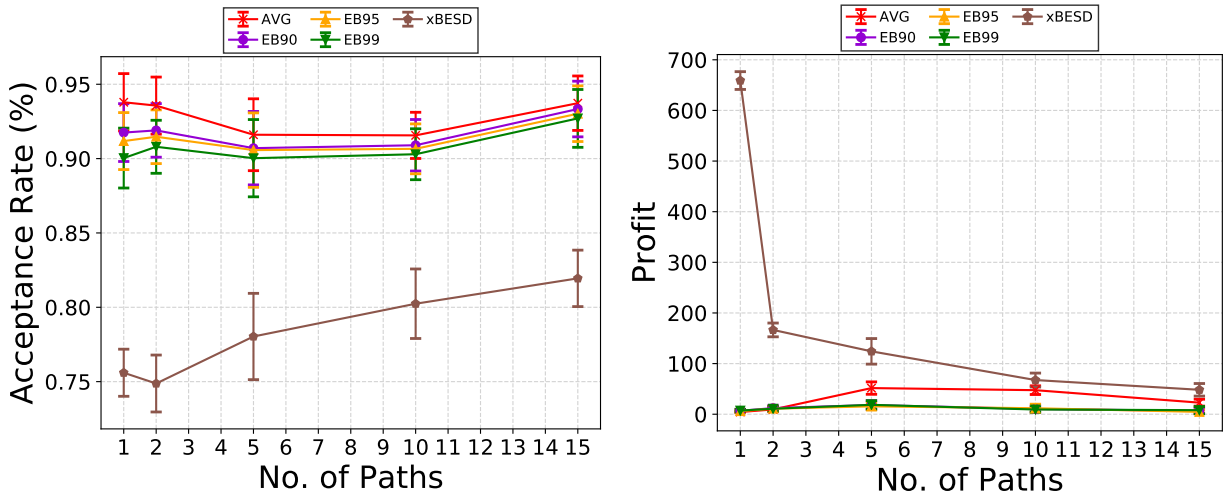


Figure 5.3: Run-time of different algorithms.

close to optimal since there is only one path and our algorithm is aware of its fluctuations, while the other algorithms fail to transmit many transfers before their deadline successfully; therefore, they gain low to no profit. Specifically, we observe a very high gained profit in the path deviation model due to our algorithm's considering the worst-case scenario. For our algorithm, we notice a slight upward trend in the acceptance rate by increasing the number of available paths, whereas there is a downward trend for the profit. Exhibiting the downward trend arises from an increased misprediction probability due to a higher number of paths and partial information about the fluctuations. In the path deviation experiment, we notice an upward trend in the baseline algorithms' profit since they no longer suffer from



(a) Effect on acceptance rate.

(b) Effect on profit.

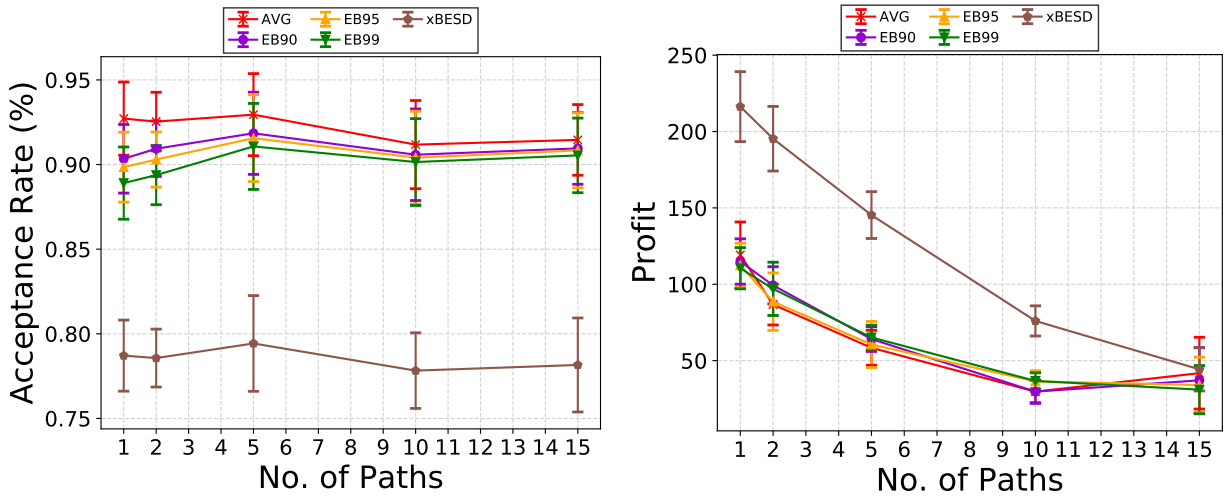
Figure 5.4: Effect of number of paths in path deviation model.

constant fluctuations. On the other hand, in the time deviation model, a similar trend to our algorithm is demonstrated. In summary, our algorithm shows superior performance over the baseline for path values in range 1 to 15, which indicates the advantage of our robust formulations.

By increasing the number of paths beyond this point and to the infinity, while our algorithm's gained profit will converge to zero, the baseline algorithms' gained profits will converge to the optimal value. However, having a large number of paths is very costly and defeats the purpose of our algorithm, which is reducing the expenses of the network.

Effect of Number of Requests

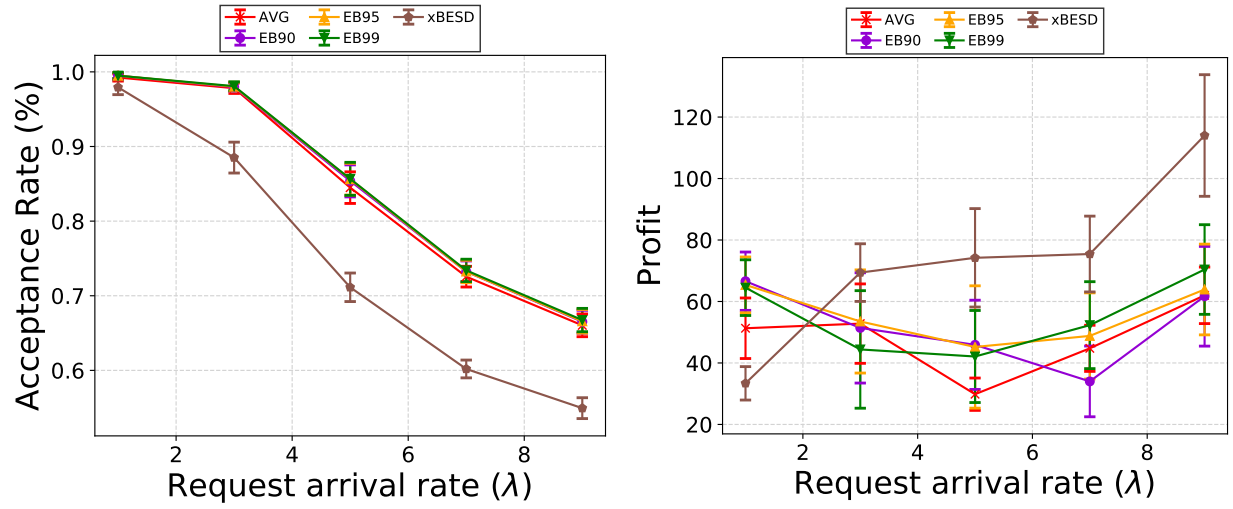
Figure 5.6 and 5.7 show the behavior of different algorithms as the transfer request rate increases. Under both experiments, algorithms show a decrease in the acceptance rate, which implies that the paths are already close to capacity saturation with the current environment values. On the other hand, with more requests arriving in the system, there is a higher chance of admitting a request with higher profit and lower demand due to a more varied selection, resulting in an upward trend in our algorithm's profit.



(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.5: Effect of number of paths in time deviation model.



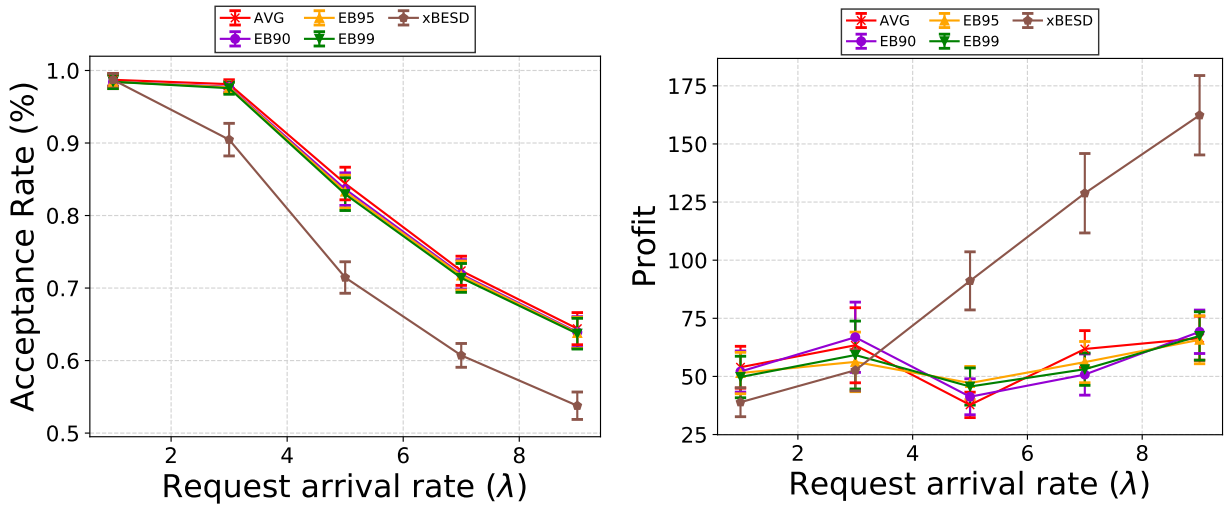
(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.6: Effect of number of requests in the path deviation model.

Effect of Request Deadline

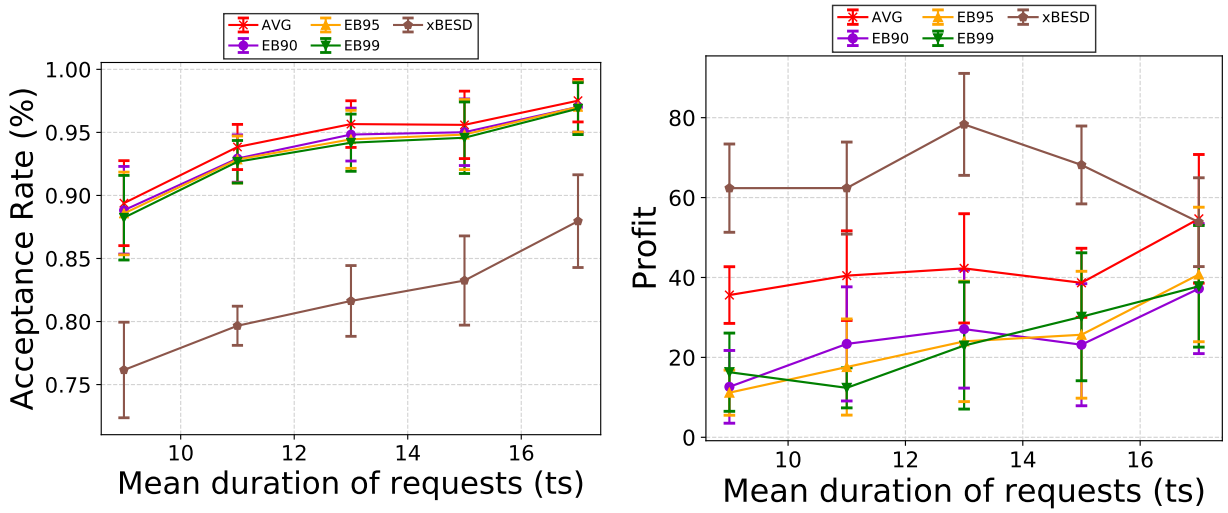
Figs. 5.8a and 5.9a show that the acceptance rate of all algorithms increases as the duration of requests increases. For the baseline algorithms that do not consider fluctuations (*i.e.*, AVG Algorithm) or take conservative decisions about them (*i.e.*, EB Algorithm), the increased acceptance rate results in an upward trend in the profit, which results from loose deadlines that allow a higher number of requests to finish before their associated deadlines on non-



(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.7: Effect of number of requests in the time deviation model.

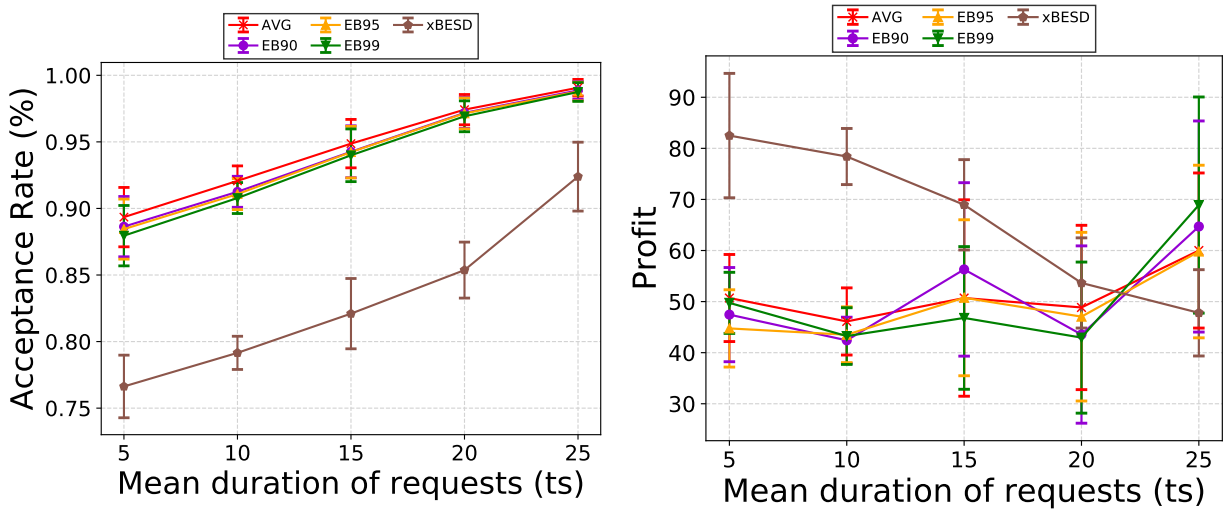


(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.8: Effect of request deadline in path deviation model.

fluctuating paths or when fluctuations do not occur. On the other hand, our algorithm suffers from more mispredictions when a request stays longer in the system, hence, the downward trend. However, to avoid repeated rejection of the requests with longer deadlines, the relevant profit value can be modified according to the time the request has spent awaiting admission.



(a) Effect on acceptance rate.

(b) Effect on profit.

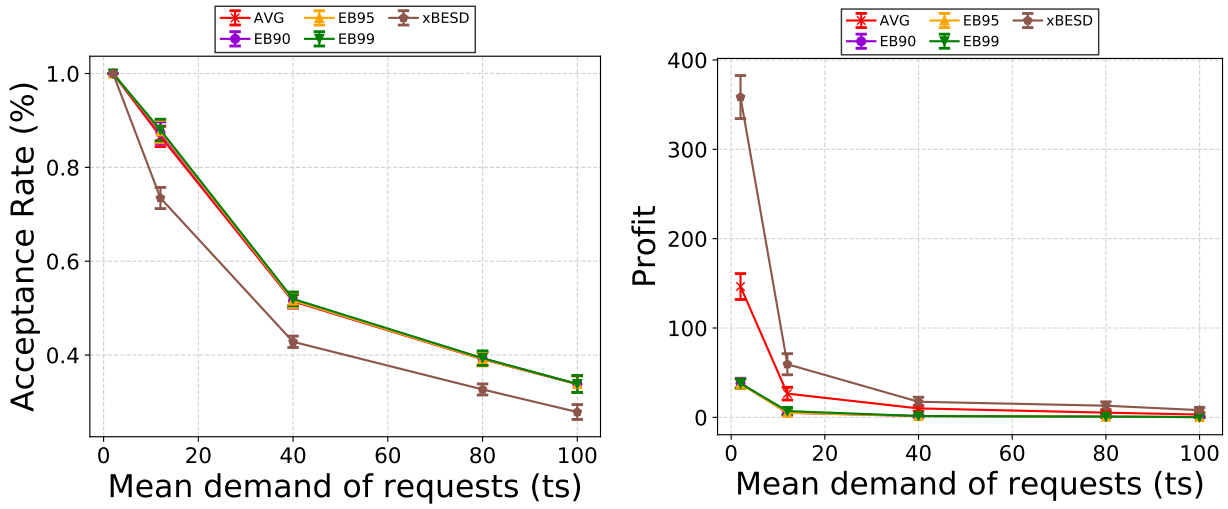
Figure 5.9: Effect of request deadline in time deviation model.

Effect of Request Demand Volume

In both experiments, as we increase the mean demand of requests, we observe a drop in the profit in Figs. 5.10b and 5.11b. The drop in the profit is anticipated because, as shown in Figs. 5.10a and 5.11a, fewer requests are admitted; therefore, less profit is gained from their transmission.

Effect of Capacity Fluctuations

In this scenario, we study the effect of maximum capacity deviation from the estimated values. We let δ to be 10%, 20%, 30%, 40%, and 50% to demonstrate the effect of small and large fluctuations. In Fig. 5.12a, we observe that while the baseline algorithms do not react to the severity of the fluctuations in the network, our algorithm admits fewer requests in such a situation. Furthermore, Fig. 5.12b shows that our algorithm outperforms other algorithms in terms of profit under different fluctuations.



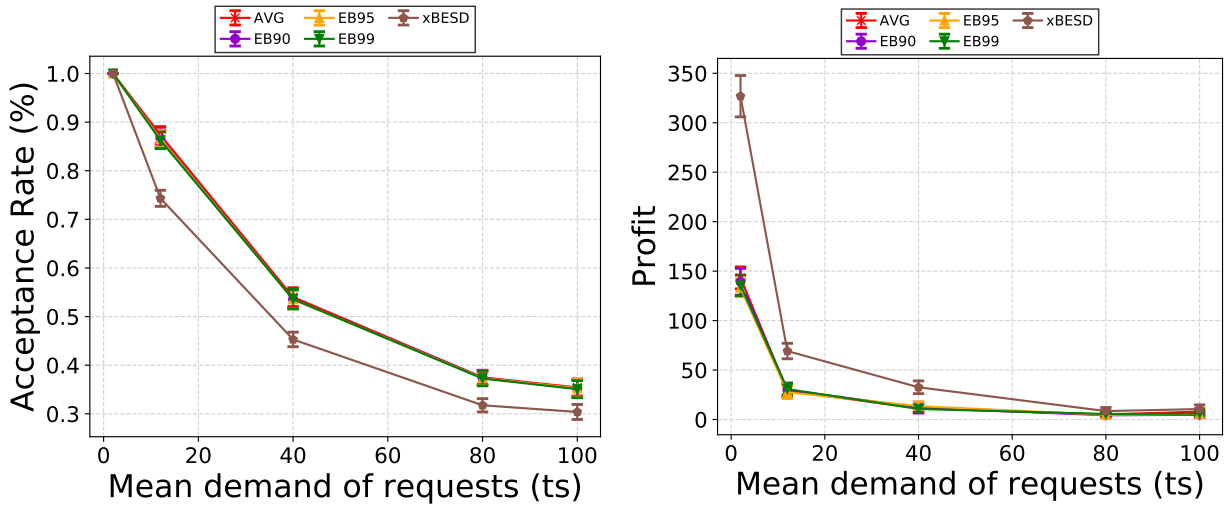
(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.10: Effect of request demand in the path deviation model.

Effect of Number of Fluctuations

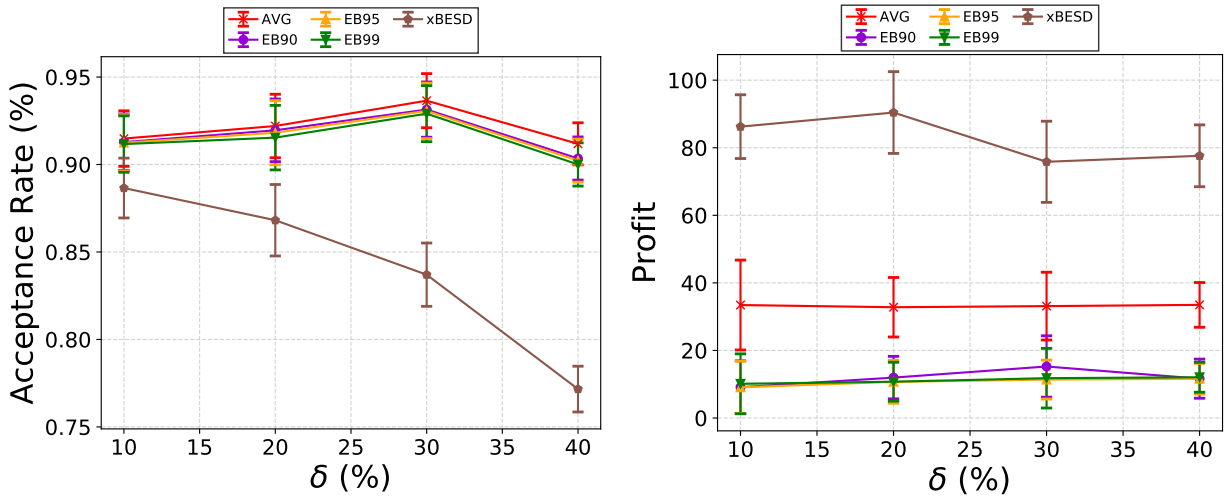
In this scenario, we study the effect of different values of Γ (*i.e.*, number of paths whose capacity fluctuate in a timeslot) and Υ (*i.e.*, number of times a path's capacity fluctuates in a transmission period) on the performance of our algorithm. In the first experiment, we set the number of available paths to 10, and increase Γ from 7 to 10. The paths with capacity fluctuations are chosen in such a way to account for the worst-case scenario (*i.e.*, paths with the lowest capacities fluctuate and hit their maximum deviation). Fig. 5.14a shows that our algorithm admits fewer requests when this number increases. Our algorithm outperforms the other algorithms significantly in terms of profit, which is shown in Fig. 5.14b. In the next experiment, we increase Υ from 35 to 50 and observe the behaviour of different algorithms. Similar to the previous experiment, Figs. 5.15a and 5.15b show that our algorithm admits fewer requests and successfully transmits a higher portion of them compared to baseline algorithms.



(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.11: Effect of request demand in the time deviation model.



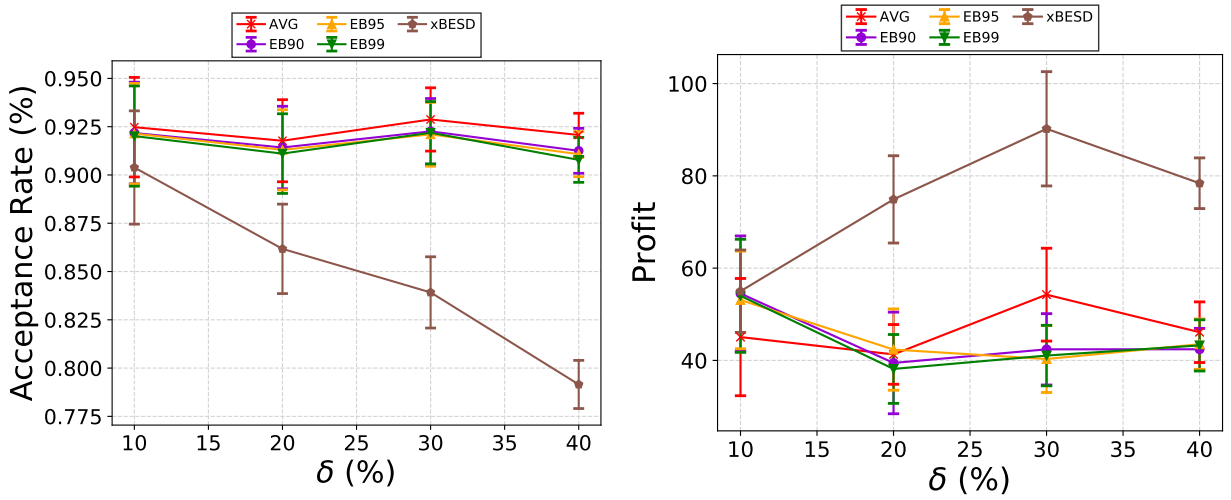
(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.12: Effect of maximum capacity deviation from average in path deviation model.

5.2.2 Sensitivity Analysis

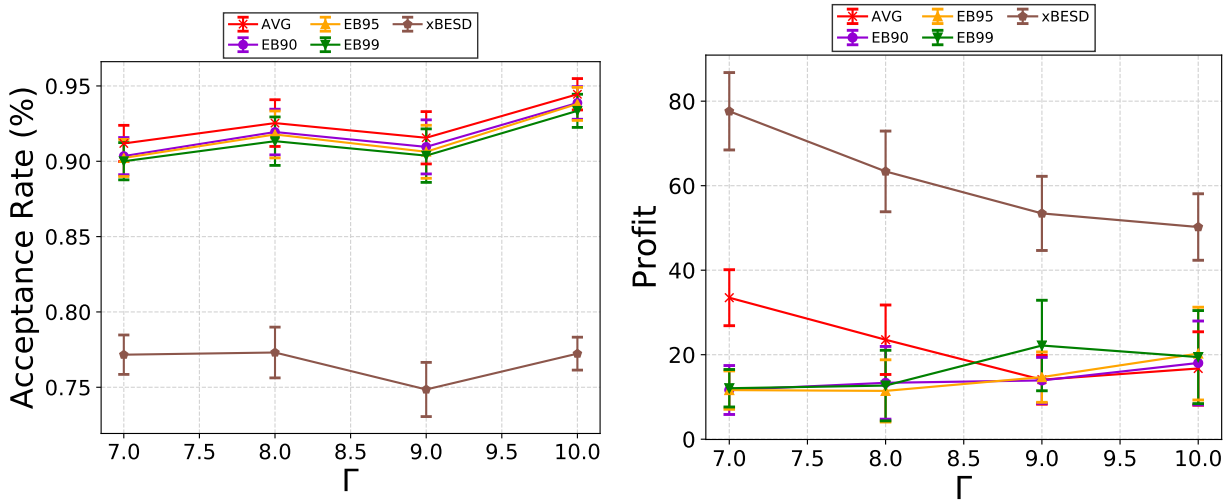
In this section, we evaluate the performance of our algorithm under unpredictable changes to the estimations. In order to conduct this experiment, we make our algorithm unaware of the changes to the values of Γ, Υ, and δ.



(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.13: Effect of maximum capacity deviation from average in time deviation model.



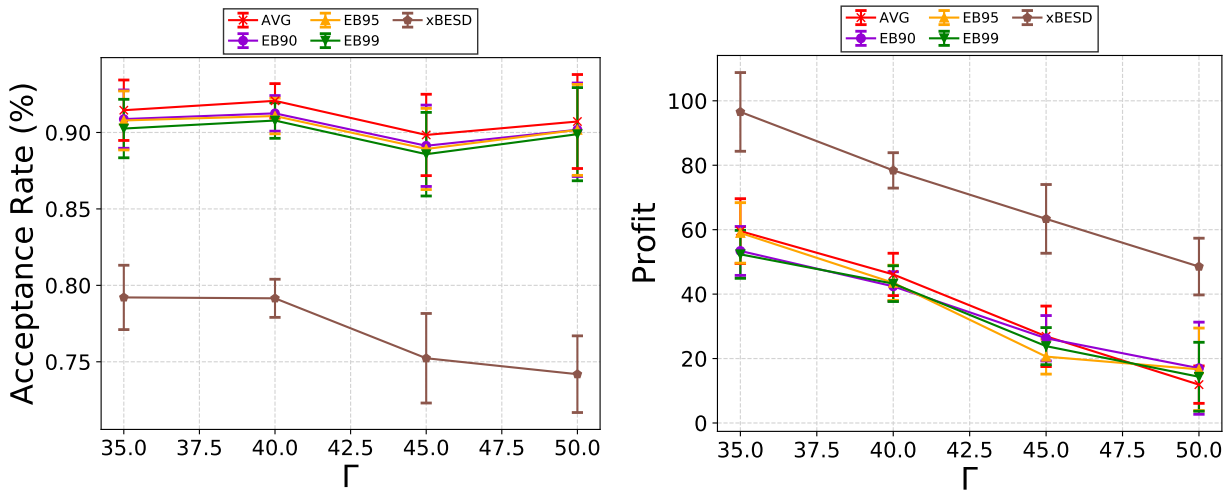
(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.14: Effect of number of fluctuations per time window.

Effect of Capacity Fluctuations

In this scenario, we demonstrate that small unpredictable changes to the maximum capacity fluctuations do not severely affect our algorithm' performance. Table 5.1 shows that 30% change to the estimations causes about the same percentage of performance drop.



(a) Effect on acceptance rate.

(b) Effect on profit.

Figure 5.15: Effect of number of fluctuations per path per time window.

δ	**Acceptance Rate**	**Profit**
30%	0.79	82.95
32%	0.83	82.45
34%	0.81	82.9
36%	0.79	71.85
38%	0.77	65.68
40%	0.83	66.63

Table 5.1: Effect of maximum capacity deviation from average

Effect of Number of Fluctuations Per Timeslot

In Table 5.2 we notice that our algorithm has a mild reaction to unpredictable changes at first, then its performance drops significantly. However, we should note that we do not expect more than 30% unpredictable change in the estimations, as was mentioned in the previous chapters.

Effect of Number of Fluctuations Per Path Per Transmission Period

Similar to the previous experiment, in Table 5.3, we observe that our algorithm has a negligible loss up to the 30% threshold. Afterwards, there is a significant performance loss.

Γ	**Acceptance Rate**	**Profit**
7	0.79	81.7
8	0.84	53.7
9	0.82	45.8
10	0.79	20.5

Table 5.2: Effect of number of fluctuations per timeslot.

Υ	**Acceptance Rate**	**Profit**
20	0.75	114.95
21	0.73	116.6
22	0.72	115.2
23	0.73	108.23
24	0.75	106.63
25	0.76	106.13
26	0.77	104.56
27	0.74	97.15
28	0.75	96.36
29	0.74	93.29
30	0.72	89.64

Table 5.3: Effect of number of fluctuations per path per transmission period.

5.2.3 Mininet Experiments

We use Mininet [3] to evaluate our algorithm's performance in an emulated realistic network environment. The available bandwidth of each path changes according to the same fluctuation models as the simulations, namely Path Deviation Model and Time Deviation Model.

Setup

To carry out Mininet experiments, we set up an environment with the following specifications:

- **Trials:** We conducted a trial of 150 minutes (*i.e.*, 50 timeslots of 3 minutes) for each of the algorithms. During each trial, we generated traffic in such a way to account for bulk transfers as well as to emulate the capacity fluctuations.

- **Topology:** Our emulated topology consists of 12 switches and 2 hosts. Ten switches that emulate the Internet paths are connected to both of the hosts through two ingress/egress switches. This topology is shown in Fig. 5.16. We set the capacity of the links using the `tc` command so as to conform to the parameters we used for the paths in the simulations. In order to create capacity fluctuations, we generated background traffic and sent it over the network. One of the hosts acted as the sender, and the other acted as the receiver of the bulk transfers.

- **Controller:** We built the developer edition of the `ONOS` SDN controller from the source and used it as our controller, which is shown in Fig. 5.16. Our solver passes the optimization problem's solution as the set of available paths per transfer per timeslot to the controller. The controller then installs the pertinent flows at the beginning of every timeslot according to that information.

- **Multipath Routing:** We implemented a software module on top of the ONOS controller using its `Java API v1.13.1` and the `TopologyService` interface in order to enable the controller to assign multiple paths to each flow.

- **Traffic Monitoring:** We implemented the traffic monitoring module on top of the controller to collect port utilization statistics using `PortStatistics` interface during each experiment.

- **Traffic Generator:** We implemented a custom traffic generator consisting of a client and a server. The client generates bulk transfers using UDP packets. For each specific transfer request, we used a unique destination port number.

- **Environment:** We conducted all experiments on an Ubuntu 20.0 LTS VM on Amazon AWS with four vCPUs and 8 GBs of RAM.

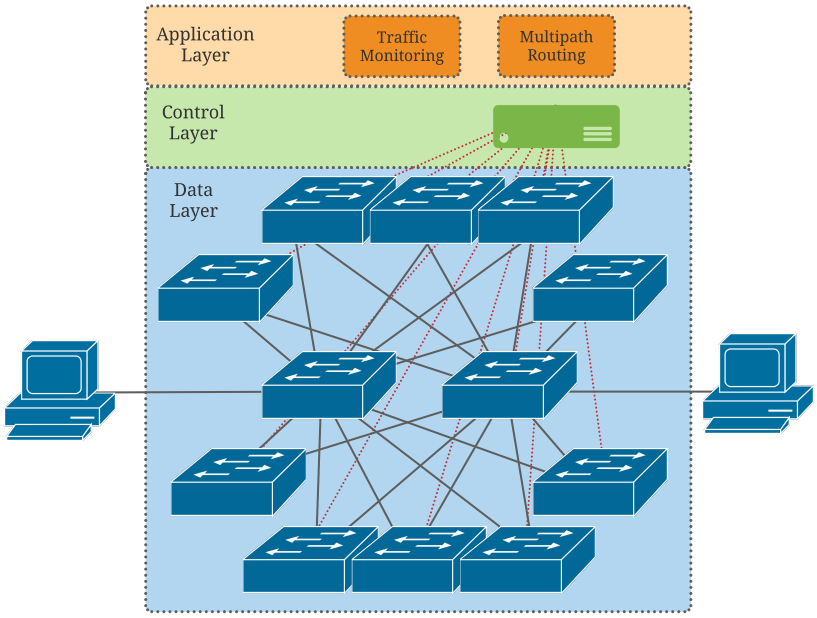


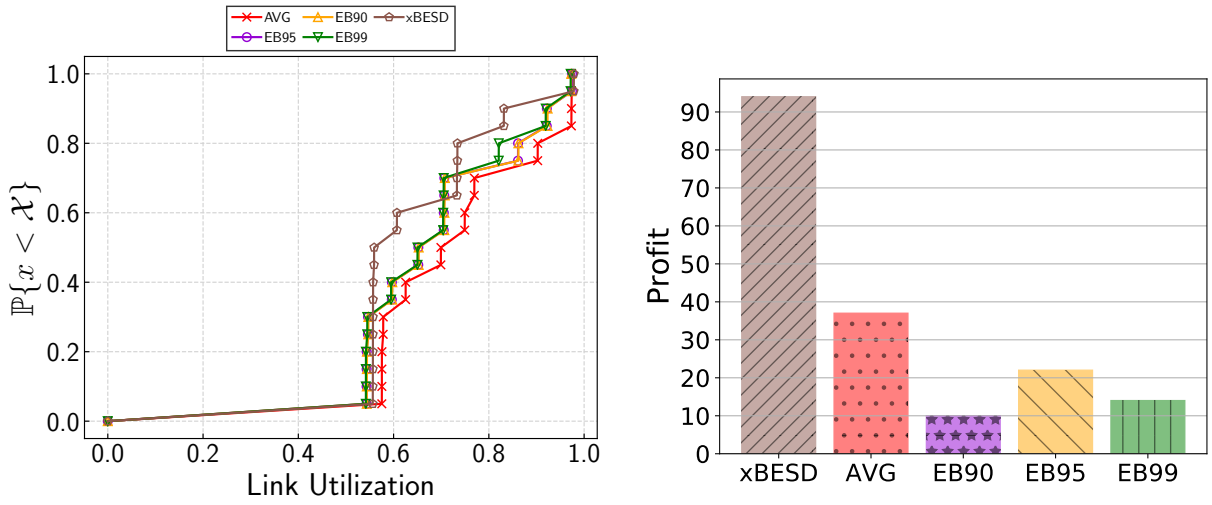
Figure 5.16: Topology of the experimental network.

Path Deviation Model

In this experiment, in order to create 70% fluctuation intensity (*i.e.*, $\Gamma = 7$), we introduce a 40% fluctuation from the estimated average for 7 out of 10 paths, while the other 3 stay within their estimated average. We chose the other parameters to conform with the simulations and studied the following metrics:

- **Link Utilization:** The amount of traffic traversing a link divided by its total link capacity.
- **Profit:** The total profit gained from successfully transmitting bulk transfer requests before their associated deadlines.

Figs. 5.17a and 5.17b demonstrate the achieved results in this experiment. Specifically, Fig. 5.17a shows that our algorithm achieves approximately 12% lower utilization in the majority of timeslots, which means significantly less bandwidth consumption. However, baseline algorithms have roughly the same utilization because they admit more or less the



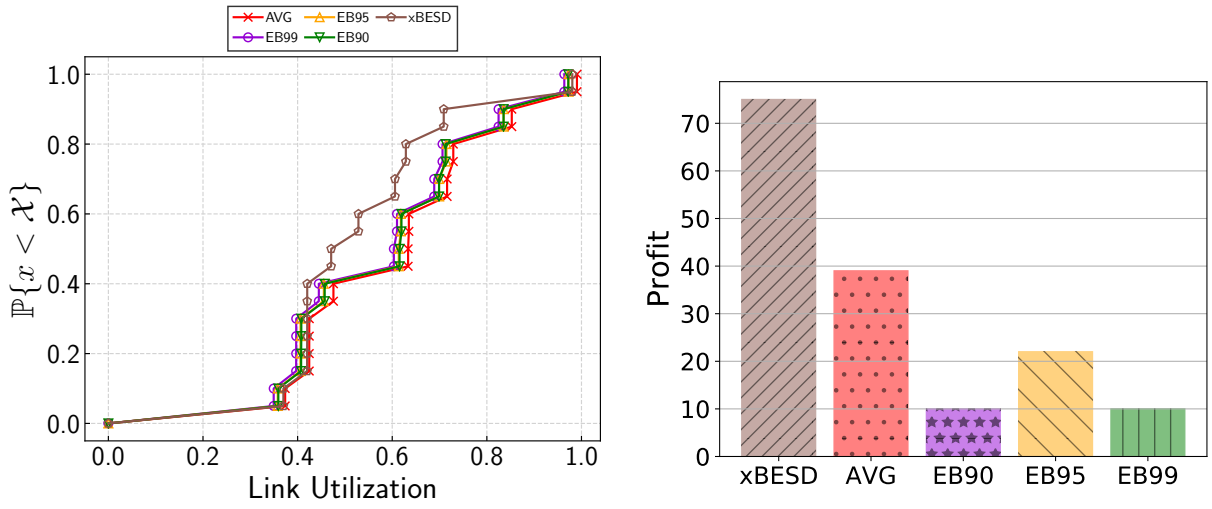
(a) CDF of link utilization.

(b) profit of different algorithms.

Figure 5.17: Path deviation experimental results.

same requests, which is also compatible with the results obtained from the simulations. This stems from the fact that our robust formulation allows our algorithm to admit requests according to the characteristics of the fluctuations, therefore, it is less likely to admit requests that cannot be successfully transmitted before their deadlines. On the other hand, not having a proper formulation of the fluctuations puts baseline algorithms in a disadvantages, leading to admitting too many requests that eventually cannot successfully finish, hence the higher utilization. Furthermore, Fig. 5.17b shows that baseline algorithms, due to over-allocation and under-allocation of the paths, perform inferior to our algorithm. The AVG algorithm does not consider fluctuations and over-allocate the paths in every fluctuation timeslot. On the other hand, the EB algorithm takes a conservative approach and considers a capacity between the maximum fluctuation and the average, causing EB to over-allocate the paths in a fluctuation timeslot and under-allocate them in other times, which puts EB at a significant disadvantage compared to AVG and xBESD, hence the lower achieved profit.

It is worth mentioning that having a lower bandwidth consumption gives xBESD a notable financial advantage. In this regard, burstable billing or usage-based billing is a common billing scheme for Internet paths [65, 22]. In these schemes, the user is charged a bill according



(a) CDF of link utilization.

(b) profit of different algorithms.

Figure 5.18: Time deviation experimental results.

to their bandwidth consumption in a billing cycle. In state-of-the-art algorithms that utilize leased lines in an MPLS network, maximizing path utilization is of utmost importance since a significant portion of the costs are dedicated to these lines; therefore, they are ought to be used most efficiently. In our work, on the other hand, we utilize Internet best effort paths that are most likely billed using a usage-based scheme. Consequently, lower path utilization leads to paying less for the provided service to the ISP and lowering the costs. In summary, our algorithm has superior performance in terms of gained profit and bandwidth consumption compared to baseline algorithms.

Time Deviation Model

In this experiment, similarly, we study bandwidth utilization and achieved profit. To this end, we create capacity fluctuations for every path in 70% of the total timeslots in a transmission period. In other words, each path fluctuates maximally in 35 timeslots randomly chosen among all timeslots (*i.e.*, $\Upsilon = 35$). Figs. 5.18a and 5.18b demonstrate the advantages of our robust formulation of capacity fluctuations compared to algorithms that either do not take them into consideration or utilize a conservative approach towards them. Baseline algorithms

can successfully transmit a transfer provided that no paths on which the transfer's traffic traverses suffer from capacity fluctuations during the total course of transmission. Since this occurrence is rare, baseline algorithms cannot transmit too many requests successfully. On the other hand, xBESD achieves a robust formulation of capacity fluctuations, giving it the advantage of a higher probability of predicting the fluctuations, thus, successfully transmitting more requests.

Chapter 6

Conclusion

6.1 Thesis Summary

In this thesis, we provided a solution for scheduling bulk transfer requests over the Internet best-effort paths. We argued that we could provide a cheaper alternative to dedicated leased lines by utilizing the software defined networks and best-effort paths. We also provided a solution to account for the capacity fluctuations on the best-effort paths due to cross traffic, traffic engineering, and maintenance by proposing a robust formulation of the scheduling problem, which requires partial information with respect to the behaviour of the path capacities. Using statistical analysis, we demonstrated that Internet paths' capacity is predictable in short periods and does not suffer significant fluctuations. Furthermore, by utilizing the machine learning technique and LSTM neural networks, we showed that it is feasible to predict an estimation range with acceptable error for capacities in more extended periods, hence the possibility of realizing the robust formulation.

Moreover, considering the computational complexity of the deduced optimization problem, we proposed an approximate algorithm using the rounding technique, which has comparable performance as well as a significant computational advantage. We evaluated our proposed proactive algorithm's performance against two baseline methods using two sets of

experiments, namely simulations and Mininet experiments. In the former, we first showcased the performance advantage of our algorithm in a general setup. Then, we provided several micro-benchmarks in order to provide a thorough analysis of our algorithm under the effects of different environments and algorithm parameters. In the Mininet experiments, we studied our proposed algorithm's performance in an emulated real-world scenario, which validated our simulation results. In summary, we showed that we could estimate the behaviour of best-effort SD-WAN capacities within an acceptable error margin and properly formulating the problem of bulk transfer scheduling over best-effort SD-WANs, and utilizing their estimated behaviour we can obtain a cheaper alternative than the leased dedicated WANs.

6.2 Future Research Directions

This section highlights interesting future research directions of resource scheduling in SD-WANs.

- Reconfigurable Networks: The myriad of the state-of-the-art works in inter-datacenter networks, though sophisticated, are demand-oblivious with regards to the network's topology. In other words, the underlying network's topology is fixed regardless of the user application's demand, and it rather suggests to the user a set of available resources from a fixed pool. However, a new paradigm has brought about a new opportunity for the network providers to reconfigure the topology at run-time based on the user's demand [6]. Combined with SDN, reconfigurable optical networks provide many opportunities for application-aware bulk transfer requests.

- Passive Optical Networks: Passive optical networks (PON) is a new paradigm of optical networks in which the upstream works in a Time Division Multiplexing fashion and the downstream is broadcast, which allows some units to put their sender or receiver to sleep at specific times in order to conserve energy. By utilizing SDN and this type of networks for inter-DC transfers we can save a significant amount of energy. To this

end, we can optimize the sleep schedules in such way to only meet the request deadlines while maximizing energy conservation.

Bibliography

[1] Gurobi, the fastest solver. Available: `https://www.gurobi.com`. Accessed: December 2020. [Online].

[2] Iperf, the ultimate speed test tool for tcp, udp and sctp. Available: `https://iperf.fr`. Accessed: December 2020. [Online].

[3] Mininet, an instant virtual network on your laptop (or other pc). Available: `http://mininet.org`. Accessed: December 2020. [Online].

[4] Abdelnaser Adas. Traffic models in broadband networks. *IEEE Communications Magazine*, 35(7):82–89, 1997.

[5] Petri Aukia, Murali Kodialam, Pramod VN Koppol, TV Lakshman, Helena Sarin, and Bernhard Suter. Rates: A server for mpls traffic engineering. *IEEE Network*, 14(2):34–41, 2000.

[6] Chen Avin and Stefan Schmid. Toward demand-aware networking: A theory for self-adjusting networks. *Proc. ACM SIGCOMM*, 48(5):31–40, 2019.

[7] Abdul Basit, Saad Qaisar, Mudassar Ali, Muhammad Naeem, Marc Bruyere, and Joel JPC Rodrigues. Interconnecting networks with optimized service provisioning. *Telecommunication Systems*, 73(2):223–239, 2020.

[8] Paola Bermolen and Dario Rossi. Support vector regression for link load prediction. *Computer Networks*, 53(2):191–201, 2009.

[9] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[10] Dimitris Bertsimas and Melvyn Sim. The Price of Robustness. *Operations Research*, 52(1):35–53, 2004.

[11] Matthew Caesar, Donald Caldwell, Nick Feamster, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe. Design and implementation of a routing control platform. In *Proc. USENIX NSDI*, pages 15–28, 2005.

[12] Martin Casado, Michael J Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane: Taking control of the enterprise. volume 37, pages 1–12, 2007.

[13] Samira Chabaa, Abdelouhab Zeroual, Jilali Antari, et al. Identification and prediction of internet traffic using artificial neural networks. *Journal of Intelligent Learning Systems and Applications*, 2(03):147, 2010.

[14] Federico Chiariotti, Stefano D'Aronco, Laura Toni, and Pascal Frossard. Online learning adaptation strategy for dash clients. In *In Proc. ACM MMSys*, pages 1–12, 2016.

[15] TS Choi, SH Yoon, HS Chung, CH Kim, JS Park, BJ Lee, and TS Jeong. Wise: traffic engineering server for a large-scale mpls-based ip network. In *Proc. IEEE/IFIP NOMS*, pages 251–264, 2002.

[16] David Chou, Tianyin Xu, Kaushik Veeraraghavan, Andrew Newell, Sonia Margulis, Lin Xiao, Pol Mauri Ruiz, Justin Meza, Kiryong Ha, Shruti Padmanabha, et al. Taiji: managing global user traffic for large-scale internet services at the edge. In *Proc. ACM SOSP*, pages 430–446, 2019.

[17] Massimo Corradi, Rosario Giuseppe Garroppo, Stefano Giordano, and Michele Pagano. Analysis of f-arima processes in the modelling of broadband traffic. In *Proc. IEEE ICC*, volume 3, pages 964–968, 2001.

[18] Paulo Cortez, Miguel Rio, Miguel Rocha, and Pedro Sousa. Internet traffic forecasting using neural networks. In *Proc. IEEE IJCNN*, pages 2635–2642, 2006.

[19] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[20] Emilie Danna, Avinatan Hassidim, Haim Kaplan, Alok Kumar, Yishay Mansour, Danny Raz, and Michal Segalov. Upward max-min fairness. *Journal of the ACM (JACM)*, 64(1):1–24, 2017.

[21] Emilie Danna, Subhasree Mandal, and Arjun Singh. A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering. In *Proc. IEEE INFOCOM*, pages 846–854, 2012.

[22] Xenofontas Dimitropoulos, Paul Hurley, Andreas Kind, and Marc Ph Stoecklin. On the 95-percentile billing method. In *Proc. Springer PAM*, pages 207–216, 2009.

[23] Mahdi Dolati, Majid Ghaderi, and Ahmad Khonsari. Proactive inter-datacenter multicast with realtime and bulk transfers. In *Proc. IEEE/ACM IWQoS*, pages 1–10, 2019.

[24] Svante Ekelin, Martin Nilsson, Erik Hartikainen, Andreas Johnsson, J-E Mangs, Bob Melander, and Mats Bjorkman. Real-time measurement of end-to-end available bandwidth using kalman filtering. In *Proc. IEEE/IFIP NOMS*, pages 73–84, 2006.

[25] Anwar Elwalid, Cheng Jin, Steven Low, and Indra Widjaja. Mate: Mpls adaptive traffic engineering. In *Proc. IEEE INFOCOM*, volume 3, pages 1300–1309, 2001.

[26] Alaknantha Eswaradass, X-H Sun, and Ming Wu. A neural network based predictive mechanism for available bandwidth. In *Proc. IEEE IPDPS*, pages 10–pp, 2005.

[27] Alaknantha Eswaradass, Xian-He Sun, and Ming Wu. Network bandwidth predictor (nbp): A system for online network performance forecasting. In *Proc. IEEE CCGRID*, volume 1, pages 4–pp, 2006.

[28] Ilhem Fajjari, Nadjib Aitsaadi, and Djamel Eddine Kouicem. A novel sdn scheme for qos path allocation in wide area networks. In *Proc. IEEE GLOBECOM*, pages 1–7, 2017.

[29] Yuan Feng, Baochun Li, and Bo Li. Jetway: Minimizing costs on inter-datacenter video traffic. In *Proc. ACM MM*, pages 259–268, 2012.

[30] Yuan Feng, Baochun Li, and Bo Li. Postcard: Minimizing Costs on Inter-Datacenter Traffic with Store-and-Forward. In *Proc. IEEE ICDCSW*, pages 43–50, 2012.

[31] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. Traffic engineering with traditional ip routing protocols. *IEEE communications Magazine*, 40(10):118–124, 2002.

[32] Qian Gao, Jiang Liu, Ningjie Gao, and Tao Huang. A dual decomposition method for hierarchical traffic control in inter-dc wans. In *Proc. IEEE/CIC*, pages 310–315, 2019.

[33] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. In *Proc. IET ICANN*, pages 850–855, 1999.

[34] Arthur S Goldberger. Econometric computing by hand. *Journal of Economic and Social Measurement*, 29(1-3):115–117, 2004.

[35] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[36] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *Proc. ACM SIGCOMM*, pages 15–26, 2013.

[37] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proc. PNAS*, volume 79, pages 2554–2558, 1982.

[38] Arshia Hosseini, Mahdi Dolati, and Majid Ghaderi. Bulk transfer scheduling with deadline in best-effort sd-wans. In *Proc. IFIP/IEEE IM*, (Forthcoming 2021).

[39] Van Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM*, volume 18, pages 314–329, 1988.

[40] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *Proc. Springer PAM*, pages 14–25, 2002.

[41] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, et al. B4: Experience with a globally-deployed software defined WAN. In *Proc. ACM SIGCOMM*, page 3–14, 2013.

[42] Virajith Jalaparti, Ivan Bliznets, Srikanth Kandula, Brendan Lucier, and Ishai Menache. Dynamic Pricing and Traffic Engineering for Timely Inter-Datacenter Transfers. In *Proc. ACM SIGCOMM*, pages 73–86, 2016.

[43] Andreas Johnsson, Bob Melander, Mats Björkman, and M Bjorkman. Diettopp: A first implementation and evaluation of a simplified bandwidth measurement method. In *Proc. IEEE SNCNW*, volume 5. Citeseer, 2004.

[44] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*, volume 35, pages 253–264, 2005.

[45] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana Raj Babbula. Calendaring for wide area networks. In *Proc. ACM SIGCOMM*, pages 515–526, 2014.

[46] Frank P. Kelly. Effective bandwidths at multi-class queues. *Queueing systems*, 9(1):5–15, 1991.

[47] Sukhpreet Kaur Khangura, Markus Fidler, and Bodo Rosenhahn. Neural Networks for Measurement-based Bandwidth Estimation. In *Proc. IFIP Networking*, pages 1–9, 2018.

[48] Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez. Inter-datacenter bulk transfers with netstitcher. pages 74–85, 2011.

[49] Will E Leland, Murad S Taqqu, Walter Willinger, and Daniel V Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[50] Wenxin Li, Keqiu Li, Deke Guo, Geyong Min, Heng Qi, and Jianhui Zhang. Cost-minimizing bandwidth guarantee for inter-datacenter traffic. *IEEE Transactions on Cloud Computing*, 7(2):483–494, 2016.

[51] Wenxin Li, Xiaobo Zhou, Keqiu Li, Heng Qi, and Deke Guo. TrafficShaper: Shaping Inter-Datacenter Traffic to Reduce the Transmission Cost. *IEEE/ACM Transactions on Networking*, 26(3):1193–1206, 2018.

[52] Long Luo, Hongfang Yu, Zilong Ye, and Xiaojiang Du. Online Deadline-Aware Bulk Transfer Over Inter-Datacenter WANs. In *Proc. IEEE INFOCOM*, pages 630–638, 2018.

[53] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proc. ACM SIGCOMM*, pages 197–210, 2017.

[54] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. In *Proc. ACM SIGCOMM*, volume 38, pages 69–74, 2008.

[55] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifa Han, Feng Li, and Jin Li. Realtime mobile bandwidth prediction using lstm neural network. In *Proc. Springer PAM*, pages 34–47, 2019.

[56] Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu. A machine learning approach to tcp throughput prediction. In *Proc. ACM SIGMETRICS*, volume 35, pages 97–108, 2007.

[57] Tom Mitchell. Introduction to machine learning. *Machine Learning*, 7:2–5, 1997.

[58] Mehrdad Moradi, Ying Zhang, Z Morley Mao, and Ravi Manghirmalani. Dragon: Scalable, flexible, and efficient traffic engineering in software defined isp networks. *IEEE Journal on Selected Areas in Communications*, 36(12):2744–2756, 2018.

[59] Thyaga Nandagopal and Krishna PN Puttaswamy. Lowering inter-datacenter bandwidth costs via bulk data scheduling. In *Proc. IEEE/ACM CCGRID*, pages 244–251, 2012.

[60] Mohammad Noormohammadpour, Cauligi S Raghavendra, Srikanth Kandula, and Sriram Rao. Quickcast: Fast and efficient inter-datacenter transfers using forwarding tree cohorts. In *Proc. IEEE INFOCOM*, pages 225–233, 2018.

[61] Mohammad Noormohammadpour, Cauligi S Raghavendra, Sriram Rao, and Srikanth Kandula. Dccast: Efficient point to multipoint transfers across datacenters. In *Proc. USENIX*, 2017.

[62] Vinay Joseph Ribeiro, Rudolf H Riedi, Richard G Baraniuk, Jiri Navratil, and Les Cottrell. Pathchirp: Efficient available bandwidth estimation for network paths. In *Proc. Springer PAM*, 2003.

[63] Aimin Sang and San-qi Li. A predictability analysis of network traffic. *Computer Networks*, 39(4):329–345, 2002.

[64] Caterina Scoglio, Tricha Anjali, Jaudelice Cavalcante de Oliveira, Ian F Akyildiz, and G UhI. Team: A traffic engineering automated manager for diffserv-based mpls networks. *IEEE Communications Magazine*, 42(10):134–145, 2004.

[65] Scott Shenker, David Clark, Deborah Estrin, and Shai Herzog. Pricing in computer networks: Reshaping the research agenda. *ACM SIGCOMM*, 26(2):19–43, 1996.

[66] Yantai Shu, Zhigang Jin, Lianfang Zhang, Lei Wang, and Oliver WW Yang. Traffic prediction using farima models. In *Proc. IEEE ICC*, volume 2, pages 891–895, 1999.

[67] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *Proc. ACM SIGCOMM IMC*, pages 39–44, 2003.

[68] Yi Sun, Xiaoqi Yin, Junchen Jiang, et al. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proc. ACM SIGCOMM*, pages 272–285, 2016.

[69] Jessie Hui Wang, Jilong Wang, Changqing An, and Qianli Zhang. A survey on resource scheduling for data transfers in inter-datacenter wans. *Computer Networks*, 161:115–137, 2019.

[70] Yi Wang, Jiaqi Zheng, Lijuan Tan, and Chen Tian. Joint optimization on bandwidth allocation and route selection in qoe-aware traffic engineering. *IEEE Access*, 7:3314–3319, 2018.

[71] Yiwen Wang, Sen Su, Alex X. Liu, and Zhongbao Zhang. Multiple bulk data transfers scheduling among datacenters. *Computer Networks*, 68:123–137, 2014.

[72] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Learning in situ: a randomized experiment in video streaming. In *Proc. USENIX NSDI*, pages 495–511, 2020.

[73] Z. Yang, Y. Cui, X. Wang, Y. Liu, M. Li, S. Xiao, and C. Li. Cost-efficient scheduling of bulk transfers in inter-datacenter wans. *IEEE/ACM Transactions on Networking*, 27(5):1973–1986, 2019.

[74] Zhenjie Yang, Yong Cui, Xin Wang, Yadong Liu, Minming Li, Shihan Xiao, and Chuming Li. Cost-Efficient Scheduling of Bulk Transfers in Inter-Datacenter WANs. *IEEE/ACM Transactions on Networking*, 27(5):1973–1986, 2019.

[75] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proc. ACM SIGCOMM*, pages 325–338, 2015.

[76] Gaoxiong Zeng, Wei Bai, Ge Chen, Kai Chen, Dongsu Han, Yibo Zhu, and Lei Cui. Congestion control for cross-datacenter networks. In *Proc. IEEE ICNP*, pages 1–12, 2019.

[77] Hong Zhang, Kai Chen, Wei Bai, Dongsu Han, et al. Guaranteeing Deadlines for Inter-Data Center Transfers. *IEEE/ACM Transactions on Networking*, 25:579–595, 2017.

[78] Qingyi Zhang, Xingwei Wang, Jianhui Lv, and Min Huang. Intelligent content-aware traffic engineering for sdn: An ai-driven approach. *IEEE Network*, 34(3):186–193, 2020.

[79] Yuchao Zhang, Junchen Jiang, Ke Xu, Xiaohui Nie, Martin J Reed, Haiyang Wang, Guang Yao, Miao Zhang, and Kai Chen. Bds: a centralized near-optimal overlay network for inter-datacenter data replication. In *Proc. ACM EuroSys*, pages 1–14, 2018.

[80] Yan Zhu, Guanghua Zhang, and Jing Qiu. Network traffic prediction based on particle swarm bp neural network. *J. Networks*, 8(11):2685–2691, 2013.