Machine Learning-based Available Bandwidth Estimation

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieurin

genehmigte

Dissertation

von

M.Sc. Sukhpreet Kaur Khangura
geboren am 22. Oktober 1983 in Bhawanigarh Sangrur

2019

Referent	:	Prof. Dr.-Ing. Markus Fidler
Korreferent	:	Prof. Dr.-Ing. Bodo Rosenhahn

Tag der Promotion : 12.12.2019

ABSTRACT

Today's Internet Protocol (IP), the Internet's network-layer protocol, provides a best-effort service to all users without any guaranteed bandwidth. However, for certain applications that have stringent network performance requirements in terms of bandwidth, it is significantly important to provide Quality of Service (QoS) guarantees in IP networks. The end-to-end available bandwidth of a network path, i.e., the residual capacity that is left over by other traffic, is determined by its tight link, that is the link that has the minimal available bandwidth. The tight link may differ from the bottleneck link, i.e., the link with the minimal capacity.

Passive and active measurements are the two fundamental approaches used to estimate the available bandwidth in IP networks. Unlike passive measurement tools that are based on the non-intrusive monitoring of traffic, active tools are based on the concept of self-induced congestion. The dispersion, which arises when packets traverse a network, carries information that can reveal relevant network characteristics. Using a fluid-flow probe gap model of a tight link with First-in, First-out (FIFO) multiplexing, accepted probing tools measure the packet dispersion to estimate the available bandwidth. Difficulties arise, however, if the dispersion is distorted compared to the model, e.g., by non-fluid traffic, multiple tight links, clustering of packets due to interrupt coalescing and inaccurate time-stamping in general. It is recognized that modeling these effects is cumbersome if not intractable.

To alleviate the variability of noise-afflicted packet gaps, the state-of-the-art bandwidth estimation techniques use post-processing of the measurement results, e.g., averaging over several packet pairs or packet trains, linear regression, or a Kalman filter. These techniques, however, do not overcome the basic assumptions of the deterministic fluid model. While packet trains and statistical post-processing help to reduce the variability of available bandwidth estimates, these cannot resolve systematic deviations such as the underestimation bias in case of random cross traffic and multiple tight links. The limitations of the state-of-the-art methods motivate us to explore the use of machine learning in end-to-end active and passive available bandwidth estimation.

We investigate how to benefit from machine learning while using standard packet train probes for active available bandwidth estimation. To reduce the amount of required training data, we propose a regression-based scale-invariant method that is applicable without prior calibration to networks of

arbitrary capacity. To reduce the amount of probe traffic further, we implement a neural network that acts as a recommender and can effectively select the probe rates that reduce the estimation error most quickly. We also evaluate our method with other regression-based supervised machine learning techniques.

Furthermore, we propose two different multi-class classification-based methods for available bandwidth estimation. The first method employs reinforcement learning that learns through the network path's observations without having a training phase. We formulate the available bandwidth estimation as a single-state Markov Decision Process (MDP) multi-armed bandit problem and implement the ε-greedy algorithm to find the available bandwidth, where ε is a parameter that controls the *exploration vs. exploitation* trade-off.

We propose another supervised learning-based classification method to obtain reliable available bandwidth estimates with a reduced amount of network overhead in networks, where available bandwidth changes very frequently. In such networks, reinforcement learning-based method may take longer to converge as it has no training phase and learns in an online manner. We also evaluate our method with different classification-based supervised machine learning techniques. Furthermore, considering the correlated changes in a network's traffic through time, we apply filtering techniques on the estimation results in order to track the available bandwidth changes.

Active probing techniques provide flexibility in designing the input structure. In contrast, the vast majority of Internet traffic is Transmission Control Protocol (TCP) flows that exhibit a rather chaotic traffic pattern. We investigate how the theory of active probing can be used to extract relevant information from passive TCP measurements. We extend our method to perform the estimation using only sender-side measurements of TCP data and acknowledgment packets. However, non-fluid cross traffic, multiple tight links, and packet loss in the reverse path may alter the spacing of acknowledgments and hence increase the measurement noise. To obtain reliable available bandwidth estimates from noise-afflicted acknowledgment gaps we propose a neural network-based method.

We conduct a comprehensive measurement study in a controlled network testbed at Leibniz University Hannover. We evaluate our proposed methods under a variety of notoriously difficult network conditions that have not been included in the training such as randomly generated networks with multiple tight links, heavy cross traffic burstiness, delays, and packet loss. Our testing results reveal that our proposed machine learning-based techniques are able to identify the available bandwidth with high precision from active and passive measurements. Furthermore, our reinforcement learning-based method with-

out any training phase shows accurate and fast convergence to available bandwidth estimates.

Keywords: Available Bandwidth Estimation, Active and Passive Measurements, Machine Learning, Support Vector Machines, k-Nearest Neighbors, Bagging, AdaBoost, Neural Network, Reinforcement Learning.

ZUSAMMENFASSUNG

Das heutige *Internet Protocol* (IP), das Netzwerkprotokoll des Internets, bietet allen Benutzern einen sogenannten *best effort*-Service ohne garantierte Bandbreite. Für bestimmte Anwendungen mit hohen Anforderungen an die Netzwerkperformanz in Bezug auf Bandbreite ist es jedoch von entscheidender Bedeutung, in IP-basierten Netzwerken Garantien für Dienstgüte (Quality of Service) bereitzustellen. Die verfügbare Ende-zu-Ende-Bandbreite eines Netzwerkpfads, d. h. die Restkapazität, die durch anderen Datenverkehr nicht genutzt wird, wird durch seinen sogenannten *tight link* determiniert, d. h. die Verbindung mit der minimal verfügbaren Bandbreite. Der *tight link* kann sich allerdinger von dem *bottleneck link* unterscheiden, d. h. der Verbindung mit der minimalen Kapazität.

Passive und aktive Messungen sind die beiden grundlegenden Ansätze zur Schätzung der verfügbaren Bandbreite in IP-Netzwerken. Im Gegensatz zu passiven Messwerkzeugen, die auf der störungsfreien Überwachung des Verkehrs basieren, beruhen aktive Werkzeuge auf dem Konzept der selbst-induzierten Überlastung. Die Streuung, die entsteht, wenn Pakete ein Netzwerk durchlauten, enthält Informationen, die relevante Netzwerkmerkmale aufdecken können. Unter Verwendung eines Fluid-Flow-Probe-Gap-Modells eines *tight link* mit FIFO-Multiplexing (First-in, First-out) messen Testtools die Paketdispersion und schätzen damit die verfügbare Bandbreite ab. Schwierigkeiten entstehen jedoch, wenn die Dispersion im Vergleich zum Modell verzerrt ist, z. B. durch stockenden Datenverkehr, mehrere *tight links*, Clusterbildung von Paketen aufgrund von Interrupt-Koaleszenz und, allgemein, ungenauer Zeitstempelung. Es ist bekannt, dass eine Modellierung dieser Effekte schwierig oder gar unmöglich ist.

Um die Variabilität von Rausch-behafteten Paketlücken zu verringern, werden die Messergebnisse durch Bandbreitenschätzungsmethoden nach dem Stand der Technik zusätzlich, z. B. durch Mittelwertbildung über mehrere Paketpaare oder Paketzüge, lineare Regression oder Kalman-Filter nachbearbeitet. Diese Techniken überwinden jedoch nicht die Grundannahmen des deterministischen Fluid-Modells. Auch wenn die Verwendung von Paketzügen und die statistische Nachbearbeitung der Messergebnisse dabei helfen, die Variabilität in den Schätzungen der verfügbaren Bandbreite zu verringern, können diese keine systematischen Abweichungen, wie z. B. zu niedrige Schätzungen bei zufälligem Querverkehr und mehreren *tight links* beseitigen. Die Einschränkungen

moderner Methoden motivieren uns, den Einsatz von maschinellem Lernen bei der aktiven und passiven Schätzung der verfügbaren Ende-zu-Ende Bandbreite zu explorieren.

Wir untersuchen, wie man von maschinellem Lernen profitieren kann, wenn man normale Paketzug-Messungen für die aktive Schätzung der verfügbaren Bandbreite verwendet. Um die Menge der erforderlichen Trainingsdaten zu reduzieren, schlagen wir eine auf Regression basierende skalierungsinvariante Methode vor, die ohne vorherige Kalibrierung auf Netzwerke mit beliebiger Kapazität anwendbar ist. Um den für Messungen nötigen Datenverkehr weiter zu reduzieren, implementieren wir ein neuronales Netzwerk, das als Empfehlungssystem für Testdatenraten die schnelle Reduzierung der Schätzfehler erzielen. Wir vergleichen unsere Methode auch mit anderen, auf Regression basierenden, überwachten Techniken des maschinellen Lernens.

Darüber hinaus schlagen wir zwei verschiedene klassifikationsbasierte Mehrklassenmethoden für die Schätzung der verfügbaren Bandbreite vor. Bei der ersten Methode wird ein bestärkendes Lernen (reinforcement learning) verwendet, das durch die Beobachtung des Netzwerkpfads lernt, ohne eine Trainingsphase zu haben. Wir formulieren die Schätzung der verfügbaren Bandbreite als ein Single-State Markov-Entscheidungsprozess (MDP) Mehrarmige Banditen-Problems und implementieren den ε-greedy-Algorithmus, um die verfügbare Bandbreite zu ermitteln, wobei ε ein Parameter ist, der fortwährend die Balance zwischen Erkundung und Verwertung hält.

Wir schlagen eine weitere lernbasierte Klassifizierungsmethode vor, um zuverlässige Schätzungen der verfügbaren Bandbreite mit einem reduzierten Overhead in Netzwerken, in denen sich die verfügbare Bandbreite sehr häufig ändert, zu erhalten. In solchen Netzwerken können Methoden, die auf verstärktem Lernen basieren, mehr Zeit zum Konvergieren benötigen, da sie keine Trainingsphase haben und im Betrieb fortlaufend lernen. Wir vergleichen unsere Methode auch mit verschiedenen klassifikationsbasierten Techniken des überwachten maschinellen Lernens. Unter Berücksichtigung der korrelierten Änderungen des Netzwerkverkehrs über die Zeit wenden wir Filtertechniken auf die Schätzergebnisse an, um die verfügbaren Bandbreitenänderungen zu verfolgen.

Aktive Messmethoden bieten Flexibilität beim Entwurf der Eingabestruktur. Im Gegensatz dazu besteht die überwiegende Mehrheit des Internetverkehrs aus *Transmission Control Protocol* (TCP)-Flüssen, die ein chaotisches Verkehrsmuster aufweisen. Wir untersuchen, wie die Theorie der aktiven Messungen dazu verwendet werden kann, relevante Informationen aus passiven TCP-Messungen zu extrahieren. Wir erweitern unsere Methode, sodass die Schätzung lediglich mithilfe von senderseitigen Messungen von TCP-Daten und Be-

stätigungspaketen durchgeführt wird. Stockender Querverkehr, mehrere *tight links* und Paketverlust auf dem umgekehrten Pfad können jedoch den Abstand zwischen den Bestätigungspaketen ändern, sodass das Messrauschen erhöht wird. Um zuverlässige Schätzungen der verfügbaren Bandbreite aus den von Rauschen betroffenen Lücken zwischen den Bestätigungspaketen zu erhalten, schlagen wir ein Verfahren auf Basis von neuronalen Netzwerken vor.

Wir führen eine umfassende Messstudie in einer kontrollierten Netzwerk-Testumgebung an der Leibniz Universität Hannover durch. Wir evaluieren unsere vorgeschlagenen Methoden unter einer Vielzahl von schwierigen Netzwerkbedingungen, die nicht in das Training einbezogen wurden, wie zufällig generierte Netzwerke mit mehreren *tight links*, hohe Burstartigkeit des Datenverkehrs, Verzögerungen und Paketverlust. Unsere Testergebnisse zeigen, dass unsere vorgeschlagenen maschinellen Lerntechniken die verfügbare Bandbreite mit hoher Präzision mit aktiven und passiven Messungen ermittelen können. Unsere reinforcement learning-basierte Methode erreicht darüberhinaus genaue Ergebnisse und schnelle Konvergenz ohne Trainingphase.

Schlagwörter: Schätzung verfügbarer Bandbreite, Aktive und Passive Messungen, maschinelles Lernen, Support Vector Machines, k-*Nearest-Neighbors*, Bagging, AdaBoost, neuronales Netz, bestärkendes Lernen.

CONTENTS

I DISSERTATION 1

1 INTRODUCTION 3
- 1.1 Available Bandwidth in Packet-Switched Networks 4
- 1.2 Key Issues in Available Bandwidth Estimation 4
- 1.3 Thesis Contribution 7
- 1.4 Thesis Outline 10

2 AVAILABLE BANDWIDTH ESTIMATION TECHNIQUES 11
- 2.1 Passive Measurements 11
- 2.2 Active Probing Measurements 12
  - 2.2.1 Probe Gap Model 13
  - 2.2.2 Probe Rate Model 16

3 PROBLEM STATEMENT 21

4 REGRESSION-BASED ACTIVE AVAILABLE BANDWIDTH ESTIMATION 23
- 4.1 Related Work 24
- 4.2 Model-based Reference Implementations 25
  - 4.2.1 Iterative probing 26
  - 4.2.2 Direct probing 26
- 4.3 Neural Network-based Method 26
  - 4.3.1 Scale-invariant Implementation 27
  - 4.3.2 Training Data: Exponential Cross Traffic, Single Tight Link 28
  - 4.3.3 Evaluation: Exponential Cross Traffic, Single Tight Link . 29
  - 4.3.4 Network Parameter Variation Beyond the Training Data . 32
- 4.4 Variation of the Tight Link Capacity and Multiple Tight Links . . 33
  - 4.4.1 Random Networks 33
  - 4.4.2 Multiple Tight Links 38
- 4.5 Iterative Neural Network-based Method 44
  - 4.5.1 Partly Populated Input Vectors 45
  - 4.5.2 Recommender Network for Probe Rate Selection 45
  - 4.5.3 Neural Network as Available Bandwidth Classifier 46
  - 4.5.4 Evaluation of other Machine Learning Techniques 49

5 MULTI-CLASS CLASSIFICATION-BASED ACTIVE AVAILABLE BANDWIDTH ESTIMATION 53
- 5.1 Model-based Reference Implementation 54

5.1.1 Direct probing . . . 54
5.2 Reinforcement Learning-based Method . . . 56
5.2.1 ε-greedy Algorithm . . . 56
5.2.2 Choice of Reward Function . . . 58
5.2.3 Convergence Speed . . . 59
5.3 Experimental Evaluation . . . 62
5.3.1 Cross Traffic Burstiness . . . 62
5.3.2 Cross Traffic Intensity . . . 64
5.3.3 Multiple Tight Links . . . 64
5.3.4 Tight Link differs from Bottleneck Link . . . 64
5.4 Supervised Learning-based Method . . . 68
5.4.1 Available Bandwidth Classes . . . 69
5.4.2 Training Data . . . 70
5.4.3 Machine Learning Techniques . . . 71
5.4.4 Evaluation Metrics . . . 73
5.5 Performance Evaluation . . . 76
5.5.1 Evaluation: Exponential Cross Traffic, Mutually Exclusive Classes . . . 76
5.5.2 Network Parameter Variation beyond the Training Data . . . 79
5.5.3 Effect of Number of Packet Trains . . . 87
6 PASSIVE AVAILABLE BANDWIDTH ESTIMATION 91
6.1 TCP Throughput as Available Bandwidth Estimator . . . 92
6.2 Experimental Setup . . . 93
6.3 Estimation from Passive Measurements . . . 95
6.3.1 Model-based Passive Estimation Method . . . 95
6.3.2 Experimental Verification . . . 97
6.4 Estimation from TCP Measurements . . . 99
6.4.1 TCP (g_{in}, g_{out}) Characteristics . . . 99
6.4.2 Parameter Evaluation . . . 101
6.5 Estimation from Sender-side TCP Measurements . . . 103
6.5.1 Acknowledgment Gaps . . . 103
6.6 Neural Network-based Method . . . 106
6.6.1 Data-binning Implementation . . . 106
6.6.2 Training Data: Exponential Cross Traffic, Single Tight Link 107
6.6.3 Evaluation: Exponential Cross Traffic, Single Tight Link . . . 107
6.6.4 Network Parameter Variation beyond the Training Data . . . 108
6.7 Multiple Tight Links . . . 110
6.8 Evaluation of Loss . . . 111
6.8.1 Training for Loss . . . 114

7 CONCLUSIONS AND FUTURE WORK 115
7.1 Conclusions 115
7.2 Future Work 117

II APPENDIX 119

DATA SET 121

BIBLIOGRAPHY 123

PUBLICATIONS 129

CURRICULUM VITAE 131

LIST OF FIGURES

Figure 2.1 Active network measurement. 13
Figure 2.2 FIFO multiplexer 14
Figure 2.3 Gap response curve. 14
Figure 2.4 Measurements of g_{in} and g_{out} compared to the fluid-flow model. 15
Figure 2.5 Rate response curve. 17
Figure 2.6 Rate response curve in the presence of exponential cross traffic. 18
Figure 4.1 Dumbbell topology set up for multiple tight links. 28
Figure 4.2 Testing, interpolation and extrapolation 31
Figure 4.3 Available Bandwidth Estimates (ABE) for cross traffic burstiness. 33
Figure 4.4 ABE for exponential cross traffic intensities in random networks. 35
Figure 4.5 ABE for unknown cross traffic burstiness in random networks. 36
Figure 4.6 Capacity estimates for random networks. 37
Figure 4.7 ABE in multiple tight links. 39
Figure 4.8 Tight link differs from the bottleneck link. 40
Figure 4.9 Rate response curves of a two-hop network. 41
Figure 4.10 ABE obtained from the neural network trained for a single tight link. 42
Figure 4.11 ABE obtained from the neural network trained for different tight link and bottleneck link. 43
Figure 4.12 Error in the ABE for randomly selected probe rates. . . . 46
Figure 4.13 Error in the ABE for recommended probe rates. 47
Figure 4.14 Classification maps for individual and full classifier. . . . 48
Figure 4.15 Evaluation of supervised machine learning techniques. . 51
Figure 5.1 An agent-environment interaction. 57
Figure 5.2 Reward function 59
Figure 5.3 Choice of hyperparameters 60
Figure 5.4 Randomly selected repeated experiments. 63
Figure 5.5 ABE and their SD for different cross traffic burstiness. . . 65
Figure 5.6 ABE and their SD for different exponential cross traffic. . 66

Figure 5.7 ABE and their SD for multiple tight links. 67
Figure 5.8 ABE and their SD for tight link different from bottleneck link. 68
Figure 5.9 ABE classes. 69
Figure 5.10 The maximum error in classification 75
Figure 5.11 Confusion matrix for SVM 77
Figure 5.12 ABE estimates for exponential cross traffic. 78
Figure 5.13 ABE estimates for CBR cross traffic. 81
Figure 5.14 ABE estimates for Pareto cross traffic. 82
Figure 5.15 ABE estimates for exponential cross traffic in the two-hop network. 84
Figure 5.16 ABE estimates for exponential cross traffic in the three-hop network. 85
Figure 5.17 ABE estimates for exponential cross traffic in the four-hop network. 86
Figure 5.18 ABE estimates for exponential cross traffic using five packet trains. 88
Figure 5.19 ABE estimates for exponential cross traffic using four packet trains. 89
Figure 6.1 Average TCP throughput. 93
Figure 6.2 Dumbbell topology set up for TCP passive measurements. 94
Figure 6.3 Gap response of samples from a chunk of 1 MB. 98
Figure 6.4 ABE for varying cross traffic rates. 99
Figure 6.5 Gap response curve for varying OWD 100
Figure 6.6 Parameter evaluation. 102
Figure 6.7 Multi-gap and ack-gap models 104
Figure 6.8 Gap response curve of input and acknowledgment gaps. 105
Figure 6.9 ABE obtained from individual gaps, multi-gaps, and ack-gaps. 106
Figure 6.10 ABE obtained from ack-gaps for different exponential cross traffic rates. 108
Figure 6.11 ABE for different types of cross traffic burstiness. 109
Figure 6.12 ABE for different One Way Delay (OWD)s. 110
Figure 6.13 ABE for multiple tight links. 111
Figure 6.14 Use of the ack-gap model in the presence of packet loss. 112
Figure 6.15 ABE for TCP CUBIC and BBR in the presence of loss. . . 113

LIST OF TABLES

Table 5.1 Performance evaluation for exponential traffic 78
Table 5.2 Performance evaluation for CBR traffic 81
Table 5.3 Performance evaluation for Pareto traffic 82
Table 5.4 Performance evaluation for exponential traffic in a 2-hop network 84
Table 5.5 Performance evaluation for exponential traffic in a 3-hop network 85
Table 5.6 Performance evaluation for exponential traffic in a 4-hop network 86
Table 5.7 Performance evaluation for exponential traffic using $p = 5$ packet trains 88
Table 5.8 Performance evaluation for exponential traffic using $p = 4$ packet trains 89

LIST OF SYMBOLS

A	available bandwidth
$\hat{A}$	estimated available bandwidth
$\bar{A}$	average available bandwidth
A_c	class to which available bandwidth belongs
C	capacity
$\mathbb{Z}^+$	set of positive integers
λ	average rate of cross traffic
H	network hop
l	packet length
g_{in}	input gap
g_{out}	output gap
r_{in}	input rate
r_{out}	output rate
δ_r	constant rate increment
p	number of packet trains
n	number of packets in a packet train
t_{out}^j	time when the j^{th} packet arrives at the receiver
P	probability distribution
$\mathcal{A}$	a set of actions
$\mathcal{R}$	a set of rewards
$\mathcal{S}$	a set of states
ρ	reward metric
K_r	number of repeated experiments
ξ	inter-packet strain
$z(t)$	measured strain
A	transition matrix
$w(t)$	process noise
Q	covariance matrix of process noise
$v(t)$	measurement noise
R	covariance matrix of measurement noise
$H(t)$	observation matrix
I	identity matrix
N	number of packets in a chunk
α_s	shape parameter for Pareto traffic

N_b	number of bins
N_s	number of samples
N_c	number of available bandwidth classes
K	number of folds for cross-validation
k	number of the nearest neighbors

ACRONYMS

ABE	Available Bandwidth Estimates
ACK	Acknowledgment
AdaBoost	Adaptive Boosting
AQM	Active Queue Management
ARPANET	Advanced Research Projects Agency Network
Bagging	Bootstrap Aggregation
BBR	Bottleneck Bandwidth and Round-trip propagation time
CAC	Call Admission Control
CBR	Constant Bit Rate
CoDel	Controlled Delay
CPU	Central Processing Unit
CRUDE	Collector for RUDE
CWND	Congestion Window
DASH	Dynamic Adaptive Streaming over HTTP
DAG	Data Acquisition and Generation
D-ITG	Distributed Internet Traffic Generator
GPR	Gaussian Process Regression
GP	Gaussian Process
GRBF	Gaussian Radial Basis Function
ECOC	Error-Correcting Output Codes
ETOMIC	European Traffic Observatory Measurement InfrastruCture
FIFO	First-In, First-Out
HyStart	Hybrid Start
IP	Internet Protocol
IC	Interrupt Coalescing
IGI/PTR	Initial Gap Increasing/Packet Transmission Rate
k-NN	k-Nearest Neighbor
LTE	Long-Term Evolution
MDP	Markov Decision Process
MLP	Multi Layer Perceptron
MPEG	Moving Picture Experts Group

MTU	Maximum Transmission Unit
NIC	Network Interface Controller
NS	Network Simulator
NN	Neural Network
NNC	NN-based classifier
NNR	NN-based regression
OWD	One Way Delay
P2P	Peer-to-Peer
PGM	Probe Gap Model
PRM	Probe Rate Model
QoS	Quality of Service
RTT	Round Trip Time
RUDE	Real-time UDP Data Emitter
SCG	Scaled Conjugate Gradient
SD	Standard Deviation
SVM	Support Vector Machine
SVC	Support Vector Classification
SVR	Support Vector Regression
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
VLAN	Virtual Local Area Network
WWW	World Wide Web

Part I

DISSERTATION

1

INTRODUCTION

With the inception of the computer age, the Internet has become a super-network worldwide. Since its origin as Advanced Research Projects Agency Network (ARPANET) in 1969, it has evolved in terms of network technologies, e.g., from dial-up to the optical fibers and from classic text-based to audio and video applications. It has become a significant medium that connects a billion of users across the world primarily through World Wide Web (WWW) and provides services, for example, e-mail, web browsing, Voice over Internet Protocol (VoIP), video streaming. For the majority of the end-users with limited knowledge about network performance, the Internet is only a black box that receives and forwards packets between end-hosts.

The motivation behind bandwidth estimation is the potential benefits that users can gain from the knowledge of bandwidth characteristics of a network path. For example, the information about the state and behavior of the networks can significantly improve the Quality of Service (QoS) guarantees that the users may receive from the network. Furthermore, Internet traffic engineering can be used to control and manage Internet traffic [1]. The congestion control algorithms and rate-adaptive applications may use the bandwidth estimates to transfer the highest possible quality over the Internet while avoiding network congestion [2, 3]. In Peer-to-Peer (P2P) networks, accurate information about the available bandwidth can help in load balancing [4]. Call Admission Control (CAC) can take advantage of available bandwidth information to ensure enough bandwidth for authorized flows in VoIP networks [5]. Dynamic overlay routing can enhance the reliability and performance of Internet Protocol (IP) networks by selecting the paths based on available bandwidth measurements [6].

The remainder of this chapter is structured as follows: In Sec. 1.1, we define the term available bandwidth in the context of packet-switched networks. We address the key issues in available bandwidth estimation in Sec. 1.2. In Sec. 1.3, we discuss the main contributions of this thesis in estimating the end-to-end available bandwidth in packet-switched communication networks. We provide thesis outline in Sec. 1.4.

1.1 AVAILABLE BANDWIDTH IN PACKET-SWITCHED NETWORKS

Today's Internet is a quintessential packet-switched network where data is transmitted between the sender and the receiver in the form of small chunks called packets. These packets are forwarded by intermediary network devices, such as routers and switches, depending upon the maximum available bandwidth of the most limiting link of an end-to-end path. In the context of data networks such as the Internet, an *end-to-end path* refers to a sequence of successive hops $H \in \mathbb{Z}^+$ or links. The *capacity* $C \in \mathbb{R}^+$ of a link refers to the maximum possible rate that the IP layer can deliver by transferring Maximum Transmission Unit (MTU)-sized IP packets of 1500 B [7]. In other words, it establishes an upper bound on the maximum achievable bandwidth in a link. For an end-to-end path, the capacity of a network path consisting of H hops is determined by the *bottleneck link*, i.e., the link with the minimal capacity as

$$C = min_{i=1..H}(C_i), \tag{1.1}$$

where C_i is the capacity of the i^{th} link. The term *available bandwidth* refers to the residual capacity of a link or a network path that is left over after the existing traffic, also referred to as cross traffic, is served. Formally, given a link with capacity C and cross traffic with long-term average rate λ, where $\lambda \in [0, C]$, the available bandwidth of a network path $A \in [0, C]$ is defined as $A = C - \lambda$ [8]. The end-to-end available bandwidth of a network path consisting of H hops is determined by its *tight link*, i.e., the link with the minimal available bandwidth as

$$A = min_{i=1..H}(A_i), \tag{1.2}$$

where $A_i \in [0, C_i]$ is the available bandwidth of the i^{th} link and $C_i \in \mathbb{R}^+$ is the capacity of the same link. The tight link may differ from the bottleneck link.

1.2 KEY ISSUES IN AVAILABLE BANDWIDTH ESTIMATION

Since estimating the available bandwidth is crucial to the effective deployment of QoS guarantees in a network, there has been much work done throughout the years on developing the tools for estimating the capacity and available bandwidth of network paths. These tools are passive, i.e., based on non-intrusive monitoring of real traffic, or active, i.e., based on injecting artificial probe traffic into the network. Despite five decades of intensive development of the bandwidth estimation techniques, an accurate estimation of available bandwidth is

still considered a challenging task because of its dynamics, especially in the Internet's environment. Some key issues involved in the bandwidth estimation are as follows:

- ***Measurement bias***: There exist a number of measurement tools that are based on injecting artificial probe traffic with a well-defined input gap g_{in} into the network. As these probe packets traverse through the network, they get dispersed due to cross traffic. At the receiver, the output gap of the received probe g_{out} is measured to deduce the available bandwidth. The existing measurement tools are based on the assumptions of a *single-hop*, *lossless* network that is modeled as a *First-In, First-Out (FIFO)* multiplexer of probe and cross traffic. It is assumed that cross traffic has a *constant rate* λ and behaves like *fluid*, i.e., it is infinitely divisible. However, when there occur deviations from the assumptions of the fluid-flow model, the estimation tools based on the above-mentioned assumptions produce measurement bias. Following are some examples of such scenarios:
  - *Random cross traffic*: The burstiness of the cross traffic violates the Constant Bit Rate (CBR) assumption of the model and can cause queueing at the tight link even if the probe rate is below the average available bandwidth. The result is an increase in the variability as well as underestimation of available bandwidth estimates [8, 9].
  - *Packet interference*: Non-conformance with the fluid model arises due to the interaction of probe traffic with discrete packets of the cross traffic. The non-fluid cross traffic affects the relation of output gaps g_{out} and input gaps g_{in} that is used to infer the available bandwidth [8, 9].
  - *Multiple tight links*: If cross traffic is non-fluid, the repeated packet interaction at each of the links distorts the probe gaps. Further, in the case of random cross traffic, there may not be a single tight link, but the tight link may vary randomly. The consequence is an underestimation of the available bandwidth [9, 10] which is analyzed in [11, 12].
  - *End-to-end pathologies*: The term *pathology* in the context of networks refers to unusual or unexpected network behavior. The unusual behavior includes *packet loss*, i.e., the network is unable to deliver all the transmitted packets, *out-of-order delivery*, i.e., packets arrive at the receiver in a different order than that in which they were sent, *packet replication*, i.e., the network delivers multiple copies of the same packet, and *packet corruption*, i.e., the network delivers an im-

perfect packet which differs from the originally sent packet. It is difficult to estimate available bandwidth in the presence of such network pathologies. For example, out-of-order delivery or lost packets distort the output gaps for packet pairs [13].

- *Queueing behavior*: The existing bandwidth estimation tools are based on the assumption that all links apply FIFO queues. This assumption is violated by the existing intermediary routers which have diverse queueing behavior. For example, to avoid network congestion a number of routers use Active Queue Management (AQM) algorithms, e.g., Controlled Delay (CoDel) [14] to drop the packets inside a buffer associated with a Network Interface Controller (NIC) [15].

- ***Measurement inaccuracies***: It is difficult to tailor the bandwidth estimation methods to specific hardware implementations that influence the measurement accuracy. Most of the state-of-the-art bandwidth estimation tools use packet capture libraries, for example, *libpcap*, where each packet is given a timestamp representing the sending and receiving time. There are several reasons which cause measurement inaccuracies such as:

  - *Resolution of the system timer*: The resolution of a timer depends on the underlying operating system and hardware of the end-host. Most platforms support a resolution of 1 ms. This resolution is too coarse for estimation tools that require accurate timing [15, 16]. Furthermore, for the end-hosts with multi-core processors, packets timestamped by different cores might not have consistent time stamps if the multi-cores are not running at the same speed, or their time counters are not synchronized.

  - *Context switching*: A context switch is the act of switching of the Central Processing Unit (CPU) from one process to another, i.e., the execution of one thread is suspended in favor of another [17]. In the case of multiple processes sharing a single CPU, the inter-packet spacing can be affected if the estimation process is paused by OS kernel abruptly and system resources are released to another process. For example, while estimating the available bandwidth from Transmission Control Protocol (TCP) measurements for long distances, where Round Trip Time (RTT) is greater than the context switch time, at least every alternative measurement is interrupted [16].

  - *Clock synchronization and resolution*: In real-time systems, two clocks on sender and receiver are rarely perfectly synchronized. The clocks may have different values and may run at different speeds. The relative difference of time as reported by two clocks is called offset,

and the relative difference of their frequencies is called skew. Due to offset and skew, the timestamps of input and output probe traffic become inaccurate leading to bias in estimation [18].

- *Interrupt Coalescing (IC)* is a feature implemented in the hardware of many high-performance NICs to avoid flooding the hosts with too many interrupts. Instead of generating an interrupt at the arrival of each packet, multiple packets are grouped and served in a single interrupt. Though IC technique can improve the performance of high-speed NICs, an incoming packet which is possibly delayed by NIC is no longer able to get an accurate timestamp from the OS kernel [19, 20].
- *System call* provides the means through which bandwidth estimation tools running in the user space programs can access kernel services. The system calls that are frequently invoked by estimation tools are gettimeofday() to obtain the timestamps of packets, and read()/write() function, e.g., to read/write these packets from the NIC buffer to the memory in kernel space. For most Linux-based operating systems, delay added by gettimeofday(), and read()/write() is up to 1 µs and 30 µs respectively [15, 16].

- ***End-host throughput***: To estimate the available bandwidth, both end-hosts must be able to send probes at rates higher than the available bandwidth, otherwise, only the maximum throughput of the slower end-host can be measured. In recent years, network bandwidth has become faster than end-host throughput making it harder to estimate the capacity and the available bandwidth [15, 16].

1.3 THESIS CONTRIBUTION

In this work, while tackling the above-mentioned challenges in available bandwidth estimation, we make the following contributions:

- ***Active available bandwidth estimation***: To alleviate the variability of noise-afflicted packet gaps, we use standard packet train probes, which are reported to be more robust to random fluctuations. Although packet trains help to reduce the variability in available bandwidth estimations, they do not take care of the systematic deviations from the deterministic fluid-flow model that cause biased estimates [8, 11, 21]. Therefore, to reduce the bias in bandwidth estimates, we investigate how to benefit from machine learning techniques while using standard packet train probes for available bandwidth estimation. We propose three methods for active

available bandwidth estimation; the first method is based on regression and the other two methods are based on multi-class classification.

- ***Regression-based method***: We implement a regression-based supervised learning technique for bandwidth estimation. Because this is a scale-invariant approach, we are able to apply the method to networks with arbitrarily changing capacity without extra training. Unlike the binary search performed in [20], our method chooses the probe rate that is expected to improve the bandwidth estimate the most, and hence, can effectively control the estimation procedure in an iterative implementation. In addition to the regression-based estimator, we also implement a binary classifier. Unlike the classifier in [20] that takes only one probe rate as input to determine whether the current probe rate is above or below the available bandwidth, our classifier uses information of all previous probe rates to classify the probe rates into two groups, i.e., the rates above and below the available bandwidth, which leads to more accurate estimates. We also evaluate our method with other supervised machine learning techniques such as Support Vector Regression (SVR), Gaussian Process Regression (GPR) and Bootstrap Aggregation (Bagging).
- ***Multi-class classification-based method***: We formulate available bandwidth estimation as a multi-class classification problem and estimate the available bandwidth class to which an instantaneous available bandwidth value belongs. We propose the following classification-based methods:
  - ***Reinforcement learning-based***: We employ a reinforcement learning-based multi-class estimation algorithm that does not require a training phase, unlike supervised learning-based techniques. We formulate available bandwidth estimation as a single-state Markov Decision Process (MDP) multi-armed bandit problem. We consider a set of input rates as a set of actions and define a reward metric as a function of input and output rates. We implement the ε-greedy algorithm, which reports the available bandwidth as the input rate that maximizes a reward metric after an exploration-exploitation mechanism is employed. Compared to a model-based direct probing technique that employs a Kalman filter, our method shows more accurate estimates and faster convergence and does not require measurement-noise statistics.
  - ***Supervised learning-based using reduced probe traffic***: Since there is no training phase in the reinforcement learning-based method and the system learns the network in an online manner, the con-

vergence time may be longer in certain network scenarios, e.g., when available bandwidth values change very frequently. Therefore, we propose another classification-based method employing supervised learning that is designed to obtain reliable available bandwidth estimates in randomly varying network conditions using fewer probe packets. We divide a entire possible range of available bandwidth, e.g., from zero bits/second to the capacity of a network, into small subranges and assign a class to each subrange. We design a classifier that takes an input feature vector of probe rates and results in the class that the available bandwidth belongs to. We evaluate our classification-based method employing supervised learning techniques such as Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), Bagging, Adaptive Boosting (AdaBoost) and Neural Network (NN). We show that our method performs better than the fluid-flow model-based direct probing technique that employs a Kalman filter. Furthermore, considering the correlated changes in a network's traffic through time, we apply filtering techniques on the estimation results in order to track the available bandwidth changes.

- ***Passive available bandwidth estimation***: We estimate the available bandwidth from existing passive TCP measurements without injecting any artificial probe traffic into the network. We identify the chaotic, non-packet train pattern of short TCP flows, and define a criterion to select traffic samples that bear relevant information. We use a regression technique to obtain robust bandwidth estimates from the passive measurements of packet gaps. The accuracy of the method is evaluated for a variety of relevant parameter settings. We propose a method that can take multiple gaps as well as acknowledgment gaps as input. This extension enables bandwidth estimation using only sender-side measurements of TCP data and acknowledgment packets. However, the acknowledgment gaps get altered due to random cross traffic, multiple bottleneck links, and packet loss, which increase the measurement noise resulting in the underestimation of the available bandwidth [8, 9]. We propose a neural network-based method that can deal with distorted acknowledgment gaps to obtain reliable available bandwidth estimates using sender-side TCP passive measurements.

- ***Providing data sets for performance evaluation***: We generate different data sets for training and testing using a controlled network testbed at Leibniz University Hannover, which can be downloaded from [22]. We present a broad evaluation of our methods using a wide range of ran-

domly generated topologies with largely varying capacities, cross traffic intensities, and burstiness. We specifically target the network scenarios that are known to be hard in bandwidth estimation, i.e., where the assumptions of the deterministic fluid-flow model are not satisfied such as bursty cross traffic, packet loss, and multiple tight links. Furthermore, we present a deeper investigation of multi-hop networks. We specifically consider cases where tight link differs from the bottleneck link and show how the order in which these occur in a network path affects the input-output gap relation that is used for available bandwidth estimation.

1.4 THESIS OUTLINE

The remainder of this thesis is structured as follows:

In chapter 2, we introduce the basic Probe Gap Model (PGM), respectively, Probe Rate Model (PRM) of a FIFO multiplexer that is used in bandwidth estimation. Further, we discuss state-of-the-art bandwidth estimation methods.

In chapter 3, we formulate the research problems addressed in this thesis. It focuses on the current limitations of the state-of-the-art available bandwidth estimation methods and provides us the motivation to investigate them.

In chapter 4, we propose a regression-based supervised learning method that is motivated by the characteristic rate response curve of a network. We investigate how to benefit from machine learning, specifically neural networks while using standard packet train probes for available bandwidth estimation.

In chapter 5, we investigate how reinforcement learning can be utilized in available bandwidth estimation. We propose a method that runs ε-greedy algorithm to find the available bandwidth by maximizing the designated reward function. To obtain reliable estimates with minimal probe-traffic overhead in network scenarios where the channel characteristics such as capacity and cross traffic intensity change over time randomly, we propose another classification method based on supervised learning. We further investigate the use of filtering techniques to smooth the variations in the available bandwidth estimates.

In chapter 6, motivated by the shortcomings of TCP throughput as available bandwidth estimator, we investigate how techniques from active probing can benefit TCP passive bandwidth estimation. We extend the proposed method to estimate the bandwidth from sender-side measurements using input and acknowledgment gaps. Furthermore, to deal with noise-afflicted packet gaps, we propose a neural network-based method for estimating the available bandwidth.

Finally, in chapter 7 we conclude the thesis and propose a perspective for future work.

2

AVAILABLE BANDWIDTH ESTIMATION TECHNIQUES

Owing to the intensive development of access technologies and network applications, monitoring the network performance through the estimation of end-to-end path properties such as available bandwidth has been the focus of many important research activities. Throughout the years, several measurement tools based on active and passive measurements have been developed and evaluated. The former injects artificial probe traffic into the network while the latter non-intrusively monitors the real traffic. In this chapter, we provide a brief introduction to both these measurements techniques.

2.1 PASSIVE MEASUREMENTS

Passive measurement tools are based on the non-intrusive monitoring of traffic. These techniques involve capturing and analyzing live network traffic at a point of interest in the network and require direct access to the router or to any node in the network path where measurements are to be taken. To obtain the more holistic view of the network's performance, one is reliant upon sufficient traffic flowing through the measurement point. It requires incorporating some additional intelligence into network devices to enable them to identify, filter, and record the traffic characteristics. Unlike active probing techniques, passive monitors don't strain the network by injecting probe traffic into the network. Therefore, it is favorable if the available bandwidth can be estimated from passive measurements of existing network traffic. However, the difficulty of passive measurements is that the input rate cannot be controlled so that it is hard to extract the desired information [23]. Additional challenges include the capturing speed during the actual measurement session, storage capacities for longer traces and processing power for off-line analysis of data traces [24]. Furthermore, to employ passive measurements for monitoring real traffic, one needs to access the complete information about all packets on the network, which might involve privacy or security issues about how to access/protect the gathered traces. Since it is a non-trivial task to obtain network information from

passive measurements, therefore most of the measurement tools are based on active probing.

2.2 ACTIVE PROBING MEASUREMENTS

The active probing techniques use a sender that actively injects artificial *probe traffic* into the network with a defined packet size l and inter-packet gap referred to as input gap g_{in} as shown in Fig. 2.1. As these probe traffic traverse through the network path, they interact with the cross traffic and reach the receiver with a new packet gap referred to as the output gap g_{out}. The packet pair dispersion $\Delta = g_{out} - g_{in}$ that arises due to the interaction between the probe traffic and cross traffic packets is analyzed to deduce the available bandwidth. The dispersion is positive, when cross traffic packets get inserted between the probing packets, i.e., $g_{out} > g_{in}$. The dispersion is zero when $g_{out} = g_{in}$, which implies that the link is not saturated. However, in practical scenarios, in the presence of discrete random cross traffic or measurement inaccuracies, the dispersion may become negative, i.e., $g_{out} < g_{in}$. The negative dispersion is, however, considered infeasible and hence discarded by bandwidth estimation tools.

Unlike passive measurements, the active probing tools provide flexibility in the design of probe streams with particular properties to match measurement requirements. For example, probes are typically sent either as *packet pairs* [25] with a defined spacing referred to as input gap g_{in}, as *packet trains* [13, 26] with a fixed input rate r_{in}, or as *packet chirps* [27] that are packet trains with an increasing rate. But the vast majority of the Internet traffic is TCP flows that exhibit a rather chaotic non-packet train traffic pattern. For certain applications, the intrusive active probing may decrease their performance and hence the satisfaction level of the users. For example, they can affect the response time of TCP connections [28]. Furthermore, on detecting congestion due to intrusive probing, TCP adjusts its Congestion Window (CWND), which affects its performance adversely in scenarios with long RTT. In the worst case, due to high-intensity probing significant delays, or even packet loss may occur which are modeled as infinite delays in [12] producing estimation bias. Therefore, in certain network conditions, it is favorable if the available bandwidth can be estimated from passive measurements of existing network traffic.

In the following subsection, we introduce the basic PGM, respectively, PRM of a FIFO multiplexer that is used in bandwidth estimation to derive its characteristic response curve. Also, we discuss the state-of-the-art bandwidth estimation methods.

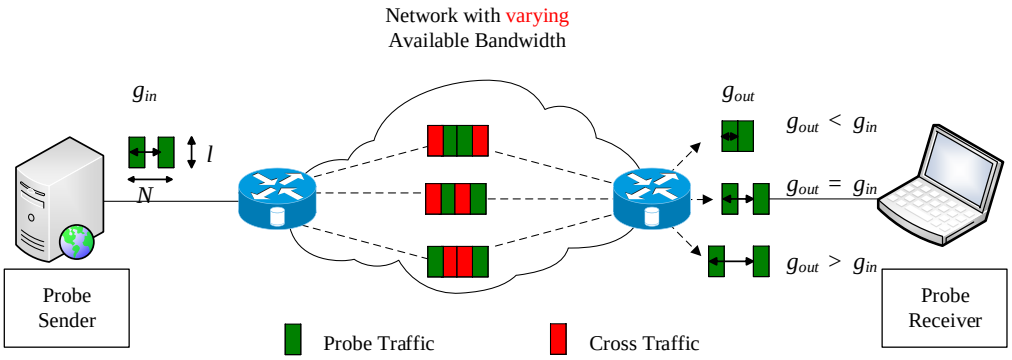


Figure 2.1: Active network measurement in a single tight link.

2.2.1 *Probe Gap Model*

A common assumption in bandwidth estimation is that the available bandwidth, respectively, the rate of the cross traffic does not change during a probe. Further, to simplify modeling, cross traffic is assumed to behave like a fluid, i.e., effects that are due to the packet granularity of the cross traffic are neglected. Modeling a single tight link as a lossless FIFO multiplexer of probe and cross traffic, the relation of g_{out} and g_{in} is followed by an intuitive argument [8] as

$$g_{out} = \max\left\{g_{in}, \frac{g_{in}\lambda + l}{C}\right\}. \tag{2.1}$$

The reasoning is that during g_{in} an amount of $g_{in}\lambda$ of the fluid cross traffic is inserted between any two packets of the probe traffic, so that the probe packets may be spaced further apart as shown in Fig 2.2. Reordering Eq. (2.1) gives the characteristic *gap response curve*

$$\frac{g_{out}}{g_{in}} = \begin{cases} 1 & \text{if } \frac{l}{g_{in}} \leqslant C - \lambda, \\ \frac{l}{g_{in}C} + \frac{\lambda}{C} & \text{if } \frac{l}{g_{in}} > C - \lambda. \end{cases} \tag{2.2}$$

The utility of Eq. (2.2) is that it shows a clear bend at $A = C - \lambda$, that enables estimating the available bandwidth using different techniques based on PGM. We note that the quotient of packet size and gap is frequently viewed as the data rate of the probe. For example, consider a tight link with capacity $C = 100$ Mbps and cross traffic with an average rate of $\lambda = 62.5$ Mbps. The packet size is $l = 1514$ B, resulting in a transmission time l/C of about 120 µs. Given an input gap $g_{in} = 270$ µs, the output gap follows from Eq. (2.1) as $g_{out} = 290$ µs. Now, assume for a moment that C is known but λ is unknown. An active probing tool can send probes with, e.g., $g_{in} = 270$ µs to measure g_{out}. Noting that $g_{out}/g_{in} > 1$, Eq. (2.2) reveals the unknown $\lambda = (g_{out}C - l)/g_{in} = 62.5$ Mbps.

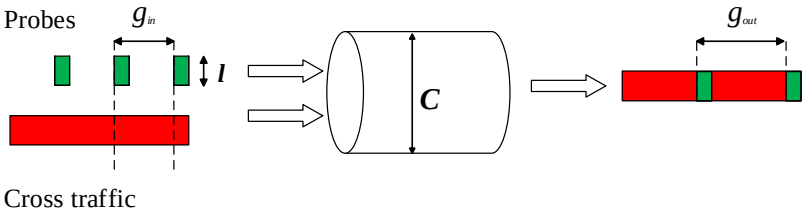


Figure 2.2: FIFO multiplexer with capacity C and fluid constant rate traffic λ

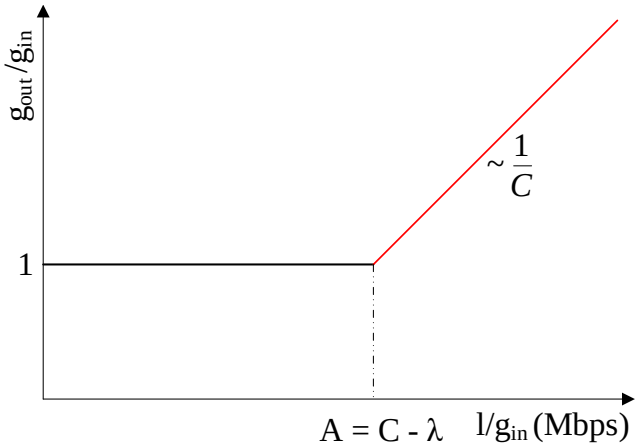


Figure 2.3: Gap response curve. The bend marks the available bandwidth.

In practice, the observations of g_{out} are distorted for various reasons as discussed in Sec. 1.2. For example, we recorded a measurement trace of 50 pairs of g_{in} and g_{out} in the network testbed in Fig. 4.1 with a single tight link and the above parameters C, λ, and l, where l is the maximal size of Ethernet packets including the header. The results are shown in Fig. 2.4. Neglecting the cases where $g_{out} < g_{in}$ that are not possible in the model and ignoring large outliers, a range of samples g_{out} of about 360 µs remains that suggests concluding $\lambda \approx 90$ instead of 62.5 Mbps erroneously.

2.2.1.1 *Distortions of output gap* g_{out}

We have discussed the challenges involved in estimating the available bandwidth in Sec. 1.2. In this section, we discuss the relevant reasons for the distortions in measurements of g_{out} shown in Fig. 2.4. These reasons include deviations from the assumptions of the model, i.e., a lossless FIFO multiplexer with constant, fluid cross traffic as well as measurement inaccuracies, such as imprecise timestamping:

- *Random cross traffic:* Eq. (2.2) is deterministic and hence it does not define how to deal with the randomness of g_{out} that is caused by variable bit rate cross traffic. It is shown in [8] that the problem cannot be easily fixed by using the expected value $E[g_{out}]$ instead. In brief, this is due to the

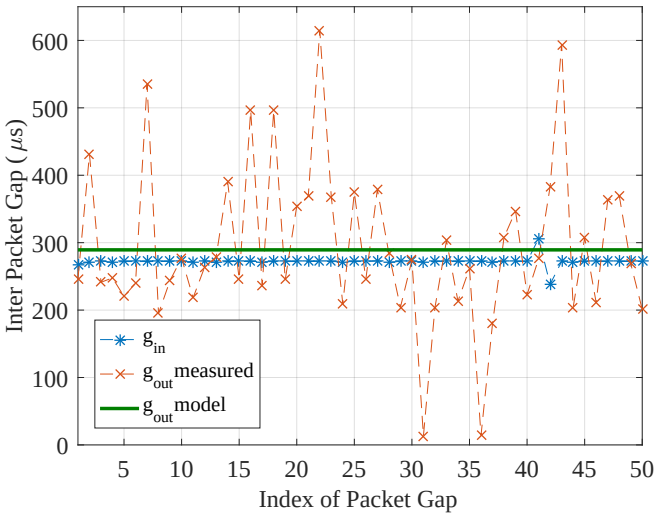


Figure 2.4: Measurements of g_{in} and g_{out} compared to the fluid model. The network has a single tight link with capacity $C = 100$ Mbps and exponential cross traffic with rate $\lambda = 62.5$ Mbps. The packet size is $l = 1514$ B.

nonlinearity of Eq. (2.2) and the fact that the location of the turning point $C - \lambda$ fluctuates if the rate of the cross traffic λ is variable. The result is a deviation that is maximal at $l/g_{in} = C - \lambda$ and causes underestimation of the available bandwidth [8, 9].

- *Packet interference:* Non-conformance with the fluid model arises due to the interaction of probes with packets of the cross traffic. In Fig. 2.4, two relevant examples are identified by frequent samples of g_{out} in the range of 240 and 360 µs, respectively. In contrast, the g_{out} of 290 µs that is predicted by the fluid model, is observed rarely. To understand this effect, consider two probe packets with $g_{in} = 270$ µs and note that the transmission time of a packet is 120 µs. The case $g_{out} = 360$ µs occurs if two cross traffic packets are inserted between the two probe packets. Instead, if one of the two cross traffic packets is inserted in front of the probe, it delays the first probe packet, resulting in $g_{out} = 240$ µs.

- *Multiple tight links:* An extension of Eq. (2.1) for multiple links is derived in [29]. Yet, if cross traffic is non-fluid, the repeated packet interaction at each of the links distorts the probe gaps. Moreover, in the case of random cross traffic, there may not be a single tight link, but the tight link may vary randomly. The consequence is an underestimation of the available bandwidth [9, 10] that is analyzed in [11, 12].

- *Measurement inaccuracies:* Besides, there exist limitations of the accuracy due to the hardware of the hosts where measurements are taken. A possible clock offset between sender and receiver is dealt with by the use of

probe gaps. A problem in high-speed networks is, however, interrupt coalescing [19, 20]. This technique avoids flooding a host with too many interrupts by grouping packets received in a short time together in a single interrupt, which distorts g_{out}. In Fig. 2.4, these are identified by samples of $g_{out} >> 290$ µs.

2.2.2 *Probe Rate Model*

In order to alleviate the observed variability of the samples of g_{out}, the state-of-the-art bandwidth estimation methods perform averaging of several output gap samples g_{out}. The samples can be collected by repeatedly sending *packet pairs* [25], or *packet trains* [13, 26], which consist of n consecutive packets and hence $n-1$ input gaps. At the receiver, the consecutive output gaps are formulated as $g_{out}^j = t_{out}^{j+1} - t_{out}^j$ for $j = 1 \dots n-1$, where t_{out}^j is the time when the j^{th} packet arrives at the receiver. Then, the output rate of a packet train with n packets is given as

$$r_{out} = \frac{(n-1)l}{t_{out}^n - t_{out}^1}. \tag{2.3}$$

Since we define $g_{out}^j = t_{out}^{j+1} - t_{out}^j$, we can re-write Eq. (2.3) as

$$r_{out} = \frac{l}{\frac{1}{n-1}\sum_{j=1}^{n-1} g_{out}^j}. \tag{2.4}$$

Notice that the denominator in Eq.(2.4) converges to the mean of the output gaps with the increasing packet train size. Herein, assuming a deterministic fluid-flow model, i.e., $g_{out}^j = g_{out}$ for $\forall j$, we can see that

$$r_{out} = \frac{l}{\frac{1}{n-1}\sum_{j=1}^{n-1} g_{out}^j} = \frac{l}{\frac{(n-1)g_{out}}{n-1}} = \frac{l}{g_{out}}. \tag{2.5}$$

Similarly, defining the input rate as $r_{in} = l/g_{in}$, and inserting $r_{in} = l/g_{in}$ and $r_{out} = l/g_{out}$ into Eq. (2.1) and Eq. (2.2), we obtain r_{out} as

$$r_{out} = \min\left\{r_{in}, \frac{r_{in}}{r_{in} + \lambda} C\right\}, \tag{2.6}$$

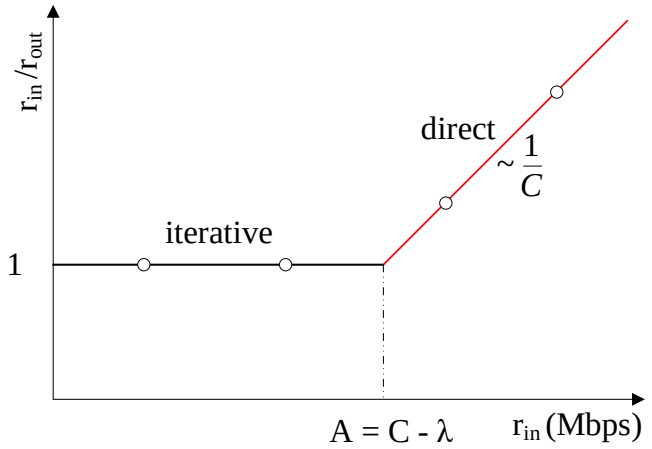


Figure 2.5: Rate response curve. The bend marks the available bandwidth.

and the equivalent representation of the *rate response curve* as

$$\frac{r_{in}}{r_{out}} = \begin{cases} 1 & \text{if } r_{in} \leqslant C - \lambda, \\ \frac{r_{in}}{C} + \frac{\lambda}{C} & \text{if } r_{in} > C - \lambda, \end{cases} \tag{2.7}$$

which mathematically describes the clear bend in the rate response curve at $r_{in} = A$ as shown in Fig. 2.5.

Active techniques for estimation of the available bandwidth can be classified to be either iterative or direct. We use the rate response curve shown in Fig. 2.5 to illustrate the difference. The same conclusions can be made if the gap response curve is used.

Iterative probing techniques basically search for the turning point of the rate response curve by sending repeated probes at increasing rates, as long as $r_{in}/r_{out} = 1$. When r_{in} reaches $C - \lambda$, the available bandwidth is saturated and increasing the probe rate further results in $r_{in}/r_{out} > 1$. As a consequence, a queue builds up at the multiplexer. This causes an increase in OWD that can be detected by the receiver. Established iterative probing tools are, e.g., Pathload [30] and Initial Gap Increasing/Packet Transmission Rate (IGI/PTR) [31]. Pathload adaptively varies the rates of successive packet trains r_{in} in a binary search until r_{in} converges to the available bandwidth. It uses feedback from the receiver that reports whether r_{in} exceeds the available bandwidth or not. The decision is made based on two statistical tests that detect increasing trends of the OWD. In comparison to Pathload, IGI/PTR tests whether $(r_{in} - r_{out})/r_{in} > \Delta^{th}$, where the threshold value Δ^{th} is set to 0.1 to detect whether the probe rate exceeds the available bandwidth. Regarding the variability of the available bandwidth, Pathload reports an available bandwidth range that is determined by the largest

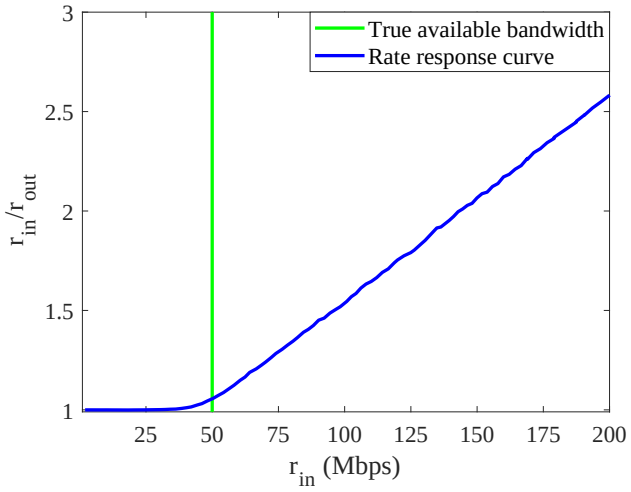


Figure 2.6: Rate response curves of a single tight link in presence of exponential cross traffic. The sharp bend around $r_{in} = C - \lambda$ that marks the available bandwidth is not clearly apparent.

probe rate that did not cause self-induced congestion and the smallest rate that did cause congestion, respectively.

Direct probing techniques estimate the upward line segment of the rate response curve for $r_{in} > C - \lambda$ instead of searching for the turning point of the rate response curve. The line is determined by C and λ. If C is known, a single probe $r_{in} = C$ yields a measurement of r_{out} that is sufficient to estimate $\lambda = C(C/r_{out} - 1)$ from Eq. (2.7). Spruce [32] implements this approach. If C is also unknown, a minimum of two different probing rates $r_{in} > C - \lambda$ is sufficient to estimate the two unknown parameters of the line. This approach is taken, e.g., by TOPP [29], DietTOPP [33], and BART [34].

The tools based on the PRM are able to alleviate the g_{out} distortions by averaging several output gap samples g_{out}, however, the curve deviates from its fluid-flow model-based prediction in the presence of random, discrete cross traffic and multiple tight links leading to biased estimates. For instance, looking at the experimental results[1] for exponential traffic with moderate burstiness in Fig. 2.6, we can see that, unlike the deterministic fluid-flow model in Fig. 2.5, the sharp bend around $r_{in} = C - \lambda$, that marks the available bandwidth is not clearly apparent. A detailed analysis of the impact of random cross traffic on the properties of rate and gap response curves is provided by [8, 11]. The main finding is an elastic deviation from the fluid-flow model that may cause biased

1 We obtained the results in Fig. 2.6 from the testbed shown in Fig. 4.1. The network is set with a single tight link of capacity $C = 100$ Mbps and access links are of capacity $C = 1$ Gbps. The cross traffic is discrete with a packet length of $l = 1514$ B. We further set the cross traffic to moderate burstiness with exponentially distributed packet inter-arrival times and the average rate of $\lambda = 50$ Mbps. Particularly, we run the experiment 1000 times and average the results for smoothness in the presentation.

estimates. The bias is significant in the middle part of the probing range around $r_{in} = C - \lambda$, where it blurs the characteristic bend of the curve. For intuition, the bend of the curve at $C - \lambda$ can be thought of as fluctuating along the x-axis if the intensity of cross traffic λ is random. Further, [8, 11] connect the concept of rate and gap response curves with known bandwidth estimation tools.

3

PROBLEM STATEMENT

In the previous chapters, we have introduced the state-of-the-art model-based bandwidth estimation techniques. In this chapter, we describe and summarize the research problems addressed in this thesis.

In the light of potential benefits that users can gain from the knowledge of network path characteristics and significance of bandwidth estimation to improve the QoS guarantees, the area of performance measurement is still a focus of ongoing research activities. Throughout the years, several available-bandwidth measurement tools based on passive [30, 35, 36] and active [8, 11, 12, 23, 26, 27, 29–34, 37] measurements have been developed. As discussed in previous chapters, these methods are based on the assumptions of the fluid-flow model. However, in practical scenarios, the methods for bandwidth estimation have to deal with noisy measurement data. For example, the observations of output gaps g_{out} are distorted for various reasons, e.g., due to inaccurate time-stamping, non-fluid traffic, multiple bottlenecks.

To alleviate the observed variability of the samples of g_{out}, state-of-the-art bandwidth estimation methods perform averaging of several g_{out} samples by sending *packet pairs* [25], or by *packet trains* [13, 26]. A common approach is the use of constant rate packet trains, which are less susceptible to random fluctuations. To improve the estimates further, different post-processing techniques are used. A typical approach is to repeat measurements several times to compute average values, as done by Pathchirp [27] and Spruce [32], or to perform a majority decision, as in the case of Pathload [30] that also reports an undecided bandwidth region. Furthermore, BART [34] uses a Kalman filter to estimate the available bandwidth from repeated measurements and to track changes of the available bandwidth online. TOPP [29] and DietTOPP [33] use linear regression. These techniques, however, do not overcome the basic assumptions of the deterministic fluid model in Eq. (2.1).

While packet trains and statistical post-processing help to reduce the variability of available bandwidth estimates, these cannot resolve systematic deviations such as the underestimation bias in case of random cross traffic and multiple tight links [8, 11, 12]. Hence, the fundamental limitations of the state-of-the-art

bandwidth estimation techniques motivated us to investigate in-depth:

The estimation of the available bandwidth using ***a reduced amount of probe traffic*** *without negatively impacting the* ***accuracy*** *in the practical network scenarios that* ***fail to comply*** *with the* ***assumptions*** *of the fluid-flow model.*

The injection of probe traffic as done in active probing, however, adds to the load of the network. As a consequence, network congestion occurs that affects the performance of existing flows [28]. Hence, it is favorable if the available bandwidth can be estimated from passive measurements of existing network traffic. But estimating the bandwidth from passive measurements comes with its own challenges. The difficulty of passive measurements is that the input rate cannot be controlled, e.g., TCP chaotic traffic pattern, which makes it hard to extract the desired information [23]. Hence, another issue that we investigate in this thesis is:

The estimation of the available bandwidth from ***chaotic, non-packet train pattern*** *of* ***TCP passive measurements****, where it is hard to extract the desired information.*

Furthermore, employing passive measurements requires direct access to the point of interest to monitor the flowing traffic. This might involve privacy or security issues about how to access/protect the gathered traces. Therefore, towards the end of this thesis we phrase another research challenge:

The estimation of the available bandwidth from ***TCP sender-side measurements*** *using* ***input and acknowledgment gaps****.*

4

REGRESSION-BASED ACTIVE AVAILABLE BANDWIDTH ESTIMATION

In this chapter, we investigate how to benefit from machine learning, specifically neural networks, while using standard packet train probes for available bandwidth estimation. Compared to packet chirps, that are favored in the related works [20, 38, 39], packet trains have been reported to be more robust to random fluctuations. In fact, the implementation of a chirp as a multi-rate probe [20], that concatenates several packet trains with increasing rates, also benefits from this. Different from multi-rate probes, packet trains are typically used in an iterative procedure that takes advantage of feedback to adapt the rate of the next packet train. Such a procedure is also proposed in [20], where machine learning is used to classify individual packet trains to control a binary search. The goal is to adapt the probe rate until it approaches the available bandwidth. In contrast, we use a feature vector that iteratively includes each additional packet train probe. This additional information enables estimating the available bandwidth directly, without the necessity that the probe rate converges to the available bandwidth. Instead of a binary search, our method chooses the next probe rate that is expected to improve the bandwidth estimate the most.

We evaluate our method in controlled experiments in a network testbed. We specifically target topologies where the assumptions of the deterministic fluid model in Eq. 2.1 are not satisfied such as bursty cross traffic and multiple tight links. For a reference, we implement three state-of-the-art methods, two model-based and one machine learning-based, to use the same data set as our neural network-based approach.

We present a broader evaluation of our method using a wide range of randomly generated topologies with largely varying capacities and cross traffic intensities and burstiness. The results confirm that our method is scale-invariant, i.e., it is applicable without prior calibration to networks of arbitrary capacity. We also train our method for multiple tight links to reduce the bias and variance in estimates. Further, we present a deeper investigation of multi-hop networks. We specifically consider cases where the tight link differs from the bottleneck

link and show how the order in which these occur in a network path affects the input-output relation Eq. (2.1) that is used for available bandwidth estimation.

To compare with state-of-the-art machine learning technique [20], we formulate the task of available bandwidth estimation as a classification problem. We implement two different classifiers named individual classifier and full classifier. The individual classifier represents state-of-the-art and uses the information of a single probe rate to classify whether the current probe rate is above or below the available bandwidth. Depending upon the result, the next probe rate is chosen iteratively until the probe rate converges to the available bandwidth. In contrast, the full classifier, which is implemented by our method, uses a p-dimensional feature vector that includes the information of all the previous probe rates to improve the classification whether the probe rates are above or below the available bandwidth.

Further, we evaluate our proposed method with three other supervised machine learning techniques: SVR, GPR and Bagging. We show that the ability of the neural networks to generalize non-locally makes them favorable for the available bandwidth estimation in arbitrary topologies where the network parameters differ significantly from the training data. We provide data sets generated in a controlled network testbed located at Leibniz University Hannover for training and testing of our proposed method. The work in this chapter is based on joint work with Markus Fidler and Bodo Rosenhahn [40, 41].

The remainder of this chapter is structured as follows: In Sec. 4.1, we discuss the related work on available bandwidth estimation using machine learning. In Sec. 4.2, we introduce the reference implementation of model-based estimation techniques. We present our neural network-based method, describe the training and show testing results in Sec. 4.3. In Sec. 4.4, we consider the estimation of available bandwidth and capacity for randomly generated networks with different tight link capacities, and networks with multiple tight links. Our iterative neural network-based method that selects the probe rates itself as well as a comparison with other machine learning techniques are presented in Sec. 4.5.

4.1 RELATED WORK

We have discussed the model-based state-of-the-art bandwidth estimation techniques in chapter 2. In this section, we will discuss the related work in bandwidth estimation employing machine learning.

The machine learning approach has been considered earlier in [38, 42, 43] and receives increasing attention in recent research [20, 39]. The works differ from each other with respect to their application: [42] considers the prediction of the available bandwidth from packet data traces that have been obtained

in passive measurements. In contrast, [20, 38, 39, 43] use active probes to estimate the available bandwidth in Network Simulator (NS)-2 simulations [38], the European Traffic Observatory Measurement InfrastruCture (ETOMIC) network [43], ultra-high speed 10 Gbps networks [20] and operational Long-Term Evolution (LTE) networks [39], respectively.

Common to these active probing methods [20, 38, 39, 43] is the use of packet chirps [27] that are probes of several packets sent at an increasing data rate. The rate increase is achieved either by a (geometric) reduction of the input gap [38, 43], by concatenating several packet trains with increasing rates to a multi-rate probe [20], or by a linear increase of the packet size [39]. Chirps permit detecting the turning point of Eq. (2.2), which coincides with the available bandwidth, using a single probe. They are, however, susceptible to random fluctuations [23].

Other than chirps, [38] evaluates packet bursts that are probes of back-to-back packets and concludes that bursts are not adequate to estimate the available bandwidth. Also, [20] considers constant rate packet trains for an iterative search for the available bandwidth. Here, machine learning solves a classification problem to estimate whether the rate of a packet train exceeds the available bandwidth or not. Depending on the result, the rate of the next packet train is reduced or increased in a binary search as in [30] until the probe rate approaches the available bandwidth. The authors of [20] give, however, preference to chirp probes.

The feature vectors that are used for machine learning are generally measurements of g_{out} [38, 39, 43] with the exception that [20] uses the Fourier transform of vectors of g_{in} and g_{out}. Supervised learning is used and [20, 39] take advantage of today's availability of different software packages to compare the utility of state-of-the-art machine learning techniques in bandwidth estimation.

4.2 MODEL-BASED REFERENCE IMPLEMENTATIONS

As discussed in chapter 2, the methods for available bandwidth estimation that are based on the fluid model of Eq. (2.1) essentially fall into two different categories: *iterative probing* and *direct probing*. For each of the two categories, we implement a bandwidth estimation technique that is representative of state-of-the-art. While available bandwidth estimation tools differ significantly regarding the selection and the amount of probe traffic, our implementations are tailored to use the same database so that they provide a reference for the neural network-based method.

4.2.1 *Iterative probing*

To implement the iterative probing technique, we use a data set of equidistantly spaced r_{in} and the corresponding r_{out} in our experiments. We process these entries iteratively in increasing order of r_{in} and apply the threshold test of IGI/PTR [31] $(r_{in} - r_{out})/r_{in} > \Delta^{th}$, where the threshold value Δ^{th} is set to 0.1, to determine whether r_{in} exceeds the available bandwidth. We denote r_{in}^{th} the largest rate before the test detects that the available bandwidth is exceeded for the first time and report r_{in}^{th} as the available bandwidth estimate. We note that there may, however, exist $r_{in} > r_{in}^{th}$, where the test fails again. This may occur, for example, due to the burstiness of the cross traffic that causes fluctuations of the available bandwidth.

4.2.2 *Direct probing*

To implement the direct probing technique, we combine it with a threshold test to select relevant probe rates. Direct probing techniques require that $r_{in} > C - \lambda$ where C and λ are unknown. We adapt a criterion from DietTOPP [33] to determine a minimum threshold r_{in}^{min} that satisfies $r_{in}^{min} > C - \lambda$ and use only the probe rates $r_{in} \geqslant r_{in}^{min}$.

The procedure to find r_{in}^{min} is as follows: We use the maximal input rate in the measurement data denoted by r_{in}^{max} and extract the corresponding output rate r_{out}^{max}. If $r_{in}^{max} > r_{out}^{max}$, it can be seen from Eq. (2.7) that both $r_{in}^{max} > C - \lambda$ as well as $r_{out}^{max} > C - \lambda$. Hence, we use $r_{in}^{min} = r_{out}^{max}$ as a threshold to filter out all $r_{in} \leqslant r_{in}^{min}$.

If the assumptions of the fluid model do not hold, e.g., in the case of random cross traffic, the regression technique may occasionally fail. We filter out bandwidth estimates that can be classified as infeasible. This is the case if the slope of the regression line is so small that the intersection with 1 is on the negative r_{in} axis in Fig. 2.5, implying the contradiction $A < 0$, or if the slope of the regression line is negative, implying $C < 0$.

4.3 NEURAL NETWORK-BASED METHOD

In this section, we present our neural network-based implementation of bandwidth estimation, describe the training data sets, and show a comparison of available bandwidth estimates for a range of different network parameters.

4.3.1 *Scale-invariant Implementation*

We use a neural network that takes a p-dimensional vector of values r_{in}/r_{out} as input. The corresponding r_{in} are equidistantly spaced with an increment δ_r. Hence, r_{in} is in $[\delta_r, 2\delta_r, \ldots, p\delta_r]$ that is fully defined by the parameters p and δ_r that determine the measurement resolution. Since the actual values of r_{in} do not provide additional information, they are not input to the neural network. Instead, the neural network refers to values of r_{in}/r_{out} only by their index $i \in [1, p]$. The output of the neural network is the tuple of bottleneck capacity and available bandwidth that are also normalized with respect to δ_r, i.e., we use C/δ_r and A/δ_r, respectively. While C/δ_r and A/δ_r are not necessarily integers, they can be thought of as the index i_C and i_A where r_{in} saturates the bottleneck capacity or the available bandwidth, respectively. To obtain the actual capacity and the available bandwidth, the output of the neural network has to be multiplied by δ_r.

Since the capacity of the network is a priori unknown, an adequate measurement resolution δ_r has to be estimated first. For this purpose, we use a packet train probe consisting of n packets that are sent as a burst, i.e., at the maximum possible rate denoted as r_{in}^{δ}, and measure the corresponding output rate r_{out}^{δ} at the receiver. Sending the packet train as a burst ensures that $r_{in}^{\delta} > C - \lambda$ so that we have $r_{out}^{\delta} = r_{in}^{\delta} C/(r_{in}^{\delta} + \lambda)$ from Eq. (2.7). Further, $r_{out}^{\delta} \in [C - \lambda, C]$ if $r_{in}^{\delta} \to C - \lambda$ or $r_{in}^{\delta} \to \infty$, respectively. Hence, r_{out}^{δ} provides a useful estimate of the probing range for bandwidth estimation. Since our feature vector is p-dimensional, we divide $\delta_r = r_{out}^{\delta}/p$ to determine the set of probing rates $r_{in} \in [\delta_r, 2\delta_r, \ldots, p\delta_r]$. In practice, we use a smoothed estimate $\bar{r}_{out}^{\delta}$ that is obtained as the average output rate of a number of repeated probes.

The normalization by δ_r achieves a neural network that is scale-invariant, since the division by δ_r replaces the units, e.g., Mbps or Gbps, by indices. Considering the fluid model in Eq. (2.7), the normalization of all quantities r_{in}, r_{out}, C, and λ by δ_r results in

$$\frac{r_{in}}{r_{out}} = \begin{cases} 1 & \text{if } i \leqslant i_C - i_\lambda, \\ \frac{i + i_\lambda}{i_C} & \text{if } i > i_C - i_\lambda. \end{cases} \tag{4.1}$$

where we used the indices $i = r_{in}/\delta_r$, $i_C = C/\delta_r$, $i_\lambda = \lambda/\delta_r$, and $i_A = A/\delta_r = i_C - i_\lambda$. Eq. (4.1) confirms that the shape of r_{in}/r_{out} is independent of the scale, e.g., sampling a 100 Mbps network in increments of $\delta_r = 10$ Mbps or a 1 Gbps network in increments of $\delta_r = 0.1$ Gbps reveals the same characteristic shape. The advantage of the scale-invariant representation is that the neural network requires less training and is applicable to networks of arbitrary capacity. We

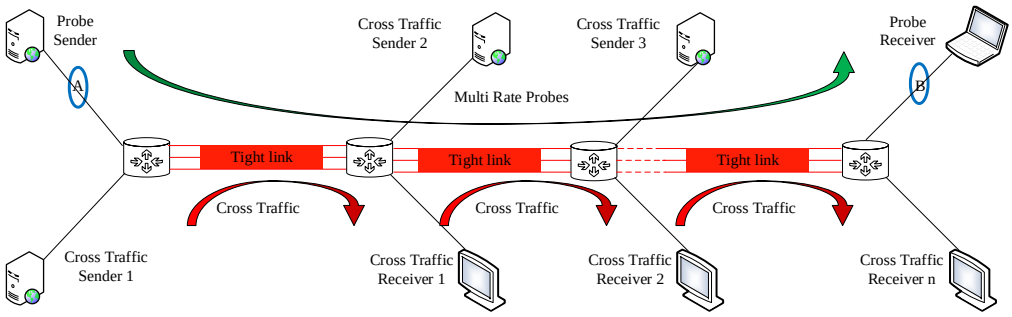


Figure 4.1: Dumbbell topology set up using the Emulab and MoonGen software. A varying number of tight links with single hop-persistent cross traffic are configured. Probe-traffic is path-persistent to estimate the end-to-end available bandwidth from measurements at points A and B.

note that the identity Eq. (4.1) is derived under the assumptions of the fluid model and does not consider effects that are not scale-invariant such as the impact of the packet size or interrupt coalescing.

For implementation, we use a $p = 20$-dimensional input vector of equidistantly sampled values of r_{in}/r_{out}. We decided for a shallow neural network consisting of one hidden layer with 40 neurons. Thus, the network comprises a 20-dimensional input vector, 40 hidden neurons and two output neurons. The output neutrons encode C/δ_r and A/δ_r.

4.3.2 *Training Data: Exponential Cross Traffic, Single Tight Link*

We generate different data sets for training and evaluation using a controlled network testbed. The testbed is located at Leibniz University Hannover and comprises about 80 machines that are each connected by a minimum of 4 Ethernet links of 1 Gbps and 10 Gbps capacity via Virtual Local Area Network (VLAN) switches. The testbed is managed by the Emulab software [44] that configures the machines as hosts and routers, and connects them using VLANs to implement the desired topology. We use a dumbbell topology with multiple tight links as shown in Fig. 4.1. To emulate the characteristics of the links, such as capacity, delay, and packet loss, additional machines are employed by Emulab. We use the MoonGen software [45] for emulation of link capacities that differ from the native Ethernet capacity. To achieve an accurate spacing of packets that matches the emulated capacity, MoonGen fills the gaps between packets by dummy frames that are discarded at the output of the link. We use the *forward rate Lua script* for the MoonGen API to achieve the desired forwarding rate for the transmission and reception ports of MoonGen. Cross traffic of different types and intensities is generated using Distributed Internet Traffic Generator (D-ITG) [46]. The cross traffic is *single hop-persistent*, i.e., at

each link, fresh cross traffic is multiplexed. The probe traffic is *path-persistent*, i.e., it travels along the entire network path, to estimate the end-to-end available bandwidth. We use Real-time UDP Data Emitter (RUDE) & Collector for RUDE (CRUDE) [47] to generate User Datagram Protocol (UDP) probe streams. A probe stream consists of a series of p packet trains of n packets each. The p packet trains correspond to p different probe rates with a constant rate increment of δ_r between successive trains. The packet size of the probe traffic and the cross traffic is $l = 1514$ B including the Ethernet header.

Packet timestamps at the probe sender and receiver are generated at points A and B, respectively, using libpcap at the hosts. We also use a specific Endace Data Acquisition and Generation (DAG) measurement card to obtain accurate reference timestamps. The timestamps are used to compute r_{in} and r_{out} for each packet train.

We generate two training data sets for a single tight link with exponential cross traffic. In data set (i) the capacity of the tight link is $C = 100$ Mbps. Exponential cross traffic with an average rate $\lambda = 25$, 50 and 75 Mbps is used to generate different available bandwidths. In data set (ii) the capacity of the tight link is set to $C = 50$ Mbps and the exponential cross traffic has an average rate of $\lambda = 12.5$, 25 and 37.5 Mbps, respectively. In both cases, the capacity of the access links is $C = 1$ Gbps. The probe streams comprise packet trains of $n = 100$ packets sent at $p = 20$ different rates with rate increment δ_r, where δ_r is determined as described in Sec. 4.3.1. For each configuration, 100 repeated experiments are performed.

For training of the neural network, we first implement an autoencoder for each layer separately and then fine-tune the network using Scaled Conjugate Gradient (SCG). Given a regression network, we optimize the L2-error requiring approximately 1000 epochs until convergence is achieved. Training of the network (using Matlab) takes approximately 30 s. Due to the limited amount of training data (600 experiments overall in both training data sets), the shallow network with a small number of hidden neurons allows training without much over-fitting.

4.3.3 *Evaluation: Exponential Cross Traffic, Single Tight Link*

We train the neural network using the two training data sets and generate additional data sets for testing. The test data is generated for the same network configuration as the (i) training data set, i.e., using exponential cross traffic of 25, 50 and 75 Mbps at a single tight link of 100 Mbps capacity. We also consider other cross traffic rates of 12.5, 37.5, 62.5 and 87.5 Mbps that have not been included in the (i) training data set to see how well the neural network interpo-

lates and extrapolates. We repeat each experiment 100 times so that we obtain 100 bandwidth estimates for each configuration. We compare the performance of the neural network-based method with the two model-based reference implementations of an iterative and a direct estimation method. All three methods generate available bandwidth estimates from the same measurement data.

4.3.3.1 *Testing*

The testing results of the neural network-based method are summarized in Fig. 4.2a compared to the results of the direct and the iterative method. We show the average of the available bandwidth estimates with error bars that depict the Standard Deviation (SD) of the estimates. The variability of the available bandwidth estimates is due to a number of reasons, as discussed in chapter 1. Particularly, the exponential cross traffic deviates from the fluid model and causes random fluctuations of the measurements of r_{out}.

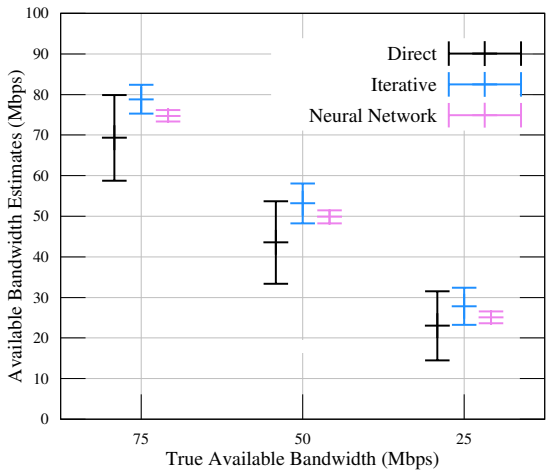
The variability of the available bandwidth estimates of the direct method is comparably large, and the average underestimates the true available bandwidth. The iterative method shows less variability but tends to overestimate the available bandwidth. This is a consequence of the threshold test, where a lower threshold increases the responsiveness of the test but makes it more sensitive to random fluctuations. The neural network-based method improves bandwidth estimates significantly. The average matches the true available bandwidth and the variability is low. The good performance of the neural network is not unexpected as it has been trained for the same network parameters.

4.3.3.2 *Interpolation*

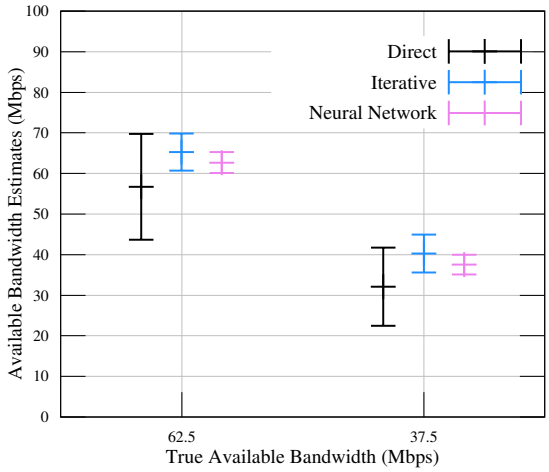
Next, we consider cross traffic of the same type, i.e., exponential, however, with a different rate that has not been included in the training data. First, we consider cross traffic rates of 37.5 and 62.5 Mbps that fall into the range of rates 25, 50 and 75 Mbps that have been used for training, hence the neural network has to interpolate. The results in Fig. 4.2b show that the available bandwidth estimates of the neural network-based method are consistent in this case too.

4.3.3.3 *Extrapolation*

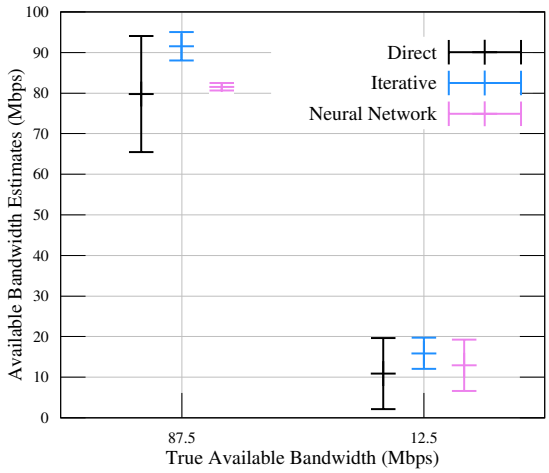
Fig. 4.2c depicts available bandwidth estimates for cross traffic rates of 12.5 and 87.5 Mbps. These rates fall outside the range of rates that have been included in the training data set so that the neural network has to extrapolate. The results of the neural network-based method are nevertheless highly accurate, with a noticeable underestimation of 5 Mbps on average only in the case of a true available bandwidth of 87.5 Mbps. A reason for the lower accuracy that is



(a) Testing



(b) Interpolation



(c) Extrapolation

Figure 4.2: Bandwidth estimates for different cross traffic rates that have been included in the training data set (testing), that fall into the range of the training data set (interpolation), and that fall outside the range of the training data set (extrapolation). The neural network-based method provides available bandwidth estimates that exhibit little variation and have an average that matches the true available bandwidth.

observed when the available bandwidth approaches the capacity is that fewer measurements are on the characteristic upward line segment, see Fig. 2.5 that is also used for estimation by the direct method. The higher intensity of the cross traffic increases the variation in the available bandwidth estimates as can be seen in the case of a true available bandwidth of 12.5 Mbps.

4.3.4 *Network Parameter Variation Beyond the Training Data*

We investigate the sensitivity of the neural network with respect to a variation of network parameters that differ substantially from the training data set. Specifically, we consider two cases that are known to be hard in bandwidth estimation. These are cross traffic with high burstiness, and networks with multiple tight links. In this section, we evaluate the variability of cross traffic. Multiple tight links are discussed in the following section.

4.3.4.1 *Burstiness of Cross Traffic*

To evaluate how the neural network-based method performs in the presence of cross traffic with an unknown burstiness, we consider three different types of cross traffic: CBR that has no burstiness as assumed by the probe rate model, moderate burstiness due to exponential packet inter-arrival times, and heavy burstiness due to Pareto inter-arrival times with infinite variance caused by a shape parameter of $\alpha_s = 1.5$. The average rate of cross traffic is $\lambda = 50$ Mbps in all cases. As before, the tight link capacity is $C = 100$ Mbps and the access links capacity is $C = 1$ Gbps.

The burstiness of the cross traffic can cause queueing at the tight link even if the probe rate is below the average available bandwidth, i.e., if $r_{in} < C - \lambda$. This effect is not captured by the fluid model. It causes a deviation from the ideal rate response curve as depicted in Fig. 2.5 that is maximal at $C - \lambda$ and blurs the bend that marks the available bandwidth. The result is an increase in the variability of available bandwidth estimates as well as an underestimation bias in both direct and iterative bandwidth estimation techniques [8, 9]. Fig. 4.3 shows the mean and the SD of 100 repeated experiments using the direct and iterative probing techniques and the neural network-based method. The average of the estimates shows a slight underestimation bias compared to the true available bandwidth if the cross traffic burstiness is increased. More pronounced is the effect of the cross traffic burstiness on the SD of the bandwidth estimates. While for CBR cross traffic, the estimates are close to deterministic, the variability of the estimates increases significantly if the cross traffic is bursty. The neural network, that has been trained for exponential cross traffic only, performs almost perfectly in the case of CBR cross traffic and shows good results

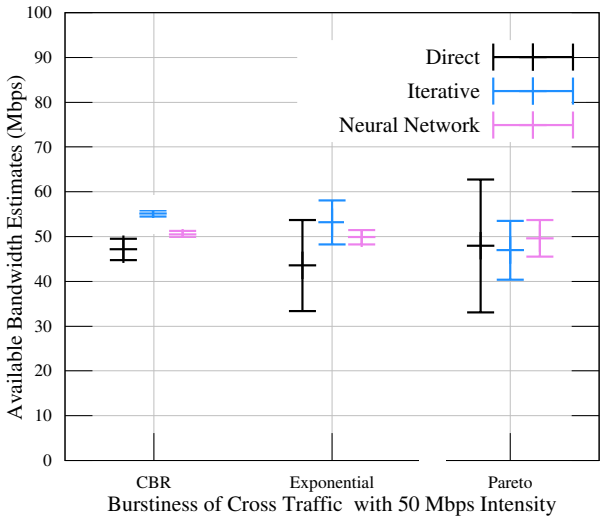


Figure 4.3: Bandwidth estimates for different types of cross traffic burstiness. An increase of the burstiness causes a higher variability of the bandwidth estimates as well as an underestimation bias.

with less variability compared to the direct and iterative techniques also for the case of Pareto cross traffic.

4.4 VARIATION OF THE TIGHT LINK CAPACITY AND MULTIPLE TIGHT LINKS

In this section, we evaluate the performance of our neural network-based method for a wide range of different network scenarios. For example, the capacity of the network may be small, e.g., in access networks, or large, e.g., in backbone networks. Further, these networks may have a high cross traffic intensity with an unknown burstiness and possibly multiple tight links. Hence, the test data may differ substantially from the training data. For the purpose of training, we use data sets (i) and (ii) generated for a single-hop network with tight link capacity $C = 50$ Mbps and $C = 100$ Mbps, respectively, in the presence of exponential cross traffic, i.e., moderate burstiness. For testing, we use data sets that have much more variation than the training data. The testing data is obtained using randomly generated networks that cover a wide range of capacities and cross traffic intensities with unknown burstiness. We also address the known underestimation bias in the case of multiple tight links and include networks where the tight link does not coincide with the bottleneck link in our evaluation.

4.4.1 *Random Networks*

Since our implementation of the neural network-based method is scale-invariant (within the limits of the fluid model), we expect that the method can perform

bandwidth estimation in arbitrary networks. To investigate the sensitivity of our method with respect to different network parameters, we set up a number of topologies where the tight link capacity and the cross traffic intensity are chosen randomly. The capacity is a random number in the interval [10 Mbps, 1 Gbps] with exponential distribution, i.e., $C = 10^{U[1,3]}$ Mbps where $U[1,3]$ is a uniform random variable in the interval $[1,3]$. The cross traffic intensity is chosen relative to the tight link capacity as $\lambda = U[0,1] \cdot C$ Mbps. The link and traffic characteristics are emulated as described in Sec. 4.3.2. The access link capacity for all networks is $C = 1$ Gbps. Since the capacity is unknown to the estimation method, the probing increment δ_r is chosen after sending initial packet train probes as a burst as described in Sec. 4.3.1.

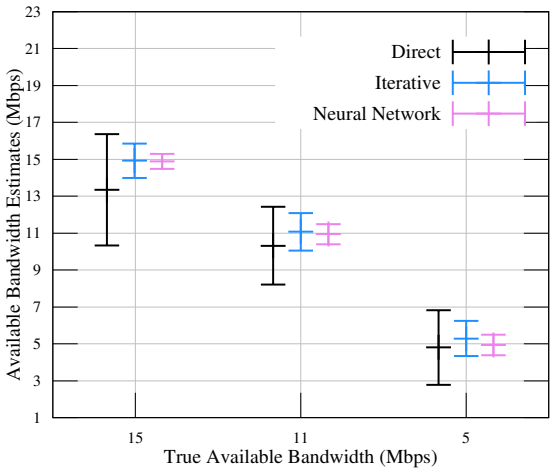
4.4.1.1 *Available Bandwidth Estimates*

We perform a number of experiments using randomly generated networks with a wide variety in the tight link capacity, cross traffic intensity, and cross traffic burstiness. We consider small, moderate, and large values of these network parameters to evaluate the performance of our proposed method. First, we consider different intensities of exponential cross traffic and second, different cross traffic burstiness: CBR (least or without), exponential (moderate) and Pareto (heavy) in random networks.

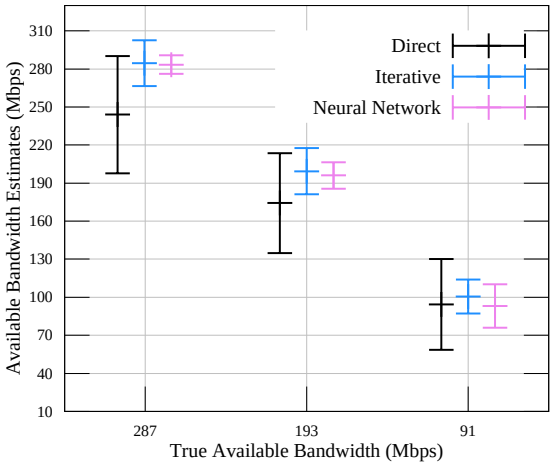
We show some selected results which are representative of the performance of our approach. Fig. 4.4a, Fig. 4.4b and Fig. 4.4c show the results for random networks with a tight link capacity of $C = 22$ Mbps (small), $C = 385$ Mbps (medium), and $C = 831$ Mbps (large). Further, corresponding to each tight link capacity, the cross traffic intensities also vary from low to high. For example, Fig. 4.4a shows the results for the data obtained for a network with a tight link capacity $C = 22$ Mbps in the presence of exponential cross traffic with intensity $\lambda = 7$ Mbps (low), $\lambda = 11$ Mbps (medium), and $\lambda = 17$ Mbps (high). Similarly, for Fig. 4.4b and Fig. 4.4c the data is obtained for networks with capacity $C = 385$ Mbps and $C = 831$ Mbps, and the corresponding exponential cross traffic of intensities 98, 192, 294 Mbps and 250, 439, 621 Mbps, respectively.

Fig. 4.5a, Fig. 4.5b and Fig. 4.5c show results in the presence of cross traffic with different types of burstiness for different randomly generated networks. The data is obtained for networks with a tight link capacity of $C = 61$ Mbps (small), $C = 276$ Mbps (medium), and $C = 1$ Gbps (large) with an average rate of the cross traffic of 46, 131, and 493 Mbps, respectively.

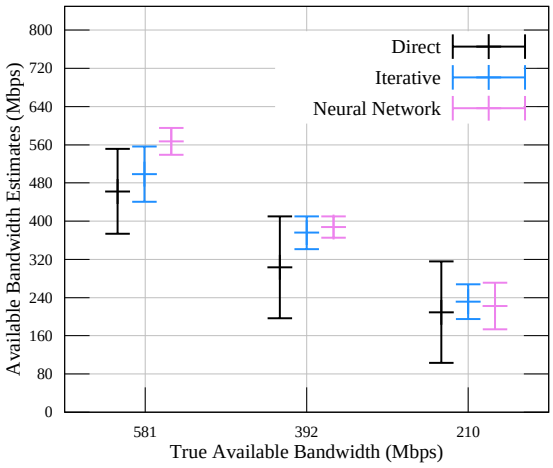
The results confirm that our method is able to estimate the available bandwidth in networks with arbitrary capacities and in the presence of different intensities and unknown burstiness of the cross traffic. The average of the avail-



(a) Bandwidth estimates in C = 22 Mbps

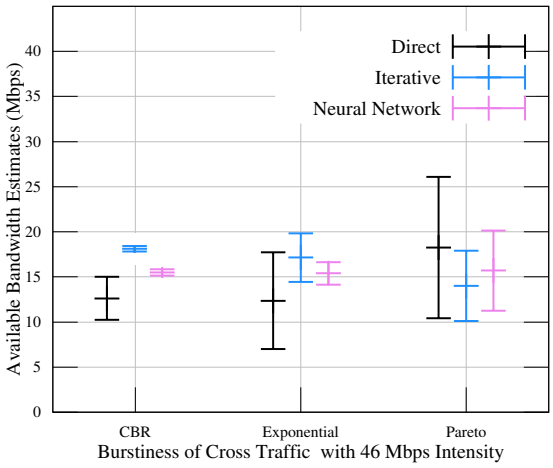


(b) Bandwidth estimates in C = 385 Mbps

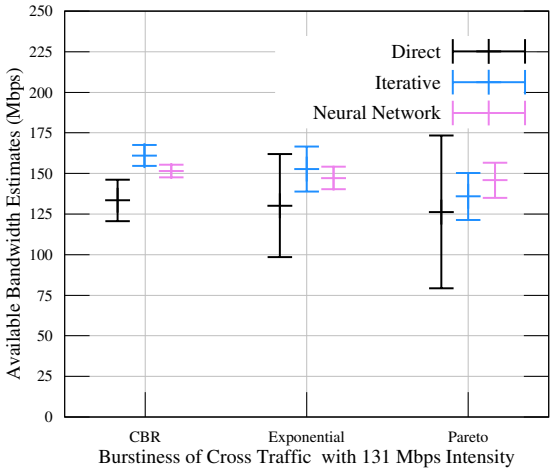


(c) Bandwidth estimates in C = 831 Mbps

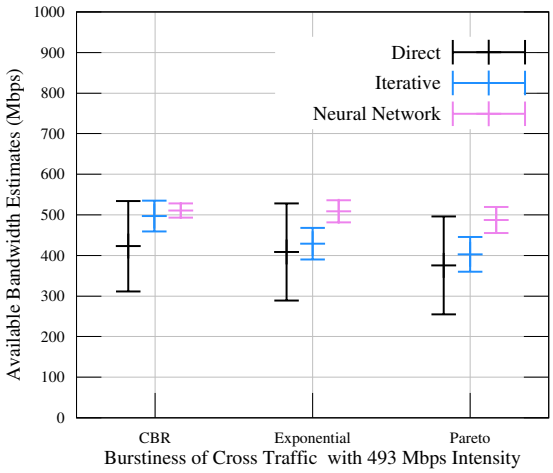
Figure 4.4: Available bandwidth estimates for a wide range of intensities of exponential cross traffic in random networks with a tight link capacity of (a) C = 225 Mbps, (b) C = 383 Mbps, and (c) C = 831 Mbps.



(a) Bandwidth estimates in C = 61 Mbps



(b) Bandwidth estimates in C = 276 Mbps



(c) Bandwidth estimates in C = 1 Gbps

Figure 4.5: Available bandwidth estimates for unknown cross traffic burstiness in random networks with a tight link capacity C and true available bandwidth A: (a) C = 61 Mbps, A = 15 Mbps, (b) C = 276 Mbps, A = 145 Mbps, and C = 1 Gbps, A = 507 Mbps.

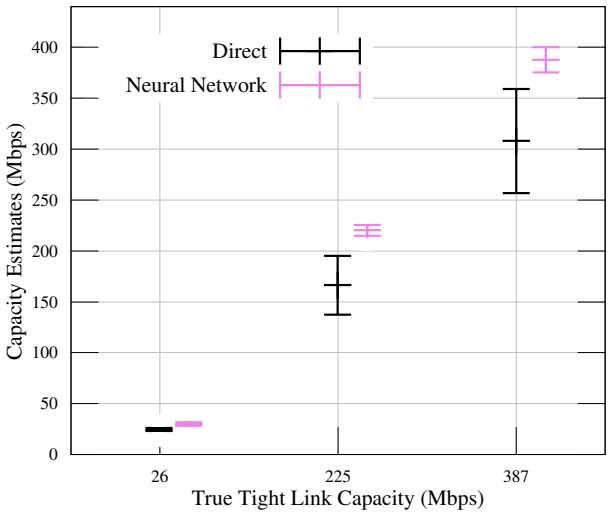


Figure 4.6: Tight link capacity estimates for random networks in the presence of exponential cross traffic with an average intensity of 4, 60, and 98 Mbps.

able bandwidth estimates corresponds to the true available bandwidth, though the variation increases with higher burstiness and intensity of the cross traffic.

4.4.1.2 *Capacity Estimates*

The output of our neural network-based approach is the tuple of bottleneck capacity and available bandwidth estimates. We evaluate our method to estimate the tight link capacity of networks. The measurement data is obtained for random networks with a true tight link capacity of $C = 26$ Mbps, $C = 225$ Mbps, and $C = 387$ Mbps in the presence of exponential cross traffic of intensity 14, 60, and 98 Mbps, respectively.

Fig. 4.6 depicts the capacity estimates obtained by the direct method and the neural network, respectively. Both methods perform well for small capacities. However, in higher capacity networks, the variation of the estimates increases. This is due to the fact that the inter-packet gaps are highly susceptible to noise introduced by bursty cross traffic in these networks. The difficulty to attain precise smaller input gaps g_{in} makes the input data noisy. As described in Sec. 4.2.2, the direct method estimates the capacity from the slope of the upward segment of a rate response curve shown in Fig. 2.5. In the presence of random cross traffic, the slope gets altered. This leads to underestimation and variability in the estimates of the direct method. However, the neural network, that is trained only for small capacities of 50 Mbps and 100 Mbps, is able to perform well even in higher capacity networks. The estimates are accurate with a mean value matching the true capacity value, though the variation of the estimates increases with increasing capacity due to noisy test data. Thus, with our

proposed method, we are able to accurately estimate the available bandwidth as well as the capacity.

4.4.2 *Multiple Tight Links*

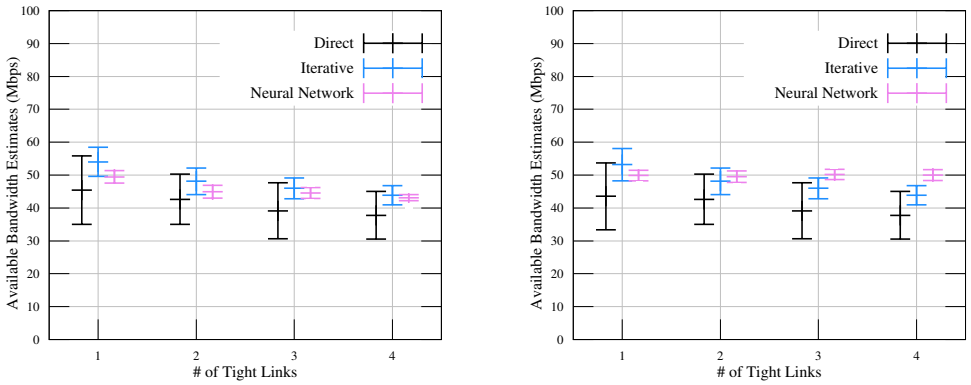
We now address networks with multiple tight links. These networks pose a well-known challenge in available bandwidth estimation. In the case of multiple tight links, the probe stream has a constant rate r_{in} with a defined input gap g_{in} only at the first link. For the following links, the input gaps have a random structure as they are the output gaps of the preceding links. At each additional link, the probe stream interacts with new, bursty cross traffic. This causes lower probe output rates and results in underestimation of the available bandwidth in multi-hop networks [8–10, 12].

To test the neural network with multiple tight links, we extend our network from single-hop to multi-hop as shown in Fig. 4.1. The path-persistent probe streams experience single hop-persistent exponential cross traffic with the average rate $\lambda = 50$ Mbps while traversing multiple tight links of capacity $C = 100$ Mbps. The capacity of the access links is 1 Gbps.

In Fig. 4.7a, we show the results from 100 repeated measurements for networks with 1 up to 4 tight links. The model-based methods, both direct and iterative, as well as the neural network-based method underestimate the available bandwidth with an increasing number of tight links. The reason for underestimation in the case of model-based techniques is the cross traffic burstiness which potentially reduces the output probe rate at each of the tight links. Though the estimates of our neural network show the least variability, the neural network underestimates the available bandwidth. This is not surprising, as it is trained only for a single tight link.

4.4.2.1 *Training for Multiple Tight Links*

Since the neural network that is trained only for a single tight link, underestimates the available bandwidth in the case of multiple tight links, we consider training the neural network for the latter. A neural network has also been trained for multiple tight links in [43], using, however, packet chirp probes. We extend the single tight link network up to four tight links as shown in Fig. 4.1. To generate the training data, a probe stream consisting of p packet trains is sent through a network with two, three, and four tight links, respectively, each with exponential cross traffic with an average rate $\lambda = 50$ Mbps. In all networks, the probe stream is path persistent and the cross traffic is single hop-persistent. The capacity of the access links is 1 Gbps and that of the tight links is 100 Mbps.



(a) Bandwidth estimates with the neural network trained for a single tight link

(b) Bandwidth estimates with the neural network trained for multiple tight links

Figure 4.7: Multiple tight links with capacity $C = 100$ Mbps in the presence of single hop-persistent exponential cross traffic with an average rate $\lambda = 50$ Mbps. (a) All methods tend to underestimate the available bandwidth in the case of multiple tight links. (b) The training of the neural network for multiple tight links improves the bandwidth estimates significantly.

The neural network is trained with this additional (iii) training set for multiple tight links along with training sets (i) and (ii) for a single tight link.

Fig. 4.7b compares the results of direct and iterative methods with the neural network. As can be seen clearly, the results have improved significantly with the neural network after it has been trained for multiple tight links. The mean value matches the true available bandwidth and the estimates have less variability.

4.4.2.2 *Tight Link differs from the Bottleneck Link*

The problem of available bandwidth estimation in multi-hop networks becomes more difficult if the tight link of a network path differs from the bottleneck link. As shown in Fig. 4.8, link i is the *bottleneck link*, i.e., the link with the minimum capacity, and link $i+1$ represents the *tight link*, i.e., the link that has the minimum available bandwidth [37]. The existence of separate tight and bottleneck links has an impact on the shape of the rate response curve. The rate response curve in Fig. 2.5 is derived under the assumption that there is a single tight link that matches the bottleneck link. If this assumption does not hold, available bandwidth estimation by direct methods becomes more difficult. We investigate the impact further by considering two different scenarios. In the first scenario, i.e., scenario I, the bottleneck link appears in front of the tight link. In the second, i.e., scenario II, it is vice versa. In both the scenarios as seen in Fig. 4.8 only the input rate r_{in}^{i} at the first link i has the desired input gap. As the packet train traverses through the first link with capacity C^{i}, it interacts

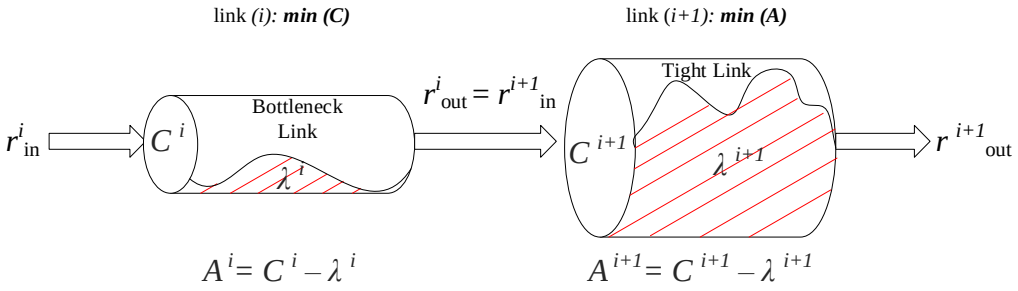


Figure 4.8: A two-hop network where the tight link differs from the bottleneck link.

with the cross traffic with average rate λ^i and the resulting output rate r^i_{out} is obtained by rearranging Eq. (2.7) as

$$r^i_{out} = \begin{cases} r^i_{in} & \text{if } r^i_{in} \leqslant C^i - \lambda^i, \\ \frac{r^i_{in} C^i}{r^i_{in} + \lambda^i} & \text{if } r^i_{in} > C^i - \lambda^i. \end{cases} \tag{4.2}$$

For the next link $i+1$, the input has a different rate as it is fed from the first link i, i.e., $r^{i+1}_{in} = r^i_{out}$. The output rate r^{i+1}_{out} from hop $i+1$ is obtained by recursive insertion of Eq. (4.2) as

$$r^{i+1}_{out} = \begin{cases} r^i_{in} & \text{if } r^i_{in} \leqslant A^{i+1} \ , \ r^i_{in} \leqslant A^i, \\ \frac{r^i_{in} C^{i+1}}{r^i_{in} + \lambda^{i+1}} & \text{if } r^i_{in} > A^{i+1} \ , \ r^i_{in} \leqslant A^i, \\ \frac{r^i_{in} C^i}{r^i_{in} + \lambda^i} & \text{if } \frac{r^i_{in} C^i}{r^i_{in} + \lambda^i} \leqslant A^{i+1} \ , \ r^i_{in} > A^i, \\ \frac{\frac{r^i_{in} C^i}{r^i_{in} + \lambda^i} C^{i+1}}{\frac{r^i_{in} C^i}{r^i_{in} + \lambda^i} + \lambda^{i+1}} & \text{if } \frac{r^i_{in} C^i}{r^i_{in} + \lambda^i} > A^{i+1} \ , \ r^i_{in} > A^i. \end{cases} \tag{4.3}$$

where C^i and C^{i+1} are the link capacities, λ^i and λ^{i+1} are the single hop-persistent cross traffic intensities, and $A^i = C_i - \lambda_i$ and $A^{i+1} = C_{i+1} - \lambda_{i+1}$ are the corresponding available bandwidths for link i and $i+1$, respectively. Eq. (4.3) represents the two scenarios defined above. In the case of scenario I, where link $i+1$ is the tight link, lines 1, 2 and 4 of Eq. (4.3) apply. Conversely, in the scenario II, where link i is the tight link, lines 1, 3 and 4 apply.

Fig. 4.9 shows the rate response curves for the two scenarios obtained using Eq. (4.3). The curves are piece-wise linear. The two bends indicate the presence of two congestible links. For both scenarios, the tight link capacity is $C =$

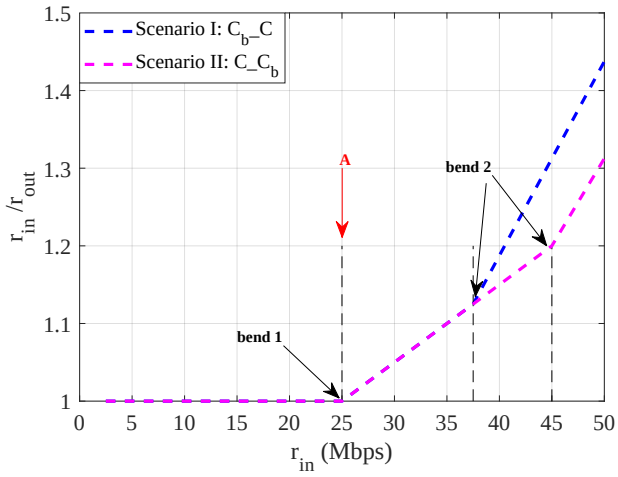


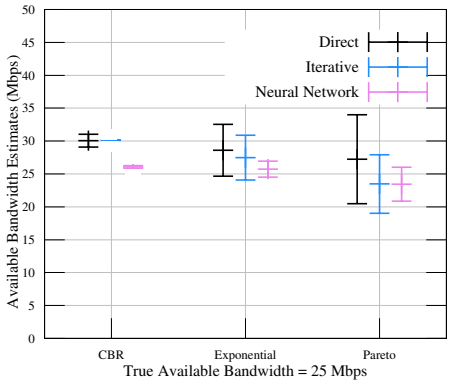
Figure 4.9: Rate response curves of a two-hop network representing scenarios I and II where the tight link occurs behind and in front of the bottleneck link, respectively. The presence of two bends indicates that two links are congested.

100 Mbps and the bottleneck capacity is $C_b = 50$ Mbps. The cross traffic is CBR with an average rate $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps, traversing the tight link and the bottleneck link, respectively.

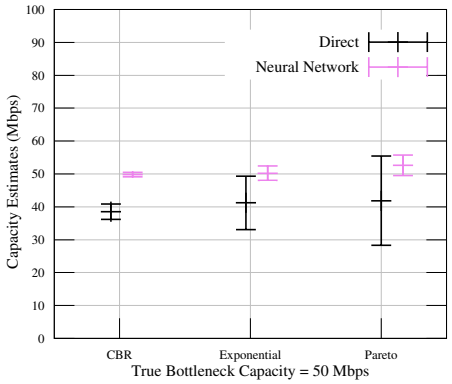
Considering Fig. 4.9, we have $r_{out}^{i+1} = r_{in}^{i}$ until the probe rate reaches the first bend at 25 Mbps, where the available bandwidth of the tight link is saturated. In the scenario I, when we increase the probe rate further, we reach a second bend at 37.5 Mbps where the available bandwidth of the bottleneck link, i.e., link i, is also saturated. However, if the order of the two-hop network is reversed, i.e., the bottleneck link succeeds the tight link, the rate response curve differs as represented by scenario II. The reason is that the output of the tight link, i.e., link i in this scenario, is shaped so that r_{out}^{i} exceeds 37.5 Mbps, that is the available bandwidth of the bottleneck link, i.e., link $i+1$, only if r_{in}^{i} exceeds 45 Mbps.

In both scenarios, the second linear segment of the rate response curve can cause overestimation of the available bandwidth if a direct probing method is used. This issue is addressed in [29] by performing a linear regression to each of the linear segments under the assumption that the tight link precedes the bottleneck link in the network. In practical scenarios, however, the order of tight and bottleneck links is a priori unknown. Iterative probing techniques, on the other hand, can still estimate the available bandwidth, even if the tight link and the bottleneck link differ by searching for the first turning point of the rate response curve, i.e., by sending repeated probes at increasing rates, as long as $r_{out}^{i+1} = r_{in}^{i}$.

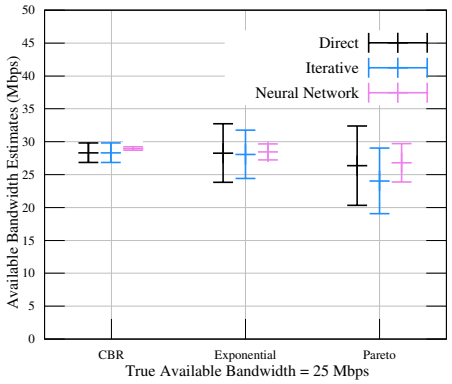
To test the performance of our neural network-based method, we extend our network to a two-hop network, where the tight link and the bottleneck link do



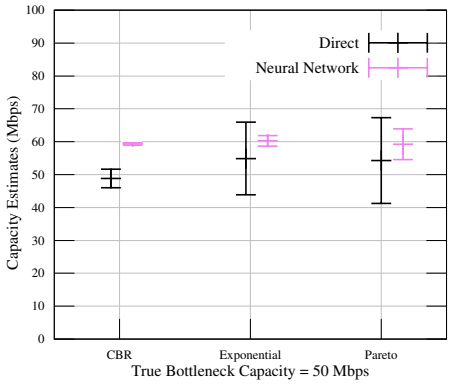
(a) Bandwidth estimates in scenario I trained for a single tight link



(b) Capacity estimates in scenario I trained for a single tight link

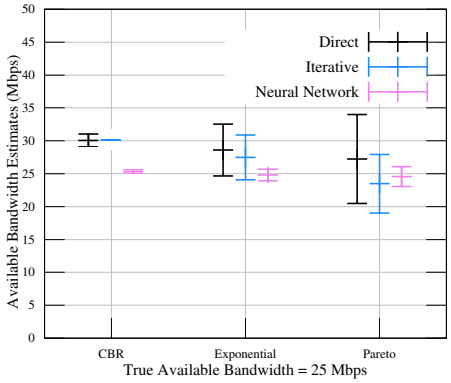


(c) Bandwidth estimates in scenario II trained for a single tight link

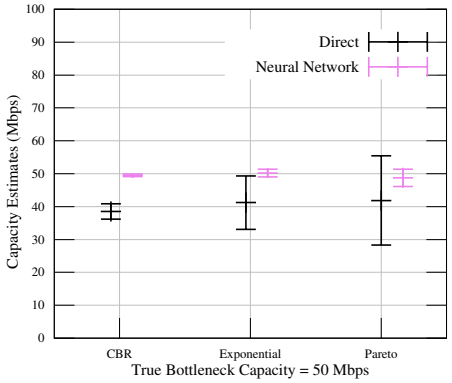


(d) Capacity estimates in scenario II trained for a single tight link

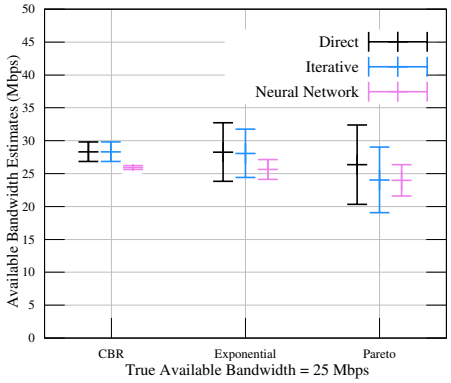
Figure 4.10: Estimates from the neural network trained for a single tight link: (a) available bandwidth and (b) capacity estimates of a two-hop network, where the tight link follows the bottleneck link and, (c) available bandwidth and (d) capacity estimates, where the tight link precedes the latter. In both scenarios, the tight link capacity is $C = 100$ Mbps and the bottleneck capacity is $C_b = 50$ Mbps with cross traffic intensity of $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps respectively.



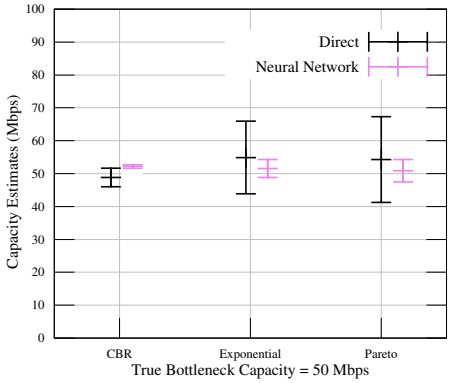
(a) Bandwidth estimates in scenario I trained for the tight link different from the bottleneck link



(b) Capacity estimates in scenario I trained for the tight link different from the bottleneck link



(c) Bandwidth estimates in scenario II trained for the tight link different from the bottleneck link



(d) Capacity estimates in scenario II trained for the tight link different from the bottleneck link

Figure 4.11: Estimates from the neural network trained for different tight link and bottleneck link: (a) available bandwidth and (b) capacity estimates of a two-hop network, where the tight link follows the bottleneck link and, (c) available bandwidth and (d) capacity estimates, where the tight link precedes the latter. In both scenarios, the tight link capacity is $C = 100$ Mbps and the bottleneck capacity is $C_b = 50$ Mbps with cross traffic intensity of $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps, respectively.

not coincide. The access link capacity is $C = 1$ Gbps. We generate test data for the two scenarios described above. Further, we also consider bursty cross traffic along with CBR.

Fig. 4.10a and Fig. 4.10c show the available bandwidth and Fig. 4.10b and Fig. 4.10d the bottleneck capacity estimates in the scenarios I and II, respectively. The neural network is trained for a single tight link only, using the training data set (i) and (ii). The results obtained from the neural network are more consistent as compared to direct and iterative probing methods. However, a bias can be noticed in the available bandwidth and the bottleneck capacity estimates, specifically for scenario II in Fig. 4.10c and Fig. 4.10d, i.e., where the tight link precedes the bottleneck link. This can be explained by the deviation of the output rate r_{out}^{i+1} as obtained from Eq. (4.3) compared to Eq. (4.2) which affects the input feature vector of r_{in}/r_{out} of the neural network.

To reduce the bias in the estimation, we consider training the neural network for networks where the tight link is different from the bottleneck link. An additional training set (iv) is generated in a two-hop network for both scenarios: I and II, where the tight link follows the bottleneck link and precedes the latter, respectively.

Fig. 4.11 shows the available bandwidth and the bottleneck capacity estimates that are obtained after additional training for the scenarios I and II. As can be seen in Fig. 4.10d, the bias of the capacity estimates is reduced significantly.

4.5 ITERATIVE NEURAL NETWORK-BASED METHOD

State-of-the-art iterative probing methods perform a search for the available bandwidth by varying the probe rate r_{in} until r_{in} converges to the available bandwidth. Pathload [30] uses statistical tests to determine whether r_{in} exceeds the available bandwidth or not and performs a binary search to adapt r_{in} iteratively. The recent method [20] adopts Pathload's binary search algorithm but uses machine learning instead of statistical tests to determine whether r_{in} exceeds the available bandwidth or not. Our proposed iterative neural network-based method differs from [20] in several respects.

We propose an iterative neural network-based method that (a) determines the next probe rate by a neural network, that is trained to select the probe rate that improves the bandwidth estimate most, instead of using the binary search algorithm, and (b) it includes the information of all previous probe rates to estimate the available bandwidth instead of considering only the current probe rate. Our implementation comprises two parts. First, we train the neural network to cope up with input vectors that are not fully populated. Second,

we create another neural network that recommends the most beneficial probe rates.

4.5.1 *Partly Populated Input Vectors*

An iterative method will only use a limited set of probe rates. Correspondingly, we mark the entries of the input vector that have not been measured as invalid by setting $r_{in}/r_{out} = 0$. To obtain a neural network that can deal with such partly populated input vectors, we perform training using the training data sets (i) and (ii), where we repeatedly erase a random number of entries at random positions. While testing the neural network, we erase entries in the same way.

In Fig. 4.12, we show the absolute error of the available bandwidth estimates that are obtained by the neural network if $m \in [1, p]$ randomly selected entries of the p-dimensional input vector are given. The bars show the average error and the SD of the error. The data set used for testing is the same as the one used for Fig. 4.2a previously, i.e., $C = 100$ Mbps and $A \in \{25, 50, 75\}$ Mbps. We show the combined results for all values of A.

The average error shows a clear improvement with increasing m. For $m = 1$, the information is not sufficient to identify the two unknown parameters capacity and available bandwidth. Hence, the neural network first reports conservative estimates in the middle range. For comparison, by guessing 50 Mbps in all cases the average error is 16.6 Mbps for the given test data set. With increasing m, the neural network starts to distinguish the range of $A \in \{25, 50, 75\}$ Mbps but tends to frequent misclassifications that can cause large errors. These misclassifications are mostly resolved when increasing m further.

We observe the same trend also for the error of the capacity estimates that shows a high correlation with the error of the available bandwidth estimates. Hence, we omit to show the results.

4.5.2 *Recommender Network for Probe Rate Selection*

When adding entries to the partly populated input vector of the neural network, the average estimation error improves. The amount of the improvement depends, however, on the position of the a priori unknown entry that is added, as well as on the m entries that are already given, i.e., their position and value. We use a second neural network that learns this interrelation. Using this knowledge, the neural network acts as a recommender that given a partly populated input vector selects the next probe rate, i.e., the next entry that is expected to improve the accuracy of the bandwidth estimate the most. The recommender network takes the $p = 20$-dimensional input vector of values r_{in}/r_{out}, has 40 hid-

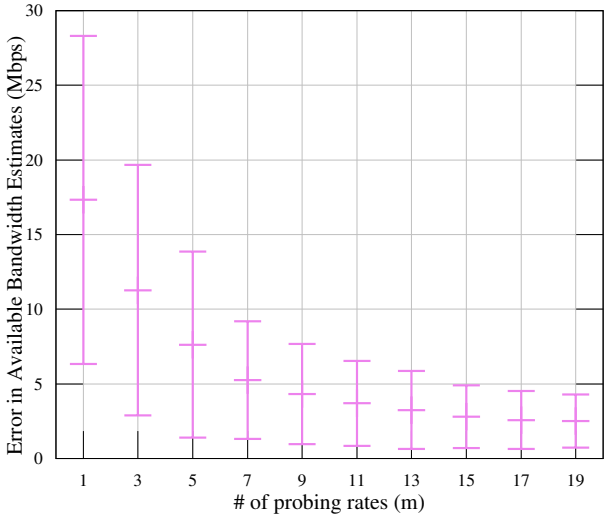


Figure 4.12: Error in the available bandwidth estimates obtained for a set of m randomly selected probe rates.

den neurons, and generates a p-dimensional output vector of estimation errors that apply if the entry r_{in}/r_{out} is added at the respective position. Given the output vector, the rate r_{in} that minimizes the estimation error is selected for probing next.

Fig. 4.13 shows how the recommender network improves the error in the bandwidth estimates compared to the random selection of probe rates shown in Fig. 4.12. Starting at 5 selected probe rates, the average estimation error, as well as the SD of the error, are small, and adding further probe rates improves the estimate only marginally. The reason is that certain probe rates, e.g., those on the horizontal line at $r_{in}/r_{out} = 1$ in Fig. 2.5, provide little additional information. We conclude that the recommender can effectively control the selection of probe rates to avoid those rates that contribute little. In this way, the recommender can save a considerable amount of probe traffic.

4.5.3 *Neural Network as Available Bandwidth Classifier*

Our iterative neural network-based approach, as proposed in Sec. 4.5.2 differs from state-of-the-art machine learning techniques for bandwidth estimation in several respects. For comparison, we formulate available bandwidth estimation as a classification problem. Specifically, to compare our approach with state-of-the-art technique [20], we implement two different classifiers: an individual classifier which represents state-of-the-art and uses the information only from the current probe rate to determine whether the current probe rate is above or below the available bandwidth, and a full classifier which is based on our proposed method and uses a p-dimensional feature vector with information of

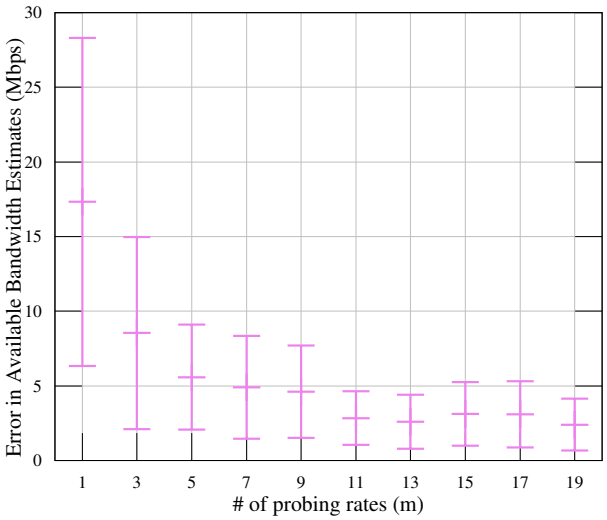


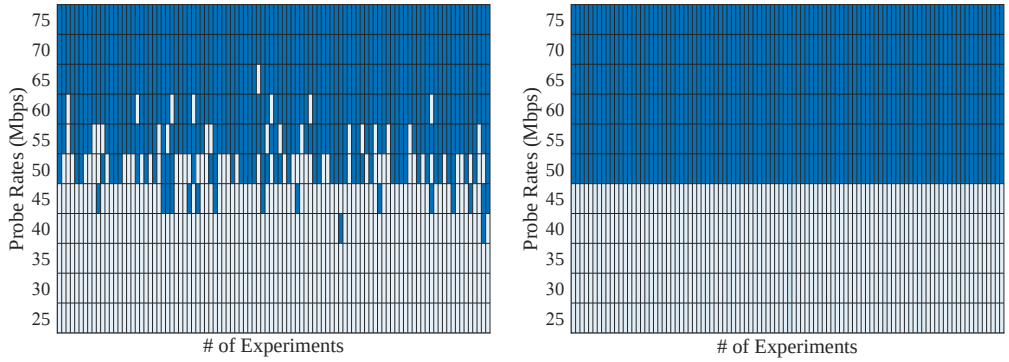
Figure 4.13: Error in the available bandwidth estimates obtained for a set of m recommended probe rates.

all previous probe rates to classify the probe rates. We test both classifiers with the measurement data generated in the presence of bursty cross traffic with an average rate of 50 Mbps. The tight link and the access link capacities are 100 Mbps and 1 Gbps, respectively, as before.

4.5.3.1 *Individual Classifier*

The individual classifier represents the state-of-the-art and uses the information of a single probe rate to classify whether the current rate is above or below the available bandwidth. The goal is to increment the probe rates iteratively by δ_r until the available bandwidth is saturated. For our reference implementation, we decided for a shallow neural network consisting of one hidden layer with 40 neurons. The input feature vector is the ratio of the current probe rate at the sender and at the receiver, i.e., r_{in}/r_{out}. The neural network has one output neuron that represents two different categories: current probe rate *below* or *above* the available bandwidth. Depending upon the result, the next probe rate is increased by δ_r iteratively until the probe rate r_{in} exceeds the available bandwidth.

Fig. 4.14a shows the results of the individual classifier for 100 experiments in the form of a classification map in the probe range $[25, \ldots, 75]$ Mbps. The true available bandwidth is 50 Mbps. The probe rates which are classified as being above the available bandwidth are shown by blue pixels, and those which are classified as below the available bandwidth are represented by white pixels. However, due to the burstiness of the cross traffic, there exist false positive errors, i.e., certain rates below the available bandwidth are misclassified as above. These misclassifications can be seen by the occurrence of blue pixels



(a) Individual Classifier (b) Full Classifier

Figure 4.14: The classification map shows the results of (a) the individual, and (b) the full classifier. The tight link capacity is $C = 100$ Mbps. In the presence of exponential cross traffic with an average rate $\lambda = 50$ Mbps, the true available bandwidth is 50 Mbps. The blue pixels indicate the probe rates that are classified as above the available bandwidth and the white pixels indicate the vice versa.

below 50 Mbps. The precision value is 0.97. Similarly, there are false negatives, i.e., certain rates above the available bandwidth are misclassified as below. This can be seen by white pixels re-occurring above the true available bandwidth of 50 Mbps. This leads to an inconsistency in the classification with the recall value of 0.87. It is not clear from the classification map, which exact probing rate saturates the available bandwidth. This is why Pathload does not report a single available bandwidth estimate but an available bandwidth region [30].

4.5.3.2 *Full Classifier*

To compare with state-of-the-art and motivated by the limitations of the individual classifier, we use our proposed method to implement a full classifier. The full classifier uses a p-dimensional vector of values r_{in}/r_{out} as input, i.e., it does not operate iteratively on individual values of r_{in}/r_{out} but uses all p values of r_{in}/r_{out} at once. The classifier is implemented by a shallow neural network consisting of one hidden layer with 40 neurons. The output of the neural network is a p-dimensional vector consisting of binary values that classify whether the respective probe rate is above or below the available bandwidth.

Fig. 4.14b shows the classification map results for the full classifier. The results are highly accurate without any misclassification with precision and recall values of 1.0, respectively. This can be seen by the occurrence of only blue pixels at probe rates which exceed the true available bandwidth of 50 Mbps. The results confirm the stability of our proposed method in comparison to the individual classifier. The consistency of results comes from the fact that instead of using only the current probe rate, the information of all the previous probe

rates is included. The ability to classify the probe rates directly, instead of doing the binary search as done in [20], increases the computational speed of estimation.

4.5.4 *Evaluation of other Machine Learning Techniques*

Regarding the parameters of our neural network, we also explored the use of deeper networks with more hidden layers, convlayers, and residual networks. However, in our setting different variants of networks did not improve the quality of our experiments. We believe that the main reason is over-fitting, which is caused by the sparse amount of data used for training.

To evaluate our proposed method with other supervised machine learning techniques, we consider three machine learning algorithms that can do a linear regression to estimate the available bandwidth. Two of the chosen techniques are kernel-based 1) SVR 2) GPR, whereas the third one is ensemble-based 3) Bagging. We provide an overview of these techniques and justify their selection for available bandwidth estimation in the section below. Our data set consists of 600 observations: training set (i) and (ii). In order to evaluate our machine learning algorithms, i.e., how accurately they are able to estimate the available bandwidth for unseen test data, we use the K-Fold cross-validation technique. In K-Fold cross-validation technique, the entire data set is split into K subsets or folds, where each subset is used for testing and the remaining $K-1$ subsets are used for training. As more partitions lead to a smaller bias but a higher variance, we set $K=5$ to have a balanced bias-variance trade-off. We used parallel computing in Matlab R2017b on a quad-core Linux-machine, with processor base frequency of 3.20 GHz running Ubuntu 14.04 LTS and kernel of version $4.4.0-130$-generic.

4.5.4.1 *Support Vector Regression*

SVR is a non-parametric kernel-based machine learning technique that performs linear regression for nonlinear functions in the high-dimension feature space using ϵ-insensitive loss. We use the *Gaussian Radial Basis Function (GRBF)* kernel and set epsilon $\epsilon = 0.55$ using the interquartile range of the response variable. In ϵ-SV regression, the goal is to find a function $f(x)$ that has at most a deviation of ϵ from the actually obtained targets for all the training data. The ϵ-insensitive loss function optimizes the generalization bounds given for regression, ignoring the errors as long as they are situated within a certain distance of the true value. We selected SVM as it has a regularization parameter that avoids over-fitting of data and has excellent generalization capability with high prediction accuracy. The computational complexity of SVR does not depend on

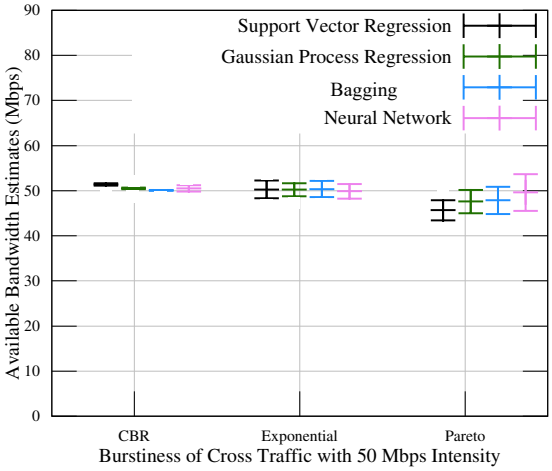
the dimensionality of the input space. The training time is approximately 30 s, with the prediction speed of 7900 obs/s.

4.5.4.2 *Gaussian Process Regression*

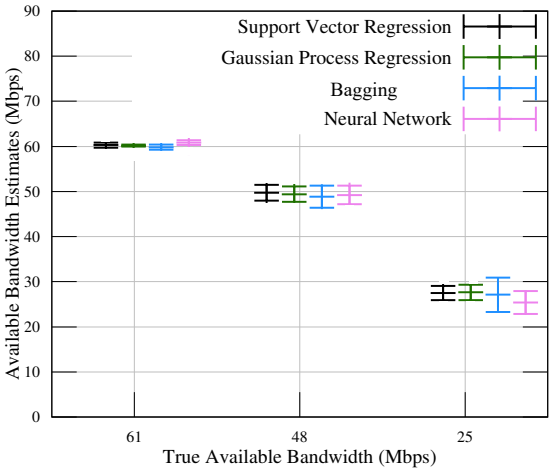
GPR is a non-parametric kernel-based probabilistic machine learning technique. We are interested in making inferences about the relationship between inputs and targets, i.e., the conditional distribution of the targets given the input. In Gaussian processes, the covariance function, which expresses the similarity between the two inputs plays a crucial role [48]. GPR seeks to determine the optimal values of the hyper-parameters governing the covariance function. We use an *exponential* kernel to define the covariance matrix. A prior over functions is defined which is converted into a posterior over functions once we have observation data. The observation data causes the resulting posterior Gaussian Process (GP) samples to be constrained to pass near the observed data points. For example, we have a p-dimensional training feature vector x for which we have observed the function f that estimates the available bandwidth. Now, given a new p-dimensional test vector x_*, we estimate the new function f_* which follows the probability distribution $P(f_*/x_*, x, f)$ where f and f_* have a joint Gaussian distribution. As compared to SVR, GPR provides a probabilistic prediction and an estimate of the uncertainty in the prediction. With GPR using an exponential kernel function, the prediction speed is 11000 obs/s and training time is approximately 38 s.

4.5.4.3 *Bootstrap Aggregation*

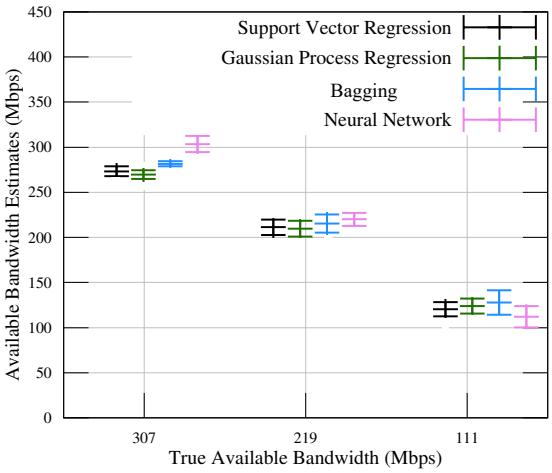
Bagging is an ensemble machine learning approach used for regression, in which *weak learners* collaborate to form *strong learners*. Bootstrap Aggregation, as suggested by its name, uses a bootstrapping technique to sample the input data randomly with replacement to create multiple subsets of data. A set of models is then trained on each of these subsets, and their predictions are aggregated to make the final combined prediction using averaging. The averaging of multiple decision trees reduces variance and bias and avoids over-fitting. The performance of the ensemble technique depends on the setting of the ensemble and of the weak learners. A higher number of trees in the bagging algorithm increases the performance and makes the predictions more stable, but it also slows down the computation, making it ineffective for real-time bandwidth predictions. We chose the hyper-parameters to have a trade-off between the predictive power and the computational speed and chose the number of ensemble learning cycles to be 30. It took approximately 31 s to train with the predicting speed of 2700 obs/s. To minimize the generalization error, the minimum number of leafs that are required to split an internal node is set to 8.



(a) Bandwidth estimates in C = 100 Mbps



(b) Bandwidth estimates in C = 89 Mbps



(c) Bandwidth estimates in C = 417 Mbps

Figure 4.15: Available bandwidth estimates (a) in the presence of bursty cross traffic with the average rate $\lambda = 50$ Mbps and $C = 100$ Mbps, (b) in the random network with the moderate capacity $C = 89$ Mbps, and (c) higher capacity $C = 417$ Mbps in the presence of a wide range of exponential cross traffic. The results for neural networks reduce bias and variance even at higher capacities.

4.5.4.4 *Evaluation*

To evaluate our method, we use the measurement data obtained from the network in the presence of cross traffic of unknown burstiness with intensity $\lambda = 50$ Mbps, and from random networks. The bottleneck capacity is $C = 100$ Mbps in the former network, whereas the capacity and cross traffic are chosen randomly in the latter. The access link capacity for all the networks is $C = 1$ Gbps. As can be seen in Fig. 4.15a, and Fig. 4.15b, our proposed method works using supervised machine learning techniques as well. These techniques are based on the assumption that if two training feature vectors are close, then their corresponding output prediction functions should also be close. With the local generalization principle, they are able to interpolate locally. However, this principle has its limitations. Fig. 4.15c shows the results where these techniques reduce the variance, but a slight bias is visible in bandwidth estimation as compared to the neural network. This is due to the fact that the test data has higher variations as compared to the training data. The measurement data is obtained for a random network with a higher capacity $C = 417$ Mbps where it is difficult to achieve a precise inter-packet gap g_{in}. These distortions in input gaps result in noisy data, i.e., higher variations which are not covered in the training data obtained for a network with bottleneck capacity $C = 50$ Mbps and $C = 100$ Mbps, respectively. However, with the neural networks the mean of 100 iterations matches the true available bandwidth and the estimates have low variability. The neural network can generalize non-locally which kernel or ensemble machines with standard generic kernels are not able to do. It has the ability to recognize complicated functions, even in the presence of noise and variability. We are able to reduce the bias and variance in available bandwidth estimates, even using a shallow neural network. For these reasons, we have chosen the neural network for bandwidth estimation.

5

MULTI-CLASS CLASSIFICATION-BASED ACTIVE AVAILABLE BANDWIDTH ESTIMATION

In this chapter, we propose two different multi-class classification-based methods for online available bandwidth estimation. The first method is based on reinforcement learning and can be used to estimate the available bandwidth in the networks where it is not feasible to create such training data sets that can represent the dynamics of networks completely. Since there is no training phase and the system learns by observing the network path online, the method may take longer to converge in certain network scenarios where capacity and the cross traffic intensity vary substantially. Therefore, we propose a second multi-class classification-based method that is based on supervised learning and can provide available bandwidth estimates accurately and fast using fewer probe packets.

We show that both of our methods perform better than the fluid-flow model-based direct probing technique that employs a Kalman filter [34] in network scenarios where the deterministic fluid-flow model assumptions fail due to bursty cross traffic nature and multiple tight links. We show that both methods converge faster and their estimates have less variations around the actual available bandwidth values. We further compare our supervised learning-based classification method with its regression-based counterpart. In addition, we evaluate the performance of our supervised learning-based method in network scenarios where the tight link capacity and the cross traffic intensity vary over time randomly and show that our method generates better results. We finally show that we can employ filtering techniques to the bandwidth estimates and decrease the estimation errors assuming that the changes in available bandwidth over time are correlated.

We provide data sets generated in a controlled network testbed located at Leibniz University Hannover. For reinforcement learning based-method, which does not require training, we provide only testing data set and for supervised learning-based method, we provide both training as well as testing data sets [22]. Both the proposed multi-class classification methods are based on joint work with Sami Akın [21, 49].

The remainder of this chapter is organized as follows. We introduce the reference implementation of the state-of-the-art model-based direct probing technique in Sec. 5.1. We present our reinforcement learning-based approach in Sec. 5.2 and show test results in Sec. 5.3. We describe our supervised learning-based method in Sec. 5.4 and evaluate our proposed method under different scenarios in Sec. 5.5.

5.1 MODEL-BASED REFERENCE IMPLEMENTATION

In order to understand how much performance gain our multi-class classification-based methods realize, we compare the performance of reinforcement learning-based as well as supervised learning-based methods with a piece-wise linear model-based direct probing technique that employs a Kalman filter [34]. Although available bandwidth estimation tools significantly differ in the selection and amount of probe traffics in experimental testbeds, we tailor our classification-based techniques and the direct probing method to work on the same database in order to provide a fairground for comparison. Hence, in the following, we initially describe the direct probing technique and describe our multi-class classification-based techniques in the next sections.

5.1.1 *Direct probing*

For the direct probing technique, we implement a Kalman filter to estimate the available bandwidth. Furthermore, to increase the convergence speed of the filter, we use a multi-rate probe stream to probe the network path over several input rates in each experiment. Our probe stream consists of a series of p packet trains of n packets each. The p packet trains correspond to p different probe rates with a constant rate increment of δ_r between successive trains. We define the inter-packet strain as [34]

$$\xi(t) = \frac{r_{in}}{r_{out}} - 1 \quad \text{for } r_{in} > C(t) - \lambda(t), \tag{5.1}$$

and $\xi(t) = 0$, otherwise, where t is the discrete time index. Now, inserting $\xi(t)$ into Eq. (2.7) when $r_{in} > C(t) - \lambda(t)$, we obtain

$$\xi(t) = r_{in}\frac{1}{C(t)} + \frac{\lambda(t) - C(t)}{C(t)}. \tag{5.2}$$

Subsequently, we define $\alpha(t) = \frac{1}{C(t)}$ and $\beta(t) = \frac{\lambda(t) - C(t)}{C(t)}$, and obtain the inter-packet strain parameter as

$$\xi(t) = \begin{cases} 0, & \text{if } r_{in} \leqslant C(t) - \lambda(t), \\ \alpha(t) r_{in} + \beta(t), & \text{if } r_{in} > C(t) - \lambda(t). \end{cases} \tag{5.3}$$

Following the assumptions of the fluid-flow model, we can see that the $\xi(t)$ is zero in the absence of congestion, and it grows proportional to the probe traffic rate r_{in} when the probing rate exceeds the available bandwidth. As can be seen in Eq. (5.3), the model is piece-wise linear due to the sharp bend at $r_{in} = C(t) - \lambda(t)$, which inhibits the direct application of the Kalman filter. In order to overcome the problem, we feed only those measurements that satisfy $r_{in} > \hat{A}(t-1)$ to the filter, where $\hat{A}(t-1)$ is the available bandwidth estimate by the direct probing technique in the $(t-1)^{th}$ time frame. Since, the direct probing technique seeks to estimate the upward line segment of the rate response curve which is determined by two parameters $C(t)$ and $\lambda(t)$, we can express the state of the system with a state vector containing two unknown parameters as

$$\mathbf{x}(t) = \begin{bmatrix} \alpha(t) \\ \beta(t) \end{bmatrix}. \tag{5.4}$$

Assuming that the network statistics remain constant over a long period of time compared to the time period t, we set the state transition matrix of Kalman filter to the identity matrix. Hence, we define the state transition process as [34]

$$\mathbf{x}(t) = \mathbf{x}(t-1) + \mathbf{w}(t), \tag{5.5}$$

where $\mathbf{w}(t)$ is the process noise vector, which is zero-mean and Gaussian-distributed with 2×2 covariance matrix Q. Subsequently, we define the measurement model as

$$\mathbf{z}(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{v}(t), \tag{5.6}$$

where $\mathbf{z}(t)$ and $\mathbf{v}(t)$ are the $p \times 1$-dimensional vectors of measured strain and measurement noise, respectively. Recall that in each measurement, we send packets at p different input rates, r_{in}, and hence, we have the $p \times 1$-dimensional observation vector. Herein, $\mathbf{v}(t)$ is zero-mean and Gaussian distributed with $p \times p$ covariance matrix R. Moreover, the observation matrix, $\mathbf{H}(t)$ is a $p \times 2$ matrix and is given as

$$\mathbf{H}(t) = \begin{bmatrix} r_{in}^{1}(t) & 1 \\ \vdots & \\ r_{in}^{p}(t) & 1 \end{bmatrix},$$

where $r_{in}^{i}(t)$ for $i \in \{1, \cdots, p\}$ is a function of time because we choose the input rates at t such that the input rates are greater than the available bandwidth estimate at $t-1$, i.e., $r_{in}^{i}(t) > \hat{A}(t-1)$.

The authors in [34] define Q as a parameter that indicates the deviation of the system from the fluid-flow model, and use it as a tuning parameter to increase the estimation quality. Q, being a symmetric matrix, provides three degrees of freedom for tuning; however, we use it in a simple form as $Q = \eta\mathbf{I}$, where $\mathbf{I}$ is the 2×2 identity matrix. We set $\eta = 10^{-2}$, in our settings because it allows faster convergence and less variations in available bandwidth estimates.

5.2 REINFORCEMENT LEARNING-BASED METHOD

In this section, we present our reinforcement learning-based method for available bandwidth estimation where we consider a set of input rates as a set of actions, and define a reward metric as a function of input and output rates. Our method employs the exploration-exploitation mechanism to find the input rate which maximizes a cumulative reward function and reports this input rate as available bandwidth. In the sequel, we start with the ε-greedy search algorithm, and then discuss the reward function mechanism and the convergence speed of our method.

5.2.1 *ε-greedy Algorithm*

Let us consider a finite-state MDP with an agent and an environment as shown in Fig. 5.1. Let us further consider that there are a set of states, $\mathcal{S}$, a set of actions, $\mathcal{A}$, and a set of rewards, $\mathcal{R}$. Here, we assume that there exists a bijective function between the sets of actions and states, and the set of rewards. Particularly, there exists a one-to-one correspondence between the action-state pairs and the rewards. At time t, the agent in state $s_t \in \mathcal{S}$ chooses an action $a_t \in \mathcal{A}(s_t)$, and the environment returns a reward, $\rho_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and changes the agent's state to $s_{t+1} \in \mathcal{S}$. Here, $\mathcal{A}(s_t)$ refers to the set of actions that the agent chooses when it is in state s_t. Particularly, $\mathcal{A}(s_t)$ is a subset of $\mathcal{A}$. In a stochastic environment, the reward values following an action in one state can be samples from a distribution with a mean and variance. In this case, the reward of the

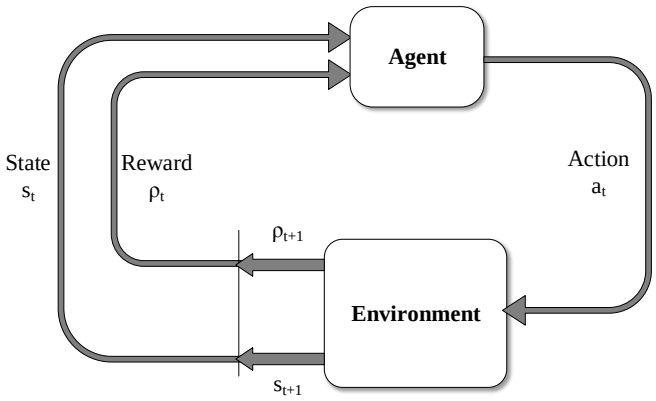


Figure 5.1: An agent-environment interaction [50].

action in that state can be the average of rewards received until the last time the action is chosen and the agent is in that state. Under these conditions, given that the agent is in state s_t, the ε-greedy algorithm chooses with probability $1-\varepsilon$ the action $a_t \in \mathcal{A}(s_t)$ that performs the best with respect to reward returns and selects uniformly one action among the others with probability ε. Here, ε indicates how greedy the agent is, and the optimal value of ε is important for the agent to decide whether to exploit or explore more in order to find the action that performs the best on average. When ε is small, the agent converges to a reward value slowly and stabilizes on an action. Although it is more stable in the long run, yet there is a risk that the reward value is not the maximum reward the agent could have. On the other hand, when ε is large, it takes short to converge to a reward value, but there are too much variations in the long-run even if the measurements are not very noisy [50].

In our experiments, we consider that the network is stationary, i.e., the network statistics remain constant for the time interval during which we make our measurements and estimate the average available bandwidth. Therefore, we treat available bandwidth estimation as a single-state MDP. Furthermore, we consider a probe stream consisting of a series of p packet trains that corresponds to p different probe rates. Herein, the set of input probe rates, i.e., $r_{in} \in \{\delta_r, 2\delta_r, \ldots, p\delta_r\}$ corresponds to the set of actions, $\mathcal{A}$. Hence, we define available bandwidth estimation as a multi-armed bandit problem with a single state. We further define a reward parameter (as a function of r_{in} and r_{out}) that reaches the maximum when the input probe rate r_{in} is equal to the available bandwidth in the network. Following the selection of one probe rate among p input rates, its associated reward is received. As the reward values are perturbed due to noisy measurements, we rely on the corresponding average rewards after a probing rate is selected. Particularly, we calculate the action-

value function $Q_t(r_{in})$ that estimates the value for choosing r_{in} at time step t by calculating the average rewards received up to time $t-1$ as

$$Q_t(r_{in}) = \frac{\sum_{j=1}^{t-1} \rho_j i_j(r_{in})}{\sum_{j=1}^{t-1} i_j(r_{in})}, \tag{5.7}$$

where $i_j(r_{in})$ is the indicator function which is set to 1 whenever the input rate r_{in} is chosen up to time $t-1$ and is 0 otherwise. Here, the ε-greedy algorithm at time t chooses the input probe rate that has the maximum average reward up to time $t-1$ with probability $1-\varepsilon$. Specifically, the algorithm sets the input rate at time t, i.e., r_{in_t} as

$$r_{in_t} = \arg\max_{r_{in} \in \mathcal{A}} \{Q_t(r_{in})\},$$

otherwise, the algorithm uniformly chooses any rate among others with probability ε and sets the input rate.

5.2.2 *Choice of Reward Function*

In real-time available bandwidth estimation, the major challenge is to define a function that produces a credible reward even in the presence of noisy measurements due to non-fluid traffic, multiple bottlenecks and inaccurate time stamping. Motivated by the characteristic rate response curve of a network, the goal is to define reward metric ρ that reaches the maximum when the input probe rate r_{in} is equal to the available bandwidth as shown in Fig. 5.2. Hence, we define the reward function as

$$\rho = r_{out}(r_{in})^{\gamma-1}, \tag{5.8}$$

where γ is the convergence parameter that has to satisfy $0 < \gamma < 1 - \frac{\lambda}{C}$ for the reward function to be maximum at $r_{in} = C - \lambda$. We set the exploration rate $\varepsilon = 0.1$ and show the measured reward function for convergence parameter $\gamma \in [0.2, 0.3, 0.4]$ as a function of r_{in} averaged over 1000 repeated measurements in Fig. 5.3a. The network has a single tight link of capacity $C = 100$ Mbps and access links are of capacity 1 Gbps. The cross traffic is exponential with an average rate $\lambda = 50$ Mbps. The packet size of cross traffic and probe traffic is $l = 1514$ B. A probe stream comprises $p = 100$ packet trains of $n = 100$ packets sent at $p = 100$ different probe rates with a constant rate increment $\delta_r = 2$ Mbps between successive trains. The error bars depict the SD from the average reward values which increases when the probing rate reaches beyond available bandwidth, i.e.,

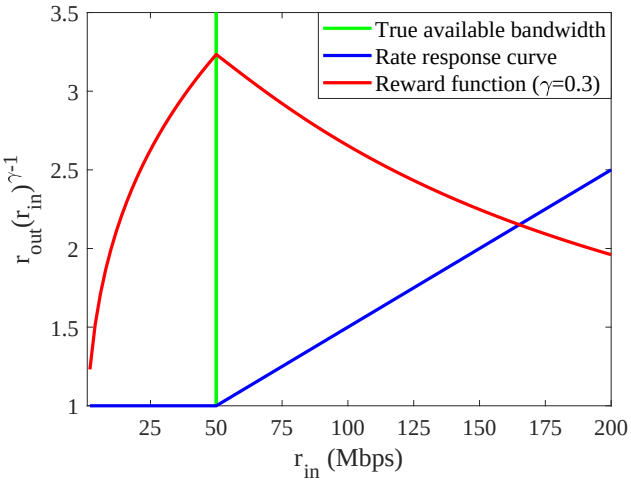


Figure 5.2: Reward metric as a function of r_{in} and r_{out}. The reward function reaches the maximum when the input probe rate r_{in} is equal to the available bandwidth.

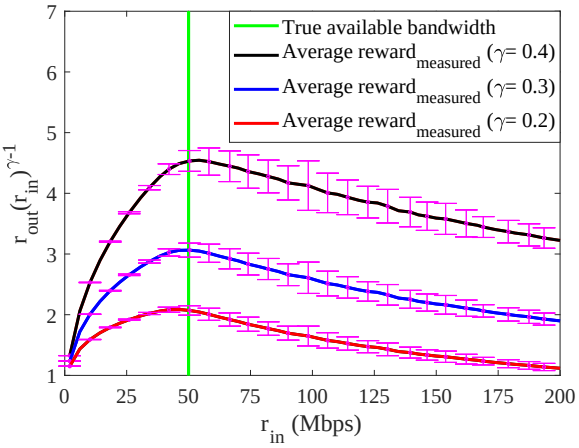
$r_{in} > C - \lambda$ due to building up of queues at the multiplexer. As seen in Fig. 5.3a, the reward function is maximized when r_{in} is equal to the available bandwidth which is 50 Mbps. We also note that with decreasing γ, the reward function also decreases which leads to slower convergence because the impact of noise is more belligerent with decreasing reward function when differentiating the maximum reward from others.

5.2.3 *Convergence Speed*

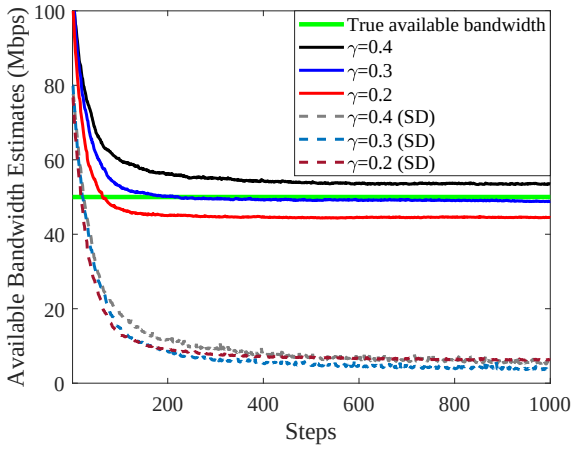
Our proposed reinforcement learning-based method is a continuous process. Once the convergence is reached, it produces a stable value of available bandwidth estimate. However, the speed at which it converges depends upon the choice of two parameters, i.e., γ and ε.

5.2.3.1 *Choice of γ*

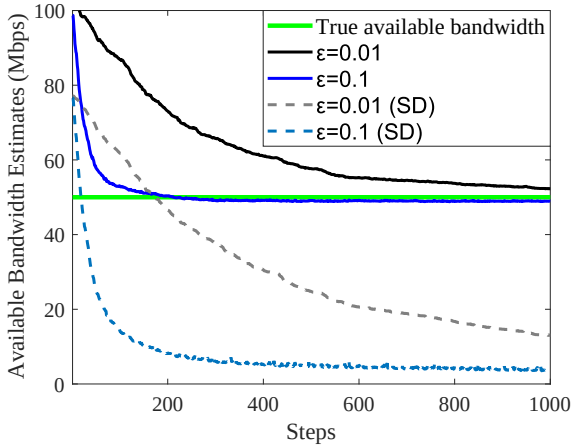
In a network with unknown C and λ, it is not trivial to determine γ, which depends on these unknowns by definition. To analyze the effects of γ on the convergence speed, we plot the average available bandwidth estimates of 1000 repeated experiments over 1000 steps for $\gamma \in [0.2, 0.3, 0.4]$ as shown in Fig. 5.3b. At each step, our method chooses one of the input rates among p input rates following the ε-greedy algorithm and provides a single available bandwidth estimate. As the number of steps increases, every input rate is sampled enough number of times leading to the convergence of the input rate with maximum



(a) Reward distribution



(b) Effect of γ on convergence speed



(c) Effect of ϵ on convergence speed

Figure 5.3: (a) Reward distribution with average measured rewards and error bars depicting their SD, and (b) & (c) average available bandwidth estimates and their SD for different values of γ and ϵ, the two parameters that affect the convergence speed of reinforcement learning-based method.

reward value to the available bandwidth. Furthermore, we use SD as a metric to measure the precision of the bandwidth estimates:

$$SD = \sqrt{\frac{1}{K_r - 1} \sum_{i=1}^{K_r} (\hat{A}_i - \bar{A}_i)^2}, \tag{5.9}$$

where $\hat{A}$ and $\bar{A}$ are the estimated and the average true values of the available bandwidth, respectively and K_r is the number of repeated experiments over which we obtain the average available bandwidth estimates and their SD around the true available bandwidth. We set $K_r = 1000$ and the exploration rate to $\varepsilon = 0.1$ unless otherwise stated. As seen in Fig. 5.3b, the convergence is faster when $\gamma = 0.3$, i.e., the method detects the available bandwidth after 200 steps. On the other hand, it takes more than 1000 steps on average for the method to converge to the available bandwidth when $\gamma = 0.2$ and $\gamma = 0.4$. It takes longer to converge when $\gamma = 0.2$ because the reward function decreases with decrease in γ as can be seen in Fig. 5.3a. The impact of noise becomes more hostile with decreasing reward function and the method has to explore more in order to differentiate the best input rate, which returns the maximum cumulative reward, from others. On the other hand, when γ is set to 0.4, possibly significant deviations due to random cross traffic cause γ to exceed the upper limit of the aforementioned range $0 < \gamma < 1 - \frac{\lambda}{C}$ at certain steps, and therefore the method takes longer to converge. We plot the graphs until 1000 steps for clarity in the comparison of different γ values. One can run the experiment for more steps and can easily observe that as long as the convergence parameter satisfies $0 < \gamma < 1 - \frac{\lambda}{C}$, the method will converge. However, the convergence speed depends not only on γ but on the ε as well.

5.2.3.2 *Choice of ε*

The choice of ε dictates the exploration-exploitation trade-off in reinforcement learning-based methods. Hence, in order to understand the impact of ε, we plot the average available bandwidth estimates and their SD for $\varepsilon \in [0.01, 0.1]$ with the convergence parameter set to $\gamma = 0.3$ as shown in Fig. 5.3c. The larger exploration rate $\varepsilon = 0.1$ leads the method to explore more and find the available bandwidth faster when compared to the smaller exploration rate $\varepsilon = 0.01$. Although the method converges more quickly with a larger ε, yet it performs better with a smaller ε eventually when noise variance is low. This is because with a larger ε, the method tests other values very often, which leads to more variations in the available bandwidth estimation in the long-run. However, in our experiments we set a large value of $\varepsilon = 0.1$ so that we can increase the convergence speed of our proposed method. Furthermore, to alleviate the vari-

ations in the available bandwidth estimates, we average over $K_r = 1000$ repeated experiments.

5.3 EXPERIMENTAL EVALUATION

We compare the performance of our reinforcement learning-based proposed method with the direct probing technique in a controlled network testbed described in Chapter 4.

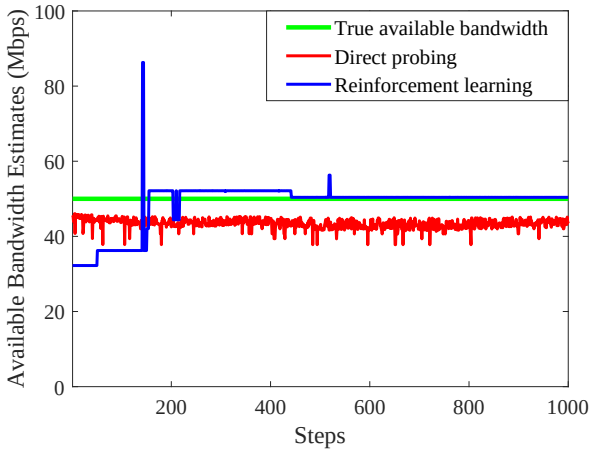
In Fig. 5.4a, Fig. 5.4b, Fig. 5.4c, we show available bandwidth estimates obtained from direct probing and reinforcement learning-based method over 1000 steps in the presence of exponential cross traffic of an average rate $\lambda = 50$ Mbps. The tigh link and access link capacities are 100 Mbps and 1 Gbps, respectively. The packet size of cross traffic and probe traffic is $l = 1514$ B including the Ethernet header. A probe stream comprises p packet trains sent at 100 different probe rates with a constant rate increment $\delta_r = 2$ Mbps between successive trains. Each packet train consists of $n = 100$ packets. The convergence parameter γ and the exploration rate ε are set to 0.3 and 0.1, respectively. We use these setting for the rest of the section unless otherwise stated. It can be seen from the results that our method outperforms the other method. However, in order to have a better view from a statistical perspective, we perform $K_r = 1000$ experiments and compare the average estimation performances and their SD around the available bandwidth values in the sequel.

5.3.1 *Cross Traffic Burstiness*

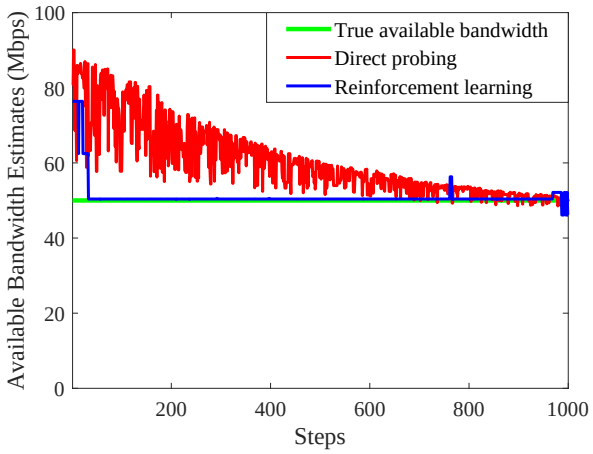
In order to evaluate how our method performs in the presence of cross traffic with an unknown burstiness, we consider three types of cross traffic as considered previously in Sec. 4.3.4.1.

1. No burstiness with constant bit rate,
2. Moderate burstiness due to exponential packet inter-arrival times,
3. Heavy burstiness due to Pareto inter-arrival times with infinite variance defined by a shape parameter $\alpha_s = 1.5$.

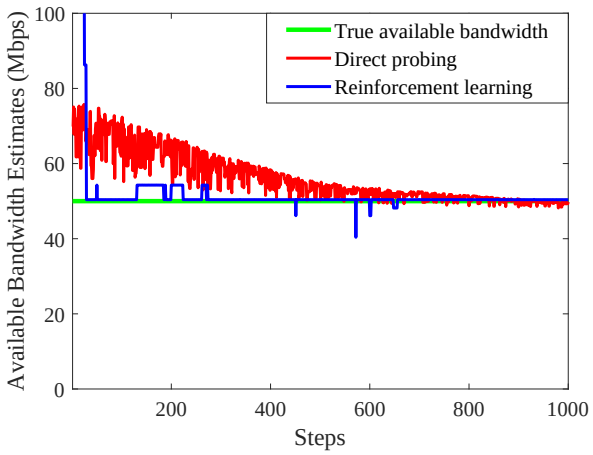
The average rate of cross traffic is $\lambda = 50$ Mbps in all cases. As before, the tight link and access links capacities are $C = 100$ Mbps and $C = 1$ Gbps, respectively. As can be seen in Fig. 5.5a, Fig. 5.5b and Fig. 5.5c, the bandwidth estimates of the reinforcement learning-based method are more accurate with low SD when compared to the direct probing technique. We can see the significant improvement in the convergence speed when we employ the reinforcement



(a) $K_r(1)$



(b) $K_r(2)$



(c) $K_r(3)$

Figure 5.4: Available bandwidth estimates for randomly selected the first three repeated experiments.

learning-based method irrespective of the cross traffic burstiness. Particularly, the reinforcement learning-based method is robust to the deviations from the fluid-flow model. In the case of direct method, an increase in the variability of available bandwidth estimates as well as an underestimation bias can be seen due to burstiness of the cross traffic.

5.3.2 *Cross Traffic Intensity*

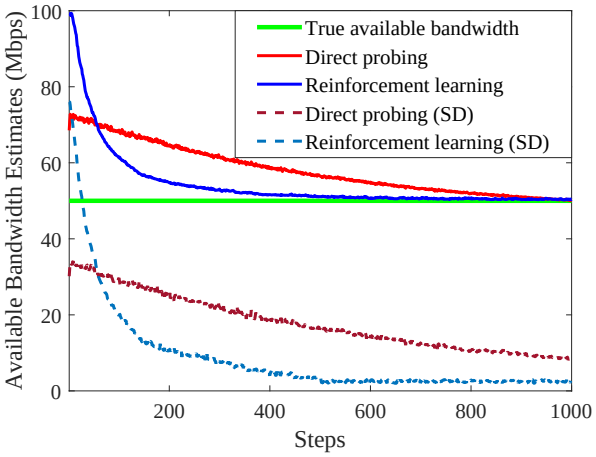
To evaluate the impacts of cross traffic intensity on available bandwidth estimation, we deploy exponential cross traffic with average rates $\lambda \in \{25, 50, 75\}$ Mbps and depict the average of the available bandwidth estimates and their SD around the true available bandwidth in Fig. 5.6a, Fig. 5.6b and Fig. 5.6c, respectively. While the SD increase in the direct probing technique with the increasing cross traffic, the SD in the reinforcement learning-based method remain almost unchanged in all the cases. Moreover, the reinforcement learning-based method converges to the available bandwidth faster than the direct probing technique does.

5.3.3 *Multiple Tight Links*

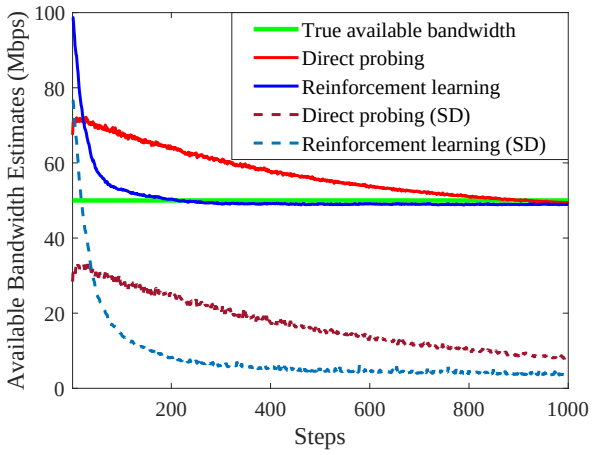
We extend our testbed from the single-hop network to the multi-hop network as shown in Fig. 4.1 in order to test the reinforcement learning-based method in multiple tight links. While traversing the entire network path with the tight link capacity $C = 100$ Mbps and the access links with capacity 1 Gbps, the path-persistent probe streams experience single hop-persistent cross traffic with exponential packet inter-arrival times and average rate $\lambda = 50$ Mbps. We show in Fig. 5.7 that the reinforcement learning-based method provides more accurate available bandwidth estimates, whereas the other method fails to converge to the available bandwidth. As explained in chapter 4, this is due to the fact that in the case of multiple tight links, the probe stream has a constant rate r_{in}, with a defined input gap g_{in}, only at the first link. In the following links, the input gaps have a random structure as they are the output gaps from the preceding links [8, 9, 12]. For the direct method, the inter-packet strain ξ does not grow linearly with the cross traffic in multi-hop networks [51] which causes the underestimation of the available bandwidth.

5.3.4 *Tight Link differs from Bottleneck Link*

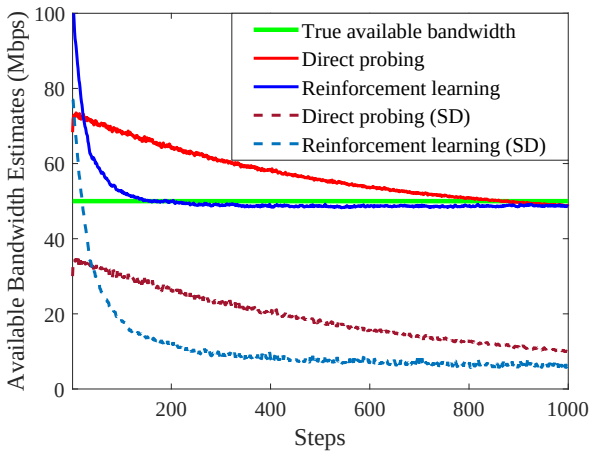
The available bandwidth estimation in multi-hop networks becomes more difficult if the tight link of a network path is not same as its bottleneck link. We



(a) Constant Bit Rate

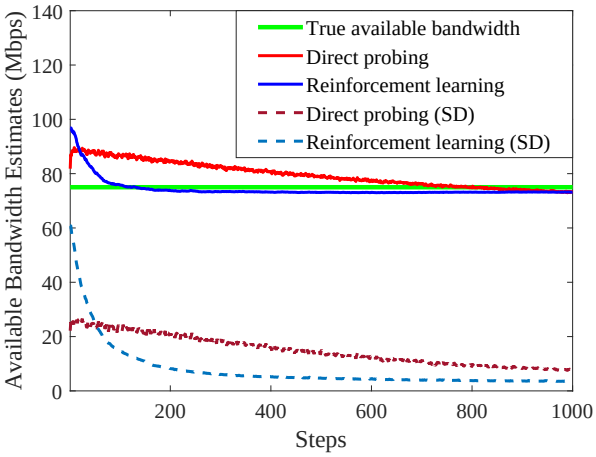


(b) Exponential

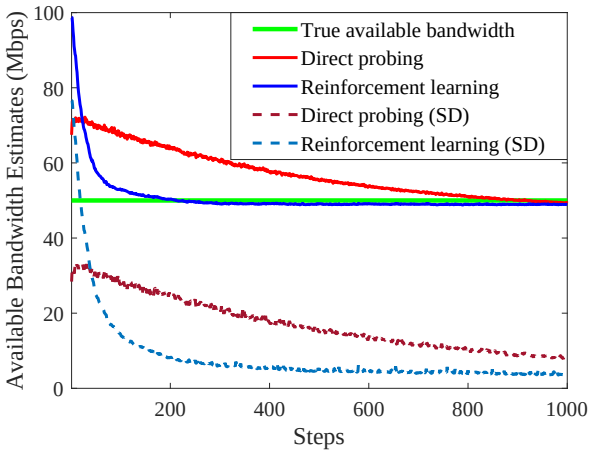


(c) Pareto

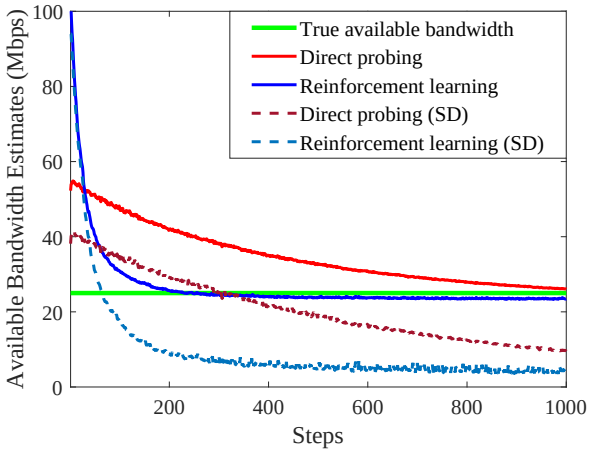
Figure 5.5: Average available bandwidth estimates and their SD for different types of cross traffic burstiness.



(a) $\lambda = 25$ Mbps

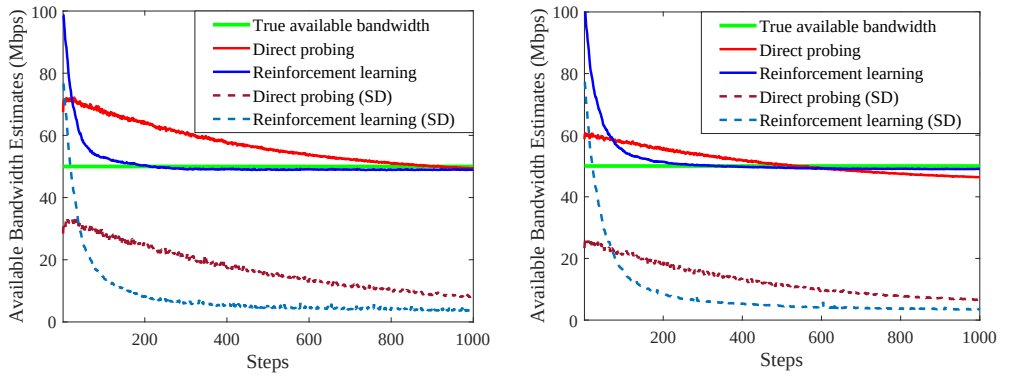


(b) $\lambda = 50$ Mbps

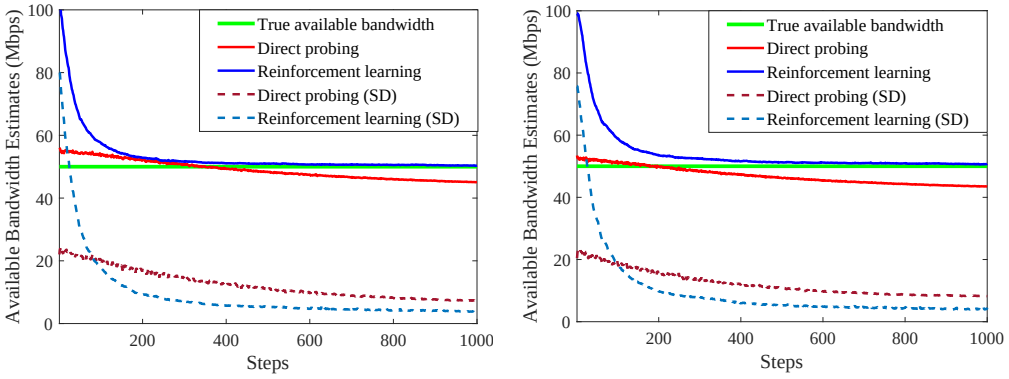


(c) $\lambda = 75$ Mbps

Figure 5.6: Average available bandwidth estimates and their SD for different exponential cross traffic rates $\lambda \in \{25, 50, 75\}$ Mbps.

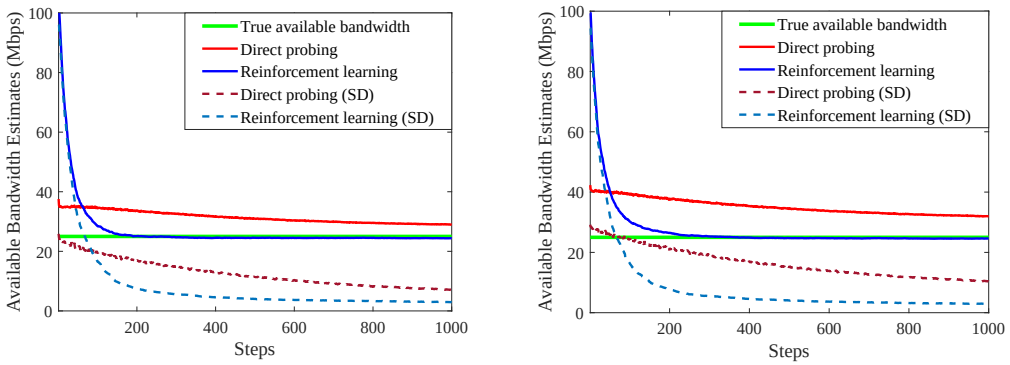


(a) 1-hop (b) 2-hop



(c) 3-hop (d) 4-hop

Figure 5.7: Average available bandwidth estimates and their SD for multiple tight links with capacity $C = 100$ Mbps in the presence of single hop-persistent exponential cross traffic with an average rate $\lambda = 50$ Mbps.



(a) Scenario I (b) Scenario II

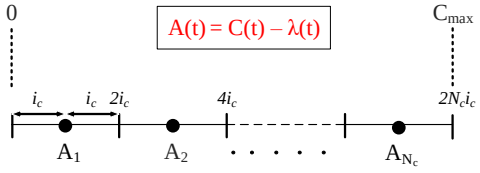
Figure 5.8: Average available bandwidth estimates and their SD with the tight link (a) succeeding and (b) preceding the bottleneck link.

investigate the available bandwidth estimation by considering two different scenarios that we described in chapter 4.

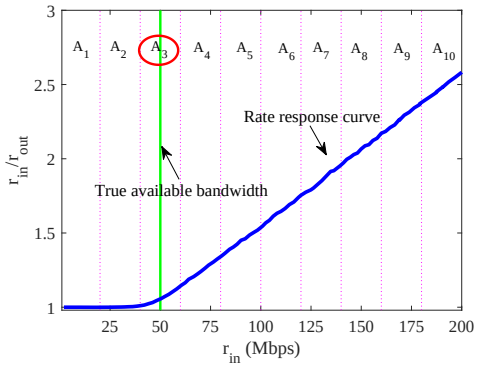
In scenario I, the bottleneck link appears before the tight link. In scenario II, the bottleneck link comes after the tight link. We set the tight link capacity to $C = 100$ Mbps and the bottleneck capacity to $C_b = 50$ Mbps in both the scenarios. The cross traffic is exponential with an average rate $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps, traversing the tight link and the bottleneck link, respectively. We show the available bandwidth estimates and the corresponding SD in the scenarios I and II in Fig. 5.8a and Fig. 5.8b, respectively. The reinforcement learning-based method results in more accurate and faster estimates than the direct probing technique does. However, we observe an estimation bias in the available bandwidth estimates of the direct probing technique in both scenarios. The estimation bias can be explained by the fact that in the case of network that has two congested links, the slope of rate response curve is determined by the combination of two linear segments. The congestion measure ξ, which otherwise would have grown linearly according to Eq. (5.3) for the input probing rates higher than the available bandwidth in a single tight link, grows faster when congestion occurs at both the tight and bottleneck links [34].

5.4 SUPERVISED LEARNING-BASED METHOD

In this section, we present our supervised learning-based approach, where we address the available bandwidth estimation as a multiclass classification problem. We initially discuss how available bandwidth classes are formed and define a feature vector for the classifier. Then, we present the implemented supervised learning techniques.



(a) Available bandwidth classes



(b) Rate response curve showing available bandwidth classes

Figure 5.9: (a) Available bandwidth classification between zero Mbps to C_{max} and (b) its projection onto the rate response curve of a single tight link of capacity $C = 100$ Mbps in the presence of exponential cross traffic with average rate $\lambda = 50$.

5.4.1 *Available Bandwidth Classes*

Let us consider a network path with capacity $C(t)$ and cross traffic rate $\lambda(t)$ that changes over time as shown in Fig. 5.9a, where $C(t) \in [C_{min}, C_{max}]$ and $\lambda(t) \in [0, C(t)]$. One can think of a path where the channel capacity varies over time due to environmental conditions, for instance, in wireless networks. Therefore, the available bandwidth is not constant, i.e., $A(t) = C(t) - \lambda(t)$. Here, we divide the entire possible range[1] $[0, C_{max}]$ into N_c subranges, where each subrange is denoted by one class A_c, for $c \in \{1, 2, ...N_c\}$, and each subrange covers a distance of $2i_c$ units. For instance, the centers of classes A_1 and A_2 are $2i_c$ units apart from each other as can be seen in Fig. 5.9a. Basically, given a feature vector, our classification-based estimator returns the class that the available bandwidth belongs to and sets the estimate to the class center. For instance, if the classification-based estimator returns A_c as the estimated class, then the available bandwidth estimate is $(2c - 1)i_c$ units. Here, one can observe that the smaller the subrange a class refers to, the more classes we need to assign to cover the entire possible range from 0 to C_{max}. Particularly, there is a trade-off between the number of classes and the subrange a class encompasses; the training process takes longer with more classes, and we need to obtain more experimental data for training, but the deviation around the available bandwidth decreases. In order to understand the aforementioned technique, let us consider the experimentally obtained rate response curve shown in Fig. 5.9b as an example, where $C_{max} = 200$ Mbps. The entire possible range,

1 One can define a range with borders from a non-zero value to another value that is greater than C_{max}.

i.e., $[0, 200]$ Mbps is divided into 10 subranges and the centers of each class are $2i_c = 200/10 = 20$ Mbps apart from each other.

We define an input to the classifier as a p-dimensional feature vector, where each instance is the ratio of input rate at the sender and corresponding output rate at the receiver r_{in}/r_{out}. Here, r_{in} values are spaced equidistantly with increment δ_r, and hence we have a vector of input rates $r_{in} = [r^1_{in} ... r^p_{in}]$, where $r^i_{in} = i\delta_r$ for $i \in \{1, ..., p\}$. Considering the case $C_{max} = 200$ Mbps as an example, one can set $\delta_r = 25$ Mbps and can use $p = 8$ packet trains[2]. Subsequently, a probe stream comprising p probe trains is sent to the network with defined p input rates, and the corresponding output rates are obtained. Hence, a feature vector, $f = \left[\frac{r^1_{in}}{r^1_{out}}, ..., \frac{r^p_{in}}{r^p_{out}}\right]$, is provided as an input to the classification-based estimator. If the classification-based estimator returns A_3 as the estimated class, then the available bandwidth estimate $(2c - 1)i_c$ is 50 Mbps.

5.4.2 *Training Data*

To investigate the sensitivity of our method with respect to different network parameters, we set up a number of topologies, where the tight link capacity and the cross traffic intensity are chosen randomly. The capacity is chosen as a random number in the interval [10, 200] Mbps, with the cross traffic intensity chosen relative to the tight link capacity as $\lambda = U[0, 1] \cdot C$ Mbps where U denotes uniform distribution. The access link capacity for all networks is $C = 1$ Gbps. The link and traffic characteristics are emulated as described in chapter 4.

A probe stream consists of a series of p packet trains, each having n packets. These p different packet trains respectively correspond to p different probe rates that successively increase with an increment rate δ_r. In order to reduce the amount of probe traffic injected into network path, we set $p = 6$, and to reduce the impact of noise-afflicted output gaps, we compute r_{out} over $n = 100$ packets. The packet trains are sent at 6 different rates with a rate increment δ_r = 25 Mbps unless otherwise stated. For each of random network configuration 100 repeated experiments are performed. Particularly, having more than 100 different network configurations, we run more than 10000 experiments to generate different data sets for training and testing.

2 Notice that here the maximum input rate $p\delta_r$ is not limited by the maximum capacity, C_{max}. However, in order to prevent any congestion, it is better to limit the maximum input rate to a value smaller than C_{max}.

5.4.3 *Machine Learning Techniques*

To evaluate our machine learning-based approach for bandwidth classification, we consider the following supervised machine learning techniques.

5.4.3.1 *Support Vector Machines*

SVMs are supervised learning algorithms, which can be used as Support Vector Classification (SVC) for classification-based and as SVR for regression-based problems. SVM enables plugging to project an input feature vector to a higher dimensional space and find a hyper-plane that separates two given classes. In this work, we use the GRBF. However, one can use polynomial kernel functions as well. We observe that in our experimental settings, even the polynomial function with degree 2 performs well but the performance of GRBF is better than polynomial function in general when data becomes more noisy. As SVM is originally designed for binary classification [52], Error-Correcting Output Codes (ECOC) [53] are used to reduce a multi-class problem to a set of multiple binary classification problems. As described in [53], the output coding for multiclass problems is composed of two stages. In the training stage, multiple independent binary classifiers are constructed, where each classifier is trained to distinguish between two disjoint subsets of the bandwidth estimation labels. In the second stage, i.e., the classification part, the predictions of trained binary classifiers are combined to test instances, and a voting scheme is used to decide the available bandwidth class.

5.4.3.2 k-*Nearest Neighbor*

k-NN algorithm is a non-parametric lazy supervised machine learning algorithm that can solve both classification and regression problems. The principle behind k-NN classification is the majority voting rule applied over k nearest neighbors, where the test data is assigned the available bandwidth class that is chosen by the majority vote of its k nearest neighbors. k-NN mainly involves two hyper-parameters, i.e., the number of neighbors involved in the decision, k, and the distance function denoting how far neighbors are from each other. Small k provides a flexible fit, which has low bias but high variance. On the other hand, large k averages more voters in each prediction and hence is more resilient to outliers, which means lower variance but increased bias. Regarding a bias-variance trade-off, we use weighted k-NN, where we set the nearest neighbor $k = 10$. Further, each nearest neighbor is given a weight of $1/d^2$, where d is set as the Euclidean distance metric. We choose k-NN for the bandwidth classification problem because it is non-parametric, and therefore, we can avoid mismodeling the underlying distribution of the data. In real bandwidth

estimation problems, which deviate from the fluid-flow model assumptions, it is difficult to find the underlying data distribution that matches the theoretical distributions. For example, choosing a learning model that assumes a Gaussian distribution for non-Gaussian distributed data will cause the algorithm to make poor predictions. In contrast to SVM, k-NN is instance-based, and hence, does not explicitly learn a model. Instead, it chooses to memorize the training instances, which are subsequently used as *knowledge* in the prediction phase. The minimal training phase of k-NN results in memory cost as we have to store possibly a huge data set. Moreover, it causes computational cost in the testing phase as well because classifying a given feature vector requires a run-down of the whole data set.

5.4.3.3 *Bootstrap Aggregation*

The main causes of error in learning models are noise, bias, and variance. However, for certain bandwidth applications, an accurate and reliable available bandwidth estimation with less bias and variance is of utmost importance, e.g., to minimize the adverse effects of quality variance and video freezing on viewers in rate-adaptive applications, such as Dynamic Adaptive Streaming over HTTP (DASH). In order to obtain better predictive performance, we consider ensemble learning methods for classification such as bagging and boosting to decrease the model's variance and bias, respectively. Bagging uses a bootstrapping technique to randomly sample the input data with replacement to create multiple subsets of data. Then, a set of models is trained on each of these subsets, and their predictions are aggregated to make the final combined prediction using averaging. The averaging of multiple decision trees reduces the variance and avoids over-fitting. The performance of the ensemble technique depends on the setting of both the ensemble and that of the weak learners. A large number of trees increase the performance and make the predictions more stable, but having many trees slows down the computation, which is not desired for real-time bandwidth predictions. We choose the hyper-parameters to have a trade-off between the predictive power and the computational speed, therefore, set the number of ensemble learning cycles to be 30.

5.4.3.4 *Adaptive Boosting*

To reduce bias in the estimation, we consider another ensemble-based algorithm: AdaBoost. Instead of training parallel models as in Bagging, AdaBoost is an iterative technique because it uses multiple iterations to generate a single composite strong learner. AdaBoost fits a classifier on the original data set. If an observation is misclassified, it tries to increase the weight of this observation. The subsequent classifier acknowledges the updated weights and attempts to

reduce the classification error of its predecessor. The procedure is repeated over and over again until the required accuracy is achieved. To achieve the balance between speed and accuracy, we set the learning rate to 0.1 and the number of ensemble members to 30.

5.4.3.5 *Neural Network*

NNs are complex models composed of simple elements, called neurons, which try to mimic the way the human brain develops classification rules. We use a feed-forward Multi Layer Perceptron (MLP) consisting of three different layers of neurons: an input layer, a hidden layer, and an output layer, with each layer receiving inputs from previous layers, and passing outputs to further layers. The neural net iterates for a predetermined number of iterations, called epochs. After each epoch, the cost function is analyzed to see where the model could be improved. We decided for a shallow neural network consisting of one hidden layer with 40 neurons. For training of the neural network, we first implement an autoencoder for each layer separately and then fine-tune the network using a SCG. Given a classification network, we optimize the cross-entropy error function requiring approximately 1000 epochs until the convergence is achieved. Due to the limited amount of training data, the shallow network with a small number of hidden neurons allows training without much over-fitting.

5.4.4 *Evaluation Metrics*

In order to gain a better assessment of the effectiveness of the aforementioned machine learning models in the testing phase, we use the K-Fold cross-validation technique. As more partitions lead to a smaller bias but a higher variance, we set $K=5$ to have a balanced bias-variance trade-off.

We employ the performance measures described in [54], i.e., *accuracy*, *precision*, and *recall*, to experimentally substantiate our classification-based available bandwidth estimation technique. The accuracy of estimates provides the overall effectiveness of the method and calculates the ratio of the number of available bandwidth estimates that are classified correctly to the total number of available bandwidth estimates. We calculate the average accuracy Acc_{avg} over N_c available bandwidth classes as follows:

$$Acc_{avg} = \frac{\sum_{c=1}^{N_c} \frac{TP_c+TN_c}{TP_c+FN_c+FP_c+TN_c}}{N_c}.$$

Recalling from Sec. 5.4.1 that A_c is the class representing c^{th} subrange of the entire possible range of available bandwidth, i.e., $[0, C_{max}]$, we have true positives (TP_c) as the number of correct bandwidth estimates classified to A_c,

false positives (FP_c) as the number of incorrect estimates assigned to A_c, true negatives (TN_c) as the number of correct estimates not assigned to A_c, and false negatives (FN_c) as the number of incorrect estimates not assigned to A_c. However, the accuracy parameter is not always the best measure to determine the effectiveness of a classification-based estimator in multi-class classifications due to the class imbalance problem. The randomly changing capacity and cross traffic of a network path may cause the number of instances of one class to exceed than the others. Here, each instance refers to the ratio of input rate at the sender and the corresponding output rate at the receiver, i.e., r_{in}/r_{out}. With imbalanced data, the test instances belonging to a small class are misclassified more often as compared to the prevalent classes with more instances. Therefore, we consider the precision and recall parameters as well.

The precision parameter indicates how precise or correct the estimates are, i.e., the bandwidth estimates that are correctly classified to the available bandwidth class, A_c, out of all the bandwidth estimates, including the incorrect ones that are classified to A_c. The recall parameter shows the sensitivity of the classification-based estimator, i.e., how many of all the actual bandwidth estimates that belong to A_c are detected correctly. High scores for both the precision and recall parameters show that the classification-based estimator returns accurate results as well as a majority of all positive results, respectively. Herein, since the precision and recall parameters are inversely related to each other, we compute F-measure by taking their harmonic mean as follows:

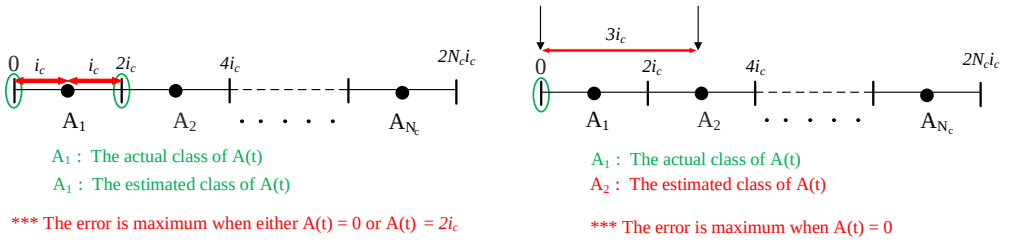
$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5.10}$$

The F-measure score takes a value between zero and one. The higher the F- measure score of a classification technique is, the better its performance is. Particularly, when the F-measure score of a classification-based estimator is one, it separates the classes without error. The over-all F-measure score of the entire classification problem can be computed by two different averaging techniques, i.e., macro-averaging (M) and micro-averaging (μ). In micro-averaging, Precision_μ and Recall_μ are obtained by summing over-all individual decisions:

$$\text{Precision}_\mu = \frac{\sum_{c=1}^{N_c} TP_c}{\sum_{c=1}^{N_c} (TP_c + FP_c)} \quad \text{and} \quad \text{Recall}_\mu = \frac{\sum_{c=1}^{N_c} TP_c}{\sum_{c=1}^{N_c} (TP_c + FN_c)}. \tag{5.11}$$

In macro-averaging, Precision_M and Recall_M are obtained for each class A_c, and then average is taken:

$$\text{Precision}_M = \frac{\sum_{c=1}^{N_c} \frac{TP_c}{TP_c + FP_c}}{N_c} \quad \text{and} \quad \text{Recall}_M = \frac{\sum_{c=1}^{N_c} \frac{TP_c}{TP_c + FN_c}}{N_c}. \tag{5.12}$$



(a) Correct classification (b) Incorrect classification

Figure 5.10: The maximum error in the case of (a) correct and (b) incorrect bandwidth classifications.

Now, we can plug Eq. (5.11) and Eq. (5.12) into Eq. (5.10) in order to calculate the F-measure score.

Furthermore, we calculate the empirical standard deviation (SD) as a metric to measure the precision of the bandwidth estimates:

$$SD = \sqrt{\frac{\sum_{t=1}^{K_r} (\hat{A}_c(t) - A(t))^2}{K_r - 1}}, \tag{5.13}$$

where t is the discrete time index and K_r is the number of measurements. Here, we assume that a measurement takes place in a certain time period, and during one period the available bandwidth stays constant. Therefore, one can consider t as the time frame index. In Eq. (5.13), $A(t)$ is the available bandwidth value and $\hat{A}_c(t)$ is the bandwidth estimate after the classification is performed. Technically, $\hat{A}_c(t)$ is the center of the class to which the bandwidth is assigned.

Notice that even if the classifier makes the correct classification, there will still be an error because each class defines a range of real numbers. In case of correct classification, the maximum error will occur when the available bandwidth value is at either of the two far edges of the actual class. For instance, let A_1 be the class the available bandwidth value belongs to and assume that the classifier estimates the class correctly. The maximum error between the available bandwidth value and the estimated value is i_c, which occurs when the available bandwidth value $A(t)$ is either 0 or $2i_c$ and the estimated value is i_c which is center of a subrange of A_1 as seen in Fig. 5.10a.

Furthermore, in case of misclassification, where a wrong class is estimated, the maximum possible error will be equal to the difference between the center of estimated class and the far edge of the actual class. For example, let A_1 be the actual class and A_2 be the estimated class. As can be seen in Fig. 5.10b, the maximum error of $3i_c$ occurs, when the estimated available bandwidth value is center of class A_2 to which estimated value is assigned, i.e., $3i_c$ and the available bandwidth value $A(t)$ is at the far edge of actual class A_1 to A_2, i.e.,

$A(t) = 0$ Mbps. Therefore, in addition to the SD provided in Eq. (5.13), we calculate an upper bound on the SD as follows:

$$SD_u = \sqrt{\frac{\sum_{t=1}^{K_r}(i_c + |\hat{A}_c(t) - A_c(t)|)^2}{K_r - 1}}, \tag{5.14}$$

where $A_c(t)$ is the actual class of the available bandwidth. Eq. (5.14) gives us the maximum deviation that we may observe at any time in our measurements.

5.5 PERFORMANCE EVALUATION

We evaluate the performance of the proposed available bandwidth estimation technique in testbed scenarios that capture the nature of practical scenarios very closely. When creating the training and the test data, we do not assume an equally likely distribution of instances to investigate the impact of *class imbalance problem* on the performance of our classification-based estimator. Furthermore, we employ cross traffic models with the exponential and Pareto distributions to reflect a moderate and heavy burstiness nature of a real network, respectively. We further consider multiple tight links with discrete and random traffic patterns. We generate a training set in the presence of exponentially distributed cross traffic as discussed in Sec. 5.4.2.

By choosing the capacity and cross traffic intensity values randomly, we generate available bandwidth values between 10 Mbps to 145 Mbps. We define the available bandwidth class centers from 1 to 200 Mbps, where each bandwidth covers a subrange of 1 Mpbs. Particularly, the set of available bandwidth classes is $\{1, 2, \cdots, 200\}$. For each available bandwidth, we repeat the experiment 100 times.

5.5.1 *Evaluation: Exponential Cross Traffic, Mutually Exclusive Classes*

For the purpose of testing, an additional data set is generated for the same network configuration as the (i) training data set in the presence of exponential traffic. In Table 5.1, we show the averaged accuracy, micro-averaged and macro-averaged F-measures after employing SVM, k-NN, Bagging, AdaBoost and NN techniques in the available bandwidth estimation. Notice that precision, recall and F-measure score are equal when calculating the micro-averaging results. This is owing to that a false positive with respect to one class will be a false negative with respect to another class in the case of misclassification. For instance, this can be also seen from the confusion matrix of SVM-based classifier

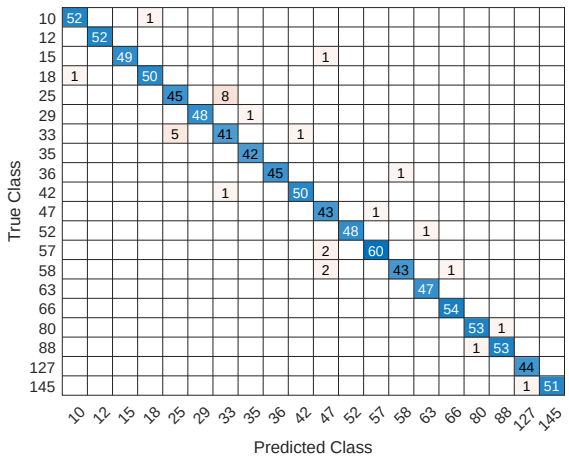


Figure 5.11: The confusion matrix obtained after using the SVM-based estimator with different network capacities and cross traffic intensities during the testing phase.

shown in Fig. 5.11, where one can show $\sum_{c=1}^{N_c} FP_c = \sum_{c=1}^{N_c} FN_c = 30$ by counting all the false instances over all the classes.

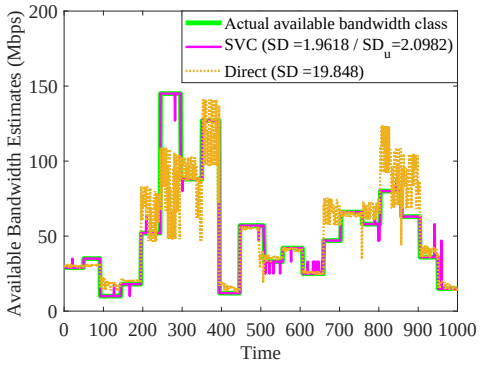
Moreover, as can be seen in Table 5.1, while all the classifiers achieve comparable accuracy values, the SVM-based classifier outperforms the rest with 99.7%. It exceeds the others in other metrics as well. Its better performance is due to the reason that SVM-based classifier projects the data to a higher dimensional feature space in order to separate the classes efficiently. Moreover, since we test all the techniques in the network we train them, looking at the results, we can say that our estimator neither over-fits nor under-fits; particularly, we are able to generalize the network model very well.

We compare the performance of the classification-based method with the model-based reference implementation of the direct probing technique described in Sec. 5.1. We also compare the classification-based method with its regression-based counterpart. We use the same testing data set for the direct method and the regression-based method that we use for the classification-based method so that both the methods can provide a reference for the classification-based method.

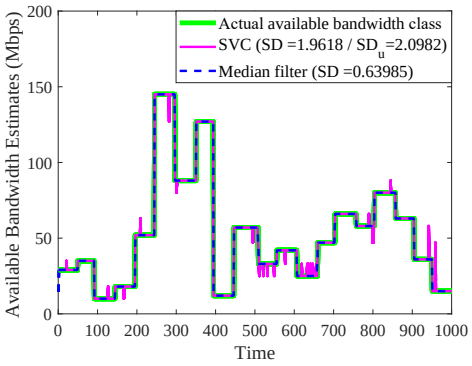
Fig. 5.12a compares the results obtained from direct probing technique and SVM-based classifier (SVC). The available bandwidth estimates obtained from SVM-based classifier are more accurate as compared to the direct method considering the SD and corresponding upper bound results (SD_u). Since the bandwidth values stay constant for a certain period, the direct probing technique can converge to the actual bandwidth. However, it is not able to track the sudden changes in the bandwidth. Moreover, as the available bandwidth approaches maximum capacity, a probe stream comprises only 6 probing rates sent with a

Table 5.1: Testing performance evaluation in a single tight link in the presence of cross traffic that has exponentially distributed packet inter-arrival times

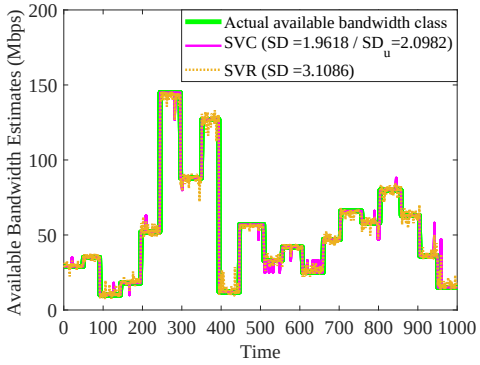
Parameters	Algorithms				
	SVM	k-NN	Bagging	AdaBoost	NN
Acc_{avg}	0.9970	0.9964	0.9941	0.9922	0.9969
$Precision_{\mu}$	0.9700	0.9640	0.9410	0.9220	0.9690
$Recall_{\mu}$	0.9700	0.9640	0.9410	0.9220	0.9690
$F1_{\mu}$	0.9700	0.9640	0.9410	0.9220	0.9690
$Precision_{M}$	0.9709	0.9648	0.9433	0.9237	0.9699
$Recall_{M}$	0.9711	0.9645	0.9426	0.9240	0.9690
$F1_{M}$	0.9710	0.9646	0.9429	0.9238	0.9694



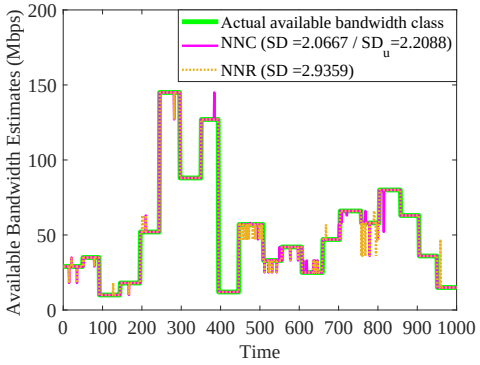
(a) Direct vs SVM-based classification



(b) Filtered estimated classes



(c) SVM-based classification vs regression



(d) NN-based classification vs regression

Figure 5.12: Available bandwidth estimates and their SD in the presence of cross traffic that has exponentially distributed packet inter-arrival times using $p = 6$ packet trains in a single tight link network.

constant rate increment $\delta_r = 25$ Mbps, are not enough to estimate the upward segment of rate response curve shown in Fig. 2.5. This effect can be seen in Fig. 5.12a, where the direct method is unable to track available bandwidth of $A(t) = 145$ Mbps, as it approaches the capacity $C(t) = 187$ Mbps of a network path, as only one measurement of inter-packet strain (ε) is available from input probing rate of $r_{in}^{max} = 150$ Mbps.

Since the bandwidth values stay constant over a certain time period, we consider the correlation among the bandwidth values over time and implement a one-dimensional median filter to smooth the estimates. Herein, one can observe that the classification errors look like the salt-and-pepper noise. Therefore, the median filter works very well in smoothing the classification errors. We set the filter order to five because increasing the order more leads to increased delays. Fig. 5.12b compares the results of estimated classes with filtered classes. The SD has been significantly reduced after applying a median filter to the estimated available bandwidth classes.

We further compare the SVM-based and NN-based results, which show the highest two performances in Table 5.1, with the results of the corresponding regression-based techniques. In the SVM-based regression (SVR) method, we set the GRBF as the kernel, and in the NN-based regression (NNR) method, we use a shallow NN consisting of one hidden layer with 40 neurons. As seen in Fig. 5.12c and Fig. 5.12d, although the results are comparable for both the classification and regression-based estimators, the SD values are better when implementing the classification-based estimators.

5.5.2 *Network Parameter Variation beyond the Training Data*

We investigate the sensitivity of our proposed method with respect to a variation of network parameters that differ substantially from the training data set. We evaluate the variability of cross traffic and multiple tight links.

5.5.2.1 *Burstiness of Cross Traffic*

In order to evaluate our method in the presence of cross traffic with an unknown burstiness, we test the aforementioned classification-based estimators with CBR that has no burstiness as assumed by the probe rate model, and Pareto traffic that has heavy burstiness with infinite variance defined by a shape parameter $\alpha_s = 1.5$.

In Table 5.2 and Table 5.3, we show the averaged accuracy, and the micro-averaged and macro-averaged F-measure results in the presence of cross traffic that has a constant bit rate and Pareto-distributed packet inter-arrival times, respectively. The classification-based estimator trained for exponential traffic

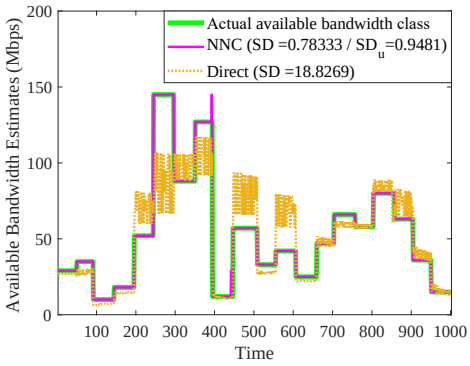
performs well when the burstiness in the cross traffic decreases as can be seen in Table 5.2 for CBR traffic. However, the misclassification increases significantly when the cross traffic becomes bursty in the case of Pareto traffic as can be seen in Table 5.3. Notice that the F-measure score decreases. This decrease in performance of classifiers for Pareto traffic can be explained by the class overlap problem. The noise introduced by the increased burstiness of cross traffic causes instances from nearby classes to reside in overlapping regions in the feature space, which makes it difficult for the classifier to assign instances to correct classes, hence the misclassification error rate increases.

In Fig. 5.13a, we compare the results obtained by implementing the direct probing technique and the NN-based classifier (NNC). Recall that the NNC shows the highest accuracy in Table 5.2. The NNC, which is trained in a network with cross traffic having exponentially distributed packet inter-arrival times, performs almost with no misclassification in the case of no burstiness. In Fig. 5.13b, we see that the SD decreases after applying a median filter to the estimates obtained by the NNC. In Fig. 5.13c and Fig. 5.13d, we show the results for both the classification and regression-based methods using NN and SVM, respectively. While all of them perform well and provide comparable results, the NNR technique has the minimum SD.

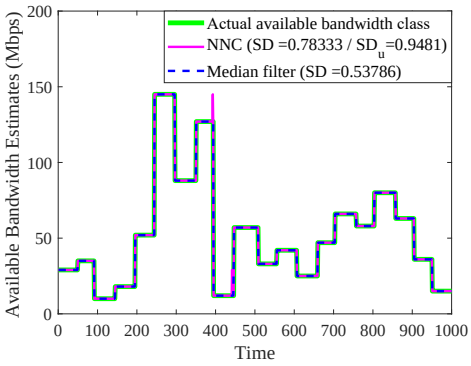
Furthermore, in Fig. 5.14a, we compare the results obtained from the direct probing technique with the SVM-based classifier, which shows the highest accuracy in Table 5.3 in the case of cross traffic with Pareto-distributed packet inter-arrival times. Again, the SVM-based classifier, which is trained in a network with cross traffic having exponentially distributed packet inter-arrival times, performs better than the direct probing technique. In Fig. 5.14b, we show that the estimates are improved after applying the median filter to the estimates of the SVM-based classifier. In Fig. 5.14c and Fig. 5.14d, we compare the classification and regression-based methods again using SVM and NN, respectively. In all techniques, the bias in the estimates increases with the increasing burstiness while the classification methods have less SD. Here, we can easily conclude that the NNR technique is better than the other when there is no burstiness in cross traffic in a network, while utilizing SVM and classification improves the estimation quality.

Table 5.2: Testing performance evaluation in a single tight link in the presence of cross traffic that has a constant bit rate

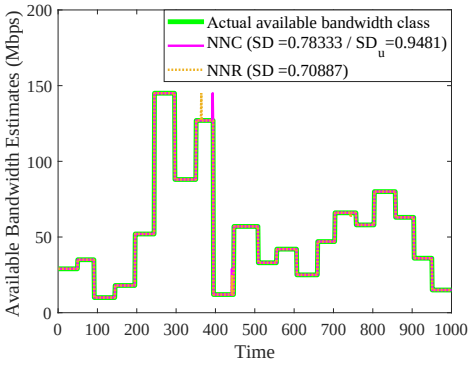
Parameters	Algorithms				
	SVM	k-NN	Bagging	AdaBoost	NN
Acc_{avg}	0.9996	0.9996	0.9991	0.9990	0.9998
$Precision_{\mu}$	0.9960	0.9960	0.9910	0.9900	0.9980
$Recall_{\mu}$	0.9960	0.9960	0.9910	0.9900	0.9980
$F1_{\mu}$	0.9960	0.9960	0.9910	0.9900	0.9980
$Precision_{M}$	0.9964	0.9961	0.9926	0.9916	0.9981
$Recall_{M}$	0.9962	0.9956	0.9898	0.9888	0.9979
$F1_{M}$	0.9963	0.9958	0.9912	0.9902	0.9980



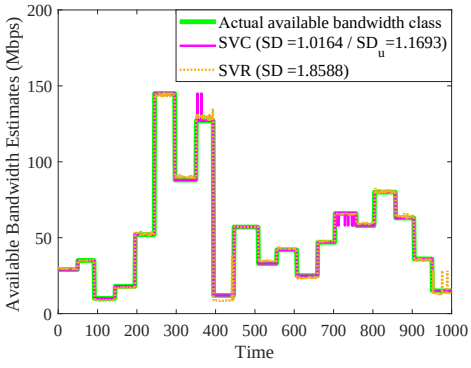
(a) Direct vs NN-based classification



(b) Filtered estimated classes



(c) NN-based classification vs regression

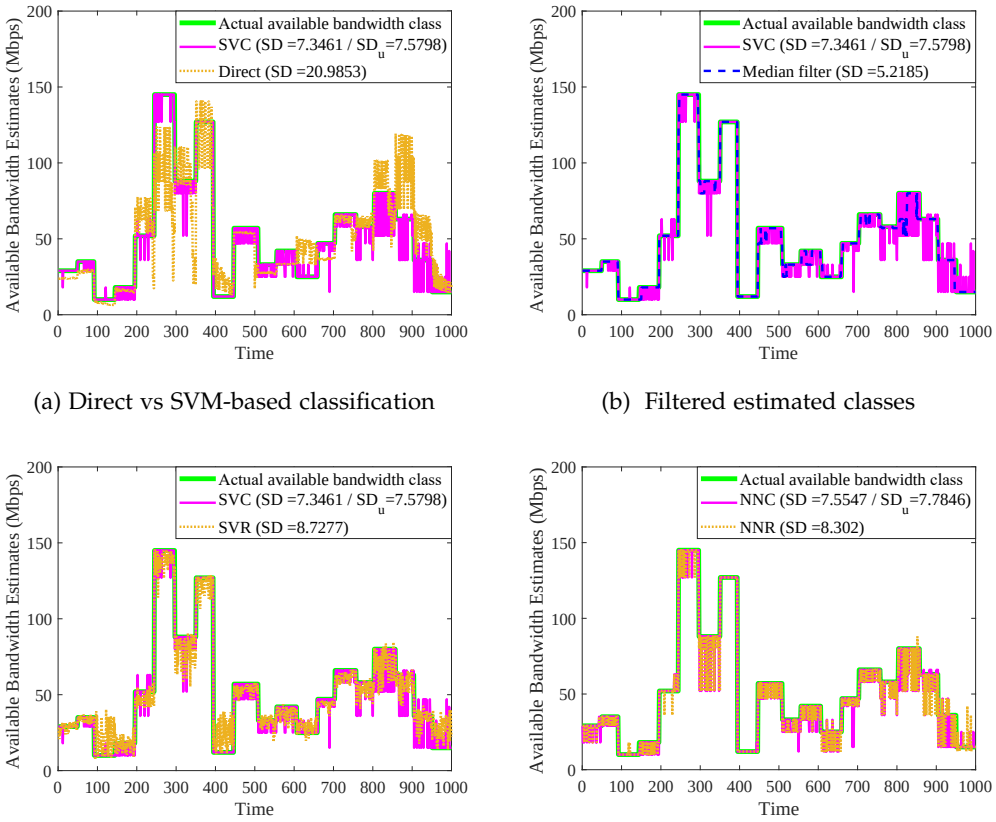


(d) SVM-based classification vs regression

Figure 5.13: Available bandwidth estimates and their SD in the presence of cross traffic that has a constant bit rate using $p = 6$ packet trains in a single tight link network.

Table 5.3: Testing performance evaluation in a single tight link in the presence of cross traffic that has Pareto-distributed packet inter-arrival times

Parameters	Algorithms				
	SVM	k-NN	Bagging	AdaBoost	NN
Acc_{avg}	0.9746	0.9724	0.9734	0.9684	0.9735
$Precision_{\mu}$	0.7460	0.7240	0.7340	0.6840	0.7350
$Recall_{\mu}$	0.7460	0.7240	0.7340	0.6840	0.7350
$F1_{\mu}$	0.7460	0.7240	0.7340	0.6840	0.7350
$Precision_{M}$	0.8018	0.7394	0.7528	0.7051	0.7576
$Recall_{M}$	0.7518	0.7238	0.7347	0.6856	0.7323
$F1_{M}$	0.7760	0.7315	0.7436	0.6952	0.7447



(a) Direct vs SVM-based classification

(b) Filtered estimated classes

(c) SVM-based classification vs regression

(d) NN-based classification vs regression

Figure 5.14: Available bandwidth estimates and their SD in the presence of cross traffic that has Pareto-distributed packet inter-arrival times using $p = 6$ packet trains in a single tight link network.

5.5.2.2 *Multiple Tight Links*

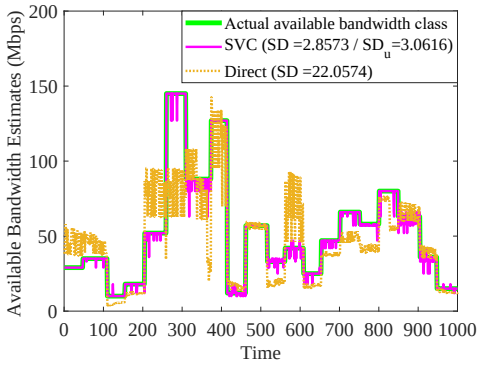
In this section, we address networks with multiple tight links. These networks pose a well-known challenge in available bandwidth estimation because the linear model presented in Fig. 2.5 does not reflect the nature of multiple hop networks completely. Hence, we extend our testbed from a single tight link network to a network with four tight links as shown in Fig. 4.1. In order to generate the (ii) training data set, we send a probe stream consisting of p packet trains through the network with 1 tight link, 2, 3 and 4 tight links consecutively, where each link is exposed to cross traffic with exponentially distributed packet inter-arrival times with an average rate $\lambda = 50$ Mbps. In all the network settings, we set the probe stream as path-persistent and the cross traffic as single hop-persistent. The maximum capacity of each tight link is 100 Mbps. The classification-based method is trained with this additional (ii) training set for multiple tight links along with (i) training set for a single tight link.

In Table 5.4, Table 5.5 and Table 5.6 we show the averaged accuracy, and the micro-averaged and macro-averaged F-measure results for 2-hop, 3-hop and 4-hop network, respectively. Compared to the results of a single tight link in Table 5.1, we notice that the misclassification error rate has increased with an increase in the number of tight links. This is because, in the case of multiple tight links, the probe stream has a constant rate r_{in} with a defined input gap g_{in} only at the first link. For the following links, the input gaps have a random structure as they are the output gaps of the preceding links. At each additional link, the probe stream interacts with new, bursty cross traffic. This introduces more noise to the r_{in} of the following links and makes the classification difficult. Particularly, we need more data for the training of the method in order to capture the randomness of the network with multiple links.

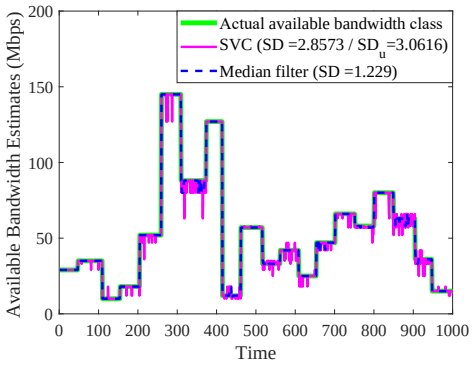
In Fig. 5.15a, Fig. 5.16a and Fig. 5.17a, we compare the results obtained from the direct probing technique with SVC for 2-hop, 3-hop, and 4-hop network, respectively. Though, for both the methods, the variability in estimates increases with increase in the number of links, the results obtained from SVC are still highly accurate as compared to the direct method. Furthermore, after applying the median filter to the of the estimates of the SVM-based estimator, we filter out the noise in the estimates as seen in Fig. 5.15b, Fig. 5.16b and Fig. 5.17b. Fig. 5.15c, Fig. 5.16c and Fig. 5.17c compare SVC and SVR methods, for 2-hop, 3-hop and 4-hop network, respectively. The comparison of NNC and NNR methods for 2-hop, 3-hop and 4-hop network is shown in Fig. 5.15d, Fig. 5.16d and Fig. 5.17d, respectively. Noting that the results are comparably closer to each other, the classification has less SD, and the noise in the estimates increases with the increasing number of links.

Table 5.4: Testing performance evaluation in a 2-hop network in the presence of cross traffic that has exponentially distributed packet inter-arrival times

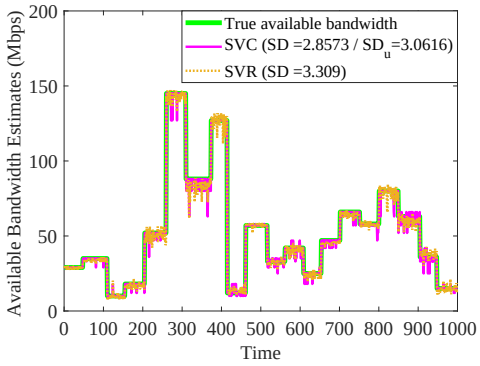
Parameters	Algorithms				
	SVM	k-NN	Bagging	AdaBoost	NN
Acc_{avg}	0.9814	0.9774	0.9811	0.9790	0.9813
$Precision_{\mu}$	0.8140	0.7740	0.8110	0.7900	0.8130
$Recall_{\mu}$	0.8140	0.7740	0.8110	0.7900	0.8130
$F1_{\mu}$	0.8140	0.7740	0.8110	0.7900	0.8130
$Precision_{M}$	0.8265	0.8019	0.8256	0.8000	0.8356
$Recall_{M}$	0.8155	0.7802	0.8179	0.7895	0.8225
$F1_{M}$	0.8210	0.7909	0.8217	0.7947	0.8290



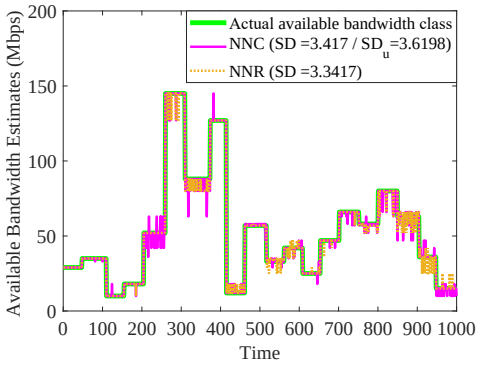
(a) Direct vs SVM-based classification



(b) Filtered estimated classes



(c) SVM-based classification vs regression

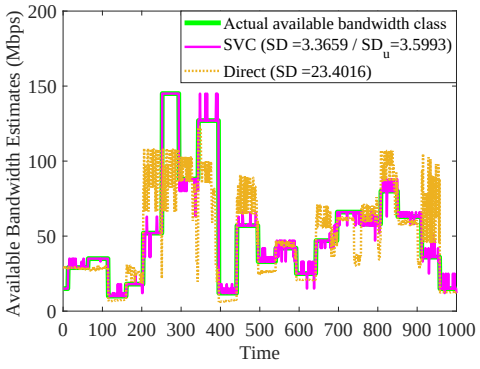


(d) NN-based classification vs regression

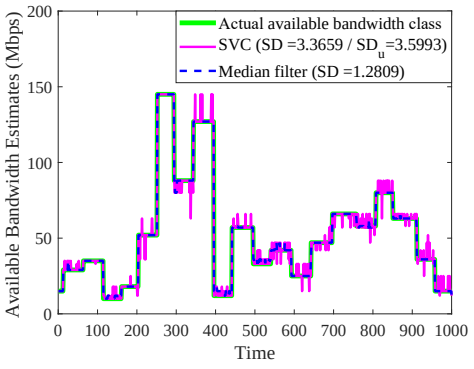
Figure 5.15: Available bandwidth estimates and their SD in the presence of cross traffic that has exponentially distributed packet inter-arrival times using $p = 6$ packet trains in a 2-hop network.

Table 5.5: Testing performance evaluation in a 3-hop network in the presence of cross traffic that has exponentially distributed packet inter-arrival times.

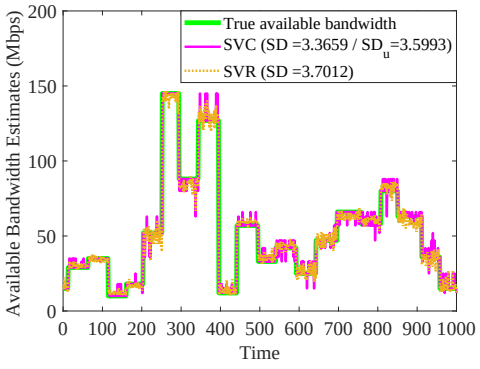
Parameters	Algorithms				
	SVM	k-NN	Bagging	AdaBoost	NN
Acc_{avg}	0.9749	0.9705	0.9727	0.9652	0.9727
$Precision_{\mu}$	0.7490	0.7050	0.7270	0.6520	0.7270
$Recall_{\mu}$	0.7490	0.7050	0.7270	0.6520	0.7270
$F1_{\mu}$	0.7490	0.7050	0.7270	0.6520	0.7270
$Precision_{M}$	0.7571	0.7117	0.7401	0.6648	0.7339
$Recall_{M}$	0.7470	0.7040	0.7264	0.6598	0.7248
$F1_{M}$	0.7520	0.7078	0.7332	0.6623	0.7293



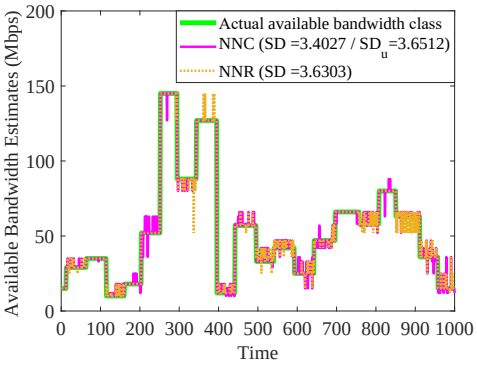
(a) Direct vs SVM-based classification



(b) Filtered estimated classes



(c) SVM-based classification vs regression

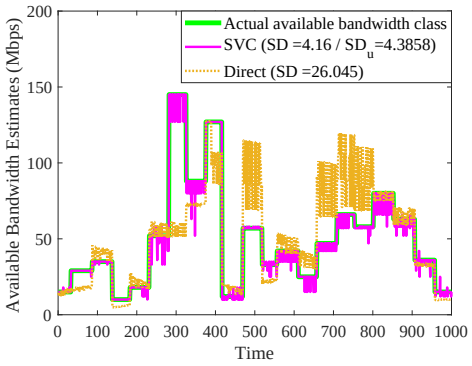


(d) NN-based classification vs regression

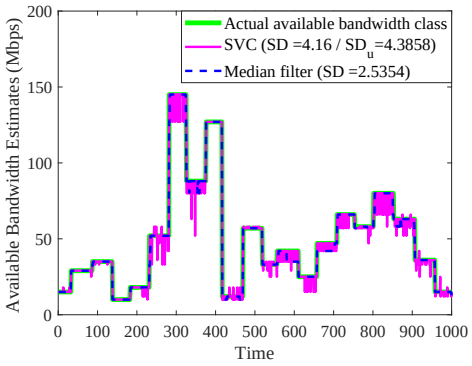
Figure 5.16: Available bandwidth estimates and their SD in the presence of cross traffic that has exponentially distributed packet inter-arrival times using $p = 6$ packet trains in a 3-hop network.

Table 5.6: Testing performance evaluation in a 4-hop network in the presence of cross traffic that has exponentially distributed packet inter-arrival times.

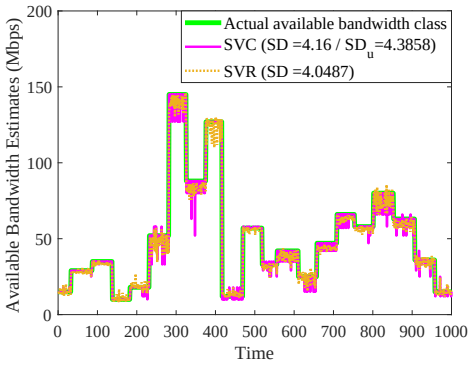
Parameters	Algorithms				
	SVM	k-NN	Bagging	AdaBoost	NN
Acc_{avg}	0.9735	0.9700	0.9701	0.9613	0.9704
$Precision_{\mu}$	0.7350	0.7000	0.7010	0.6130	0.7040
$Recall_{\mu}$	0.7350	0.7000	0.7010	0.6130	0.7040
$F1_{\mu}$	0.7350	0.7000	0.7010	0.6130	0.7040
$Precision_{M}$	0.7639	0.7323	0.7371	0.6457	0.7346
$Recall_{M}$	0.7368	0.7001	0.7112	0.6085	0.7086
$F1_{M}$	0.7501	0.7158	0.7239	0.6265	0.7214



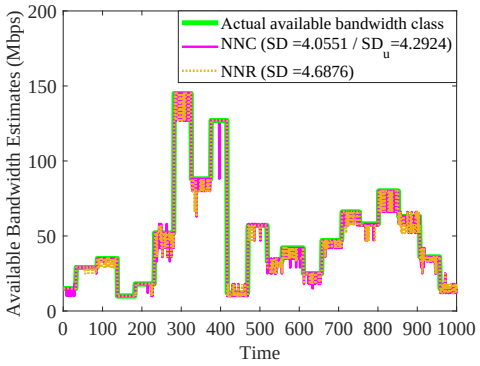
(a) Direct vs SVM-based classification



(b) Filtered estimated classes



(c) SVM-based classification vs regression



(d) NN-based classification vs regression

Figure 5.17: Available bandwidth estimates and their SD in the presence of cross traffic that has exponentially distributed packet inter-arrival times using $p = 6$ packet trains in a 4-hop network.

5.5.3 *Effect of Number of Packet Trains*

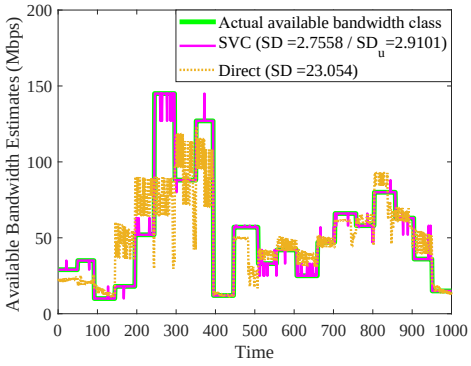
In general, the excessive amount of probe traffic is undesired in networks. In the previous sections, we use a probe traffic with $p = 6$ packet trains to estimate the bandwidth thereby reducing the amount of artificial traffic injected to the network. Now, we investigate how much we can decrease the probe traffic without a significant impact on the performance of the classification-based method.

As we can interpret from the rate response curve in Fig. 2.5, the probe rates that are greater than or equal to the available bandwidth, i.e., $r_{in} \geqslant C(t) - \lambda(t)$, provide the relevant information to estimate the available bandwidth, but meanwhile, cause congestion in the network. Therefore, in order to investigate the effect of the number of packet trains on the performance and harness the congestion risk, we exclude the highest probing rates while setting the number of packet trains to $p=5$ and $p=4$, respectively. Considering again a single link with cross traffic having exponentially distributed packet inter-arrival times, we show the results in Table 5.7 and Table 5.8, when the number of packet trains is five and four, respectively. We can see that the performance decreases with the decreasing number of packet trains. However, the decrease in accuracy is negligible, and the misclassification performance is affected slightly. Noting the trade-off between the amount of injected traffic and the performance loss, we can still use four packet trains if we need to avoid network congestion.

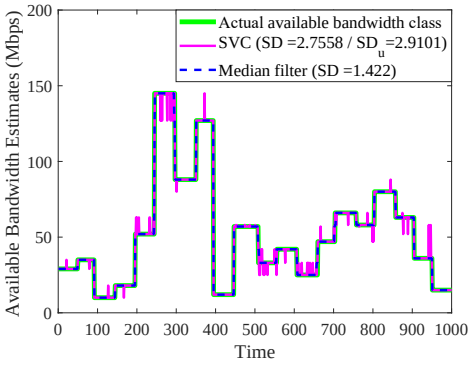
In Fig. 5.18a and Fig. 5.19a, we compare the results obtained by the direct probing technique with the ones by SVC-based method when we have $p = 5$ and $p = 4$, respectively. Despite the noise increase in the estimates with the decrease in the probe traffic, the results obtained by the SVC-based method are still more accurate than the ones by the direct probing technique. Moreover, the direct probing technique has more SD. This is again because when the available bandwidth approaches the capacity, there is no inter-packet strain measurement left on the upward segment of the rate response curve due to omitting the higher probing rates. Furthermore, in Fig. 5.18b and Fig. 5.19b, we show that after applying the median filter to the estimates of the SVC-based method, the estimates become less noisy again. Finally, we compare the SVM-based and NN-based classification and regression techniques in Fig. 5.18c and Fig. 5.19c, and in Fig. 5.18d and Fig. 5.19d, respectively. The results are close to each other. Finally, we note that the noise in the estimates increases with the less packet trains.

Table 5.7: Testing performance evaluation in a single tight link in the presence of cross traffic that has exponentially distributed packet inter-arrival times using $p = 5$ packet trains

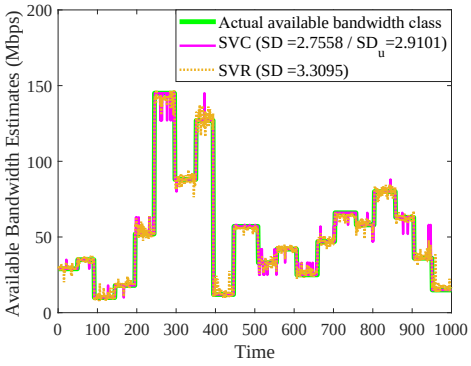
Parameters	Algorithms				
	SVM	k-NN	Bagging	AdaBoost	NN
Acc_{avg}	0.9935	0.9929	0.9934	0.9908	0.9935
$Precision_{\mu}$	0.9350	0.9290	0.9340	0.9080	0.9350
$Recall_{\mu}$	0.9350	0.9290	0.9340	0.9080	0.9350
$F1_{\mu}$	0.9350	0.9290	0.9340	0.9080	0.9350
$Precision_{M}$	0.9354	0.9309	0.9372	0.9107	0.9364
$Recall_{M}$	0.9354	0.9290	0.9349	0.9077	0.9364
$F1_{M}$	0.9354	0.9299	0.9360	0.9092	0.9364



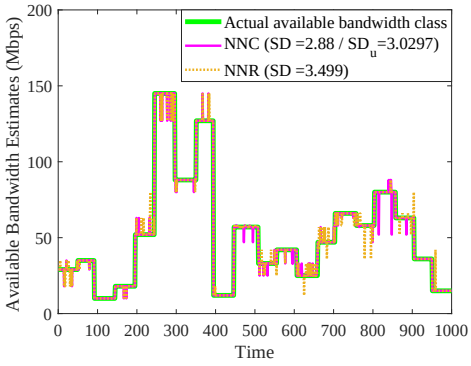
(a) Direct vs SVM-based classification



(b) Filtered estimated classes



(c) SVM-based classification vs regression

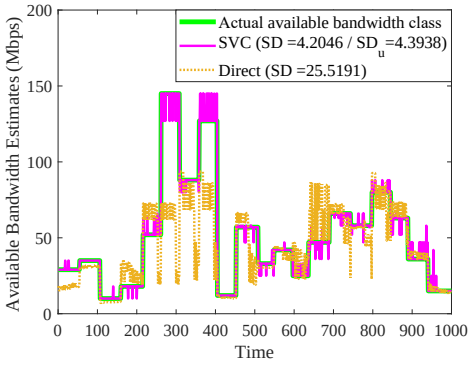


(d) NN-based classification vs regression

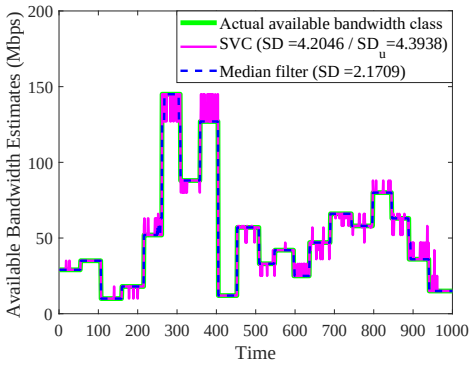
Figure 5.18: Available bandwidth estimates and their SD in the presence of cross traffic that has exponentially distributed packet inter-arrival times using $p = 5$ packet trains in a single tight link network.

Table 5.8: Testing performance evaluation in a single tight link in the presence of cross traffic that has exponentially distributed packet inter-arrival times using $p=4$ packet trains

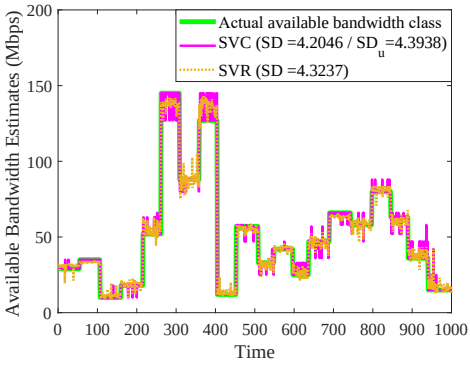
Parameters	Algorithms				
	SVM	k-NN	Bagging	AdaBoost	NN
Acc_{avg}	0.9863	0.9853	0.9857	0.9809	0.9853
$Precision_{\mu}$	0.8630	0.8530	0.8570	0.8090	0.8530
$Recall_{\mu}$	0.8630	0.8530	0.8570	0.8090	0.8530
$F1_{\mu}$	0.8630	0.8530	0.8570	0.8090	0.8530
$Precision_{M}$	0.8611	0.8521	0.8581	0.8153	0.8604
$Recall_{M}$	0.8591	0.8498	0.8571	0.8118	0.8521
$F1_{M}$	0.8601	0.8509	0.8576	0.8135	0.8562



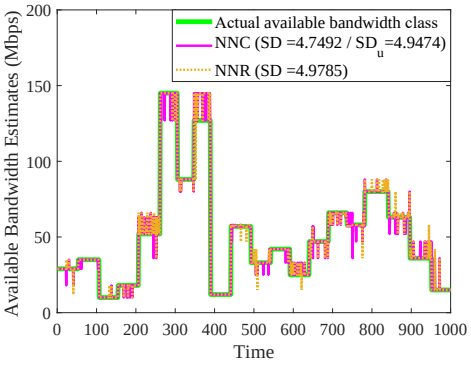
(a) Direct vs SVM-based classification



(b) Filtered estimated classes



(c) SVM-based classification vs regression



(d) NN-based classification vs regression

Figure 5.19: Available bandwidth estimates and their SD in the presence of cross traffic that has exponentially distributed packet inter-arrival times using $p = 4$ packet trains in a single tight link network.

6

PASSIVE AVAILABLE BANDWIDTH ESTIMATION

This chapter aims at understanding the information that short-lived TCP flows provide on the available bandwidth. The results may benefit new TCP versions such as the recent Hybrid Start (HyStart) algorithm [55] and Bottleneck Bandwidth and Round-trip propagation time (BBR) [56]. Furthermore, the rate-adaptive applications like Moving Picture Experts Group (MPEG)-Dynamic Adaptive Streaming over HTTP (DASH) can be benefited from bandwidth estimation, where it is a common practice to use the throughput of a TCP connection as an estimator of the available bandwidth.

We find the shortcomings of TCP throughput as available bandwidth estimator, and further investigate how techniques from active probing can benefit TCP bandwidth estimation. We identify the chaotic, non-packet train pattern of TCP flows, and define a criterion to select traffic samples that bear relevant information. This information is encoded in the form of packet gaps that are explained by the PGM known in bandwidth estimation. We use a regression technique to obtain robust bandwidth estimates from passive measurements of these gaps. The accuracy of the method is evaluated for a variety of relevant parameter settings. We propose a method that can take multiple gaps as well as acknowledgment gaps as input. This extension enables bandwidth estimation using only sender-side measurements of TCP input and acknowledgment packets. Furthermore, to deal with the distorted acknowledgment gaps, we explore the use of machine learning, specifically Neural Network (NN) in passive bandwidth estimation. We also apply the NN under a variety of notoriously difficult conditions that have not been included in the training such as randomly generated networks with the multiple bottleneck links and heavy cross traffic burstiness. We consider single packet losses and investigate how our proposed method may benefit the recent congestion control algorithm: BBR [56]. We provide data sets generated in a controlled network testbed located at Leibniz University Hannover for training and testing of our proposed method. The work in this chapter is based on joint work with Markus Fidler [57], and [58].

The remainder of this chapter is structured as follows. In Sec. 6.1 we discuss the limitations of TCP throughput as available bandwidth estimator. We de-

scribe our experimental setup in Sec. 6.2. In Sec. 6.3, we introduce a method to estimate the available bandwidth from passive measurements. We apply the method to short-lived TCP flows and evaluate the accuracy for a variety of relevant parameters in Sec. 6.4. We also extend the method to use sender-side measurements of acknowledgment gaps in Sec. 6.5. We introduce a NN-based method to estimate the available bandwidth from noise afflicted acknowledgment gaps in Sec. 6.6. We evaluate our proposed method in the presence of packet loss in Sec. 6.8.

6.1 TCP THROUGHPUT AS AVAILABLE BANDWIDTH ESTIMATOR

The Internet relies on congestion control protocols and adaptive applications that adjust their data rate to achieve good performance while avoiding network congestion. DASH is one such application that has emerged as an increasingly popular paradigm for video streaming and is the core technology used by major Internet video providers such as YouTube and Netflix [59]. In DASH, the media content is divided into segments or chunks of a certain duration, typically in the order of a few seconds. The chunks are encoded using multiple profiles, i.e., each chunk is available in different quality levels corresponding to different bandwidth requirements. The encoded chunks are stored on HTTP servers along with a manifest file which lists the available profiles of the chunks. The client downloads the chunks one by one using HTTP GET requests, where it seeks to select the profile that matches the available resources best.

An essential prerequisite for such applications is the estimation of available network resources. On the one hand, TCP protocol was not originally designed to deliver video streaming applications and, on the other hand, DASH itself does not specify the method of how to measure the available bandwidth. A typical approach is to use the average throughput achieved by TCP during the transmission of previous chunks as an estimate of the available bandwidth, e.g., [59, 60]. Formally, given a chunk of N packets each with length l, the average throughput is computed as

$$r_{out} = \frac{(N-1)l}{\sum_{b=1}^{N-1} g_{out}^b}, \tag{6.1}$$

where the output gap is defined as $g_{out}^b = t_{out}^{b+1} - t_{out}^b$ and t_{out}^b is the time of reception of packet b by the receiver. To support basic insights into the relation of TCP throughput and available bandwidth, we evaluate the average throughput of TCP in controlled network experiments. Compared to [9, 37], we use TCP CUBIC to transfer chunks of limited size. CUBIC is a high-speed TCP variant aimed at saturating networks with a large bandwidth-delay-product. In conges-

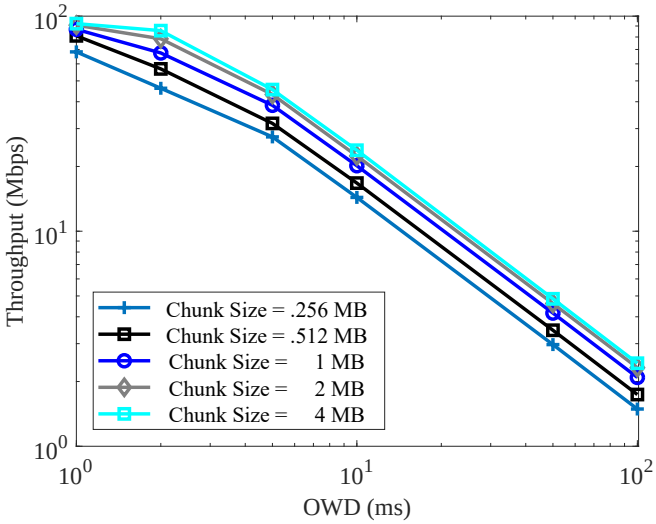


Figure 6.1: Average TCP throughput. The available bandwidth of 100 Mbps is attained only if the OWD is small and the chunk size is large.

tion avoidance, it uses a CUBIC function to increase the CWND independent of the RTT [61].

Fig. 6.1 shows the throughput that is achieved by TCP when transmitting chunks of a fixed size in the range from 256 kB to 4 MB. The network offers an available bandwidth of 100 Mbps and the experiments are conducted for a range of OWD from 1 to 100 ms. Further, details on the network are deferred to Sec. 6.2. We notice that TCP congestion control limits the throughput significantly below the available bandwidth for two reasons: first, the TCP transmission starts in slow start with a small CWND and, even despite the CWND is increased quickly, this initial phase has a considerable effect on the average throughput if chunks are small; secondly, when the OWD is non-negligible, the CWND may never reach the bandwidth-delay-product so that the actual throughput during the transmission of a finite-sized chunk generally remains below the available bandwidth.

The above limitations have motivated us to investigate in-depth: “Do observations of short TCP flows provide sufficient information to infer the available bandwidth?" And if so, “How can the available bandwidth be estimated, e.g., can techniques from active probing be adapted to passive TCP measurements?"

6.2 EXPERIMENTAL SETUP

Before we investigate how to estimate the available bandwidth from passive TCP measurements, we provide a brief overview of our testbed network that is used to obtain the experimental results presented in this chapter. The exper-

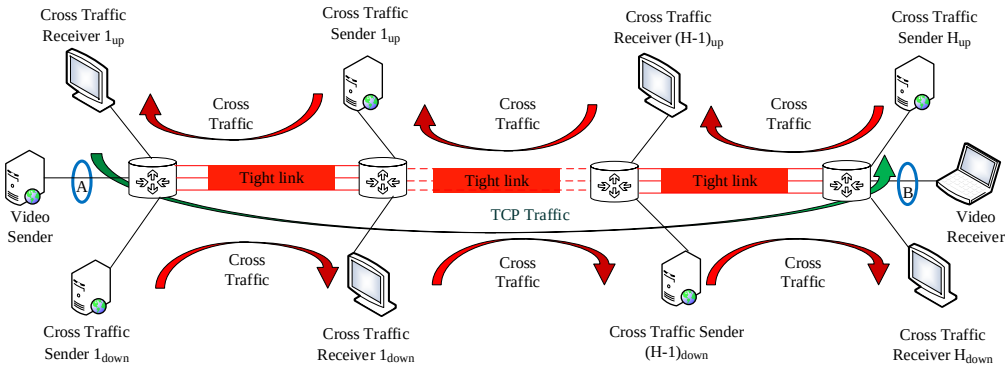


Figure 6.2: Dumbbell topology set up in Emulab with multiple tight links of a capacity of 100 Mbps and a configurable delay and loss rate. Cross traffic in the downstream and upstream directions is single hop-persistent and used to load the tight links to enable controlled bandwidth estimation experiments.

iments are conducted in a controlled network at Leibniz University Hannover that is managed by the Emulab software [44].

We use a dumbbell topology with multiple tight links as shown in Fig. 6.2 and emulate the transmission of DASH video chunks in the presence of downstream as well as upstream cross traffic. The downstream cross traffic is used to load the bottleneck link so that a defined amount of bandwidth remains available, whereas uplink cross traffic interferes with TCP acknowledgments and alters their spacing.

For our experimental purposes, we consider the animated movie named *Sintel* by Blender Foundation. We downloaded the movie from YouTube and obtained the chunk statistics. The modal value of the chunks of about 1 MB is chosen as a reference. To transmit chunks of data of a defined size via TCP and UDP, we use the traffic generators iPerf [62] and RUDE & CRUDE [47], respectively. Cross traffic of different types and intensities is generated using D-ITG [46].

Since the PCs in our Emulab testbed are connected via physical Ethernet links of 1 Gbps and 10 Gbps, respectively, we use the token bucket filter [63] to emulate a 100 Mbps bottleneck link. In addition, a delay node is used at the bottleneck to emulate a wide area link to investigate the effect of different OWDs on the bandwidth estimation. The delay node can also be configured to create packet loss with a defined probability in the downstream and upstream direction. The access links are configured to have 100 Mbps capacity, too. We note that the emulation has limited accuracy and hence contributes additional noise to the measurements.

We disable the segmentation offloading by the NIC using ethtool [64]. Hence, the TCP/IP stack is responsible for segmenting chunks into datagrams of 1500 B size, that is the maximum transmission unit carried by the Ethernet links. In-

cluding Ethernet header and trailer, the packet size is 1514 B. Packet timestamps at the video sender and receiver are generated at points A and B, respectively, using libpcap at the hosts. We also use a specific Endace DAG measurement card to obtain accurate reference timestamps.

6.3 ESTIMATION FROM PASSIVE MEASUREMENTS

As already discussed in Sec. 6.1, TCP throughput is not generally a good estimator of the available bandwidth. Two reasons that we have investigated are non-negligible OWDs and short-lived TCP flows, as caused by small to medium chunk sizes, see Fig. 6.1. Now the following questions arise:

- Is it possible to estimate the available bandwidth in such scenarios where TCP throughput is limited?
- How can the required information be extracted from the rather chaotic traffic patterns of TCP?

Before we investigate the specifics of TCP traffic in Sec. 6.4, we first develop a method for available bandwidth estimation from general passive measurements. We verify the method in controlled experiments.

6.3.1 *Model-based Passive Estimation Method*

We construct a method that is based on the PGM. It uses techniques from direct probing together with a threshold test to select relevant packet gaps. To motivate our design decisions, we start with a discussion of the different options that arise before we give details on the implementation.

Given passive measurements, we have to deal with non-structured traffic that cannot be assumed to take certain patterns, like CWND-sized packet trains as used by HyStart [55]. In order to be able to apply packet train models nevertheless, an option is to filter the sender-side measurement data for clusters of back-to-back packets that exceed a certain threshold. This approach is used in [65], where the specific requirement is that packet trains span several scheduling periods of a cellular network. A drawback of the approach is that a potentially large number of samples that do not pass the threshold test may be discarded. To avoid dependence on any kind of traffic structure, we will work with individual packet gaps. Hence, the PGM applies.

Using the PGM, the task is to estimate the parameters of the gap response curve from passive measurements of g_{in} and g_{out}. The difficulty is due to the fact that we cannot assume evenly spaced g_{in} as achievable by active probing.

For example, we may not have any samples close to the bend of the gap response curve at $l/g_{in} = C - \lambda$, see Fig. 2.3. Consequently, iterative techniques that search for the turning point may not apply, as relevant data may be missing. Further, in the presence of random cross traffic, it has been found that deviations blur the turning point unless long packet trains are used [8, 11].

Techniques from direct probing, on the other hand, require that $g_{in} < l/(C - \lambda)$ where C and λ are unknown. Filtering out all $g_{in} \geqslant l/(C - \lambda)$ beforehand may seem to be an easy task, given that $g_{out} = g_{in}$ in this case, see Eq. (2.2). In practice, g_{out} may, however, be significantly distorted. Important reasons for this are: inaccurate time-stamping, the packet granularity and the randomness of non-fluid cross traffic, and the interaction with cross traffic on links other than the bottleneck link. To determine a threshold g_{in}^{max} up to which g_{in} may safely be used by techniques from direct probing, we adapt a criterion from DietTOPP [33] to the PGM. We identify the minimal input gap denoted g_{in}^{min} in the passive measurement data and extract the corresponding output gap g_{out}^{min}. It can be shown that $g_{out}^{min} < l/(C - \lambda)$, so that we can use $g_{in}^{max} = g_{out}^{min}$ as a threshold to filter out all $g_{in} \geqslant g_{in}^{max}$.

To verify that $g_{out}^{min} < l/(C - \lambda)$, we use Eq. (2.6) to derive the corresponding gap representation

$$g_{out}^{min} = \frac{g_{in}^{min}\lambda + l}{C}, \tag{6.2}$$

where we assume that $g_{in}^{min} < l/(C - \lambda)$. The condition is satisfied if there exist samples g_{in} on the right, upward slope of the gap response curve. Otherwise, if there are no samples in this region, the measurement data does not provide sufficient information to estimate the available bandwidth. The intuition behind Eq. (6.2) is that during g_{in}^{min} an amount of fluid cross traffic of $g_{in}^{min}\lambda$ is accumulated that is transmitted in FIFO order between the two packets that span g_{in}^{min}. The condition $g_{in}^{min} < l/(C - \lambda)$ ensures that the FIFO multiplexer does not become idle during this interval. By insertion of $g_{in}^{min} < l/(C - \lambda)$ into Eq. (6.2), it follows that $g_{out}^{min} < l/(C - \lambda)$.

A lower bound of g_{out}^{min} can be obtained from Eq. (6.2) if we let $g_{in}^{min} \to 0$. It follows that g_{out}^{min} is bounded in the interval $l/C < g_{out}^{min} < l/(C - \lambda)$, i.e., using the threshold $g_{in}^{max} = g_{out}^{min}$ may filter out usable samples that satisfy $g_{in} < l/(C - \lambda)$. We ignore these samples since they are close to the middle part of the gap response curve at $l/g_{in} = C - \lambda$ that has been found to be biased if cross traffic is random [8, 11].

In practice, we cannot rely on a single sample g_{in}^{min} to determine g_{out}^{min}. Instead, we consider a bin of the x smallest gaps g_{in} and compute the average of the

corresponding g_{out} to obtain a robust estimate of g_{out}^{min}. In our experiments, we configure x so that 10% of the gaps are used to estimate g_{out}^{min}.

Once we have selected samples that satisfy $g_{in} < l/(C-\lambda)$, we can apply any technique from direct probing to estimate the available bandwidth. Here, we use linear regression to determine the upward segment of the gap response curve, as this method does not require any specific distribution of the g_{in}. The available bandwidth estimate is determined from Eq. (2.2) as the x-axis intercept where the regression line intersects with the horizontal line at 1, see Fig. 2.3.

6.3.2 *Experimental Verification*

For the first experimental verification of the estimation method, we use (g_{in}, g_{out}) samples that are evenly distributed over the range 5 Mbps $\leqslant l/g_{in} \leqslant$ 100 Mbps. The samples are obtained in our experimental testbed using the tool RUDE & CRUDE that can emit UDP packets with a defined gap. The packet size is $l = 1514$ B on the Ethernet, and we transmit chunks of 1 MB, corresponding to 660 packets. A set of (g_{in}, g_{out}) samples obtained by the transmission of one chunk is shown in Fig. 6.3. In the experiment, CBR cross traffic with rate $\lambda = 50$ Mbps is used.

The cross traffic deviates, however, from the fluid-flow assumption as it uses packets of 1514 B. The effects of the packet granularity become visible as a vertical spread of the g_{out}/g_{in} points in Fig. 6.3. To illustrate an example, we consider $l/g_{in} = 50$ Mbps that corresponds to $g_{in} = 0.24$ ms. The transmission time of a packet at $C = 100$ Mbps is 0.12 ms, so that a cross traffic packet can fit exactly into the gap. This results in $g_{out}/g_{in} = 1$ as also predicted by the fluid model. Cross traffic packets can arrive, however, at arbitrary points in time and if a cross traffic packet arrives right before the first or second packet that constitute g_{in}, it delays this packet by 0.12 ms so that $g_{out}/g_{in} = 0.5$ or 1.5, respectively.

The method for estimation of the available bandwidth from the samples proceeds in two steps. First, it estimates g_{in}^{min} and the corresponding g_{out}^{min} based on the 10% of the samples with the smallest g_{in}. Using $g_{in}^{max} = g_{out}^{min}$ as a threshold for g_{in}, only the samples with $l/g_{in} > l/g_{in}^{max} = 70$ Mbps, that are marked blue in Fig. 6.3, are used in the second step to perform the linear regression. The regression line is shown as a thick blue line. Extending this line until it intersects with the horizontal line at 1, reveals an available bandwidth estimate of 50 Mbps. Further, it follows from Eq. (2.2) that the slope of the regression line provides an estimate of $1/C$. In Fig. 6.3, the slope is approximately 0.01 corresponding to $C = 100$ Mbps.

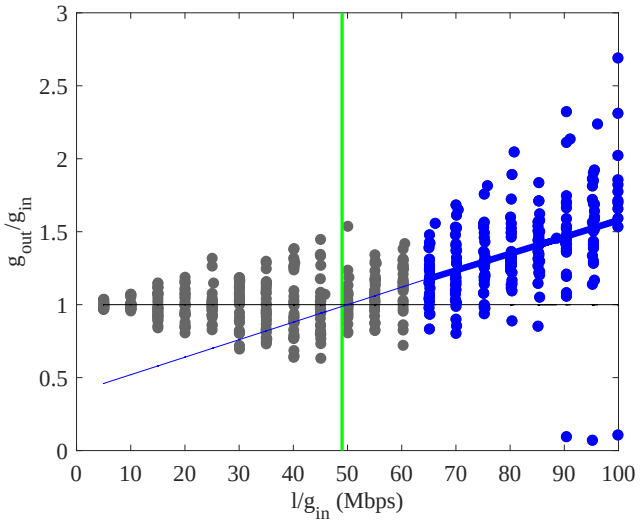


Figure 6.3: (g_{in}, g_{out}) samples obtained by the transmission of a chunk of 1 MB via UDP. The sender varies g_{in} so that the samples are evenly distributed. Samples that are marked blue are used to determine the regression line. The intersection of the regression line with the horizontal line at 1 marks the available bandwidth of 50 Mbps.

We repeated the above experiment with different cross traffic intensities of $\lambda \in \{0, 25, 20, 75\}$ Mbps. For each case, we conducted 100 repeated measurements. We report the mean value in Fig. 6.4. The available bandwidth estimates closely match the ground truth that is marked in the figure by a green line. For comparison, we also include the throughput that is achieved by a TCP sender and a UDP sender, respectively, that transmit the same amount of data of 1 MB. Clearly, the TCP throughput underestimates the available bandwidth, as soon as more than 20 Mbps are available. To understand this effect, we note that the testbed was configured to have an OWD of 10 ms. As a consequence, the TCP throughput is limited by the CWND. For further details, see the discussion of Fig. 6.1. The UDP throughput, on the other hand, overestimates the available bandwidth. This is due to the fact that a greedy UDP sender can preempt the cross traffic at a FIFO multiplexer and monopolize the link. The effect is expressed by Eq. (2.6). Given UDP traffic is injected at line rate $r_{in} = C$, it achieves a throughput of $r_{out} = C^2/(C + \lambda)$ that equates to $\{100, 80, 67, 57\}$ Mbps for the given λ. Similar values are observed in the measurement results shown in Fig. 6.4.

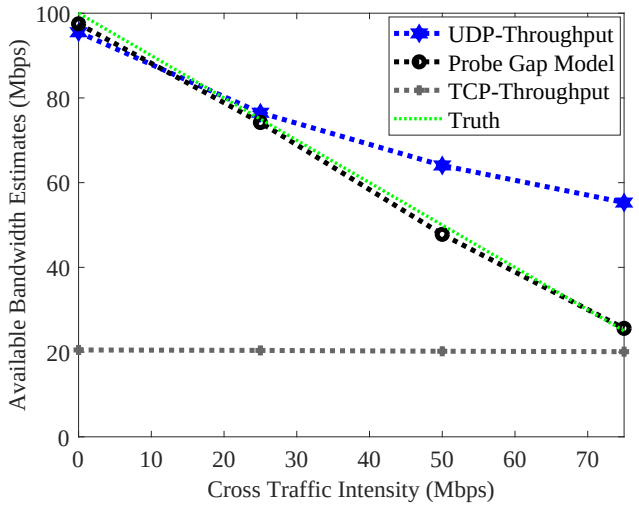


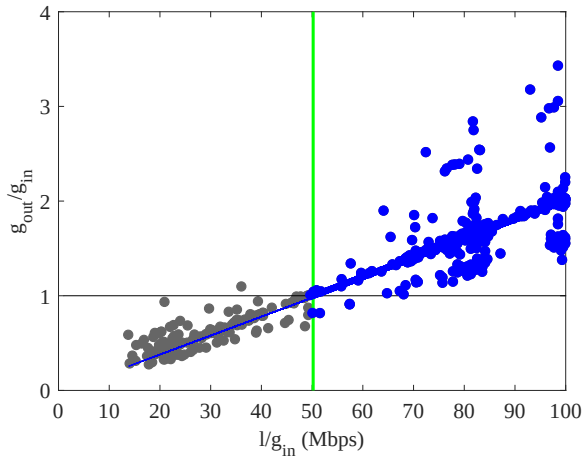
Figure 6.4: Available bandwidth estimates for varying cross traffic rates. The estimates obtained from UDP (g_{in}, g_{out}) measurements closely match the ground truth. The TCP throughput underestimates the available bandwidth, mainly due to the OWD of 10 ms and the limited chunk size of 1 MB. The UDP throughput overestimates the available bandwidth since a greedy UDP sender can preempt cross traffic at a FIFO multiplexer.

6.4 ESTIMATION FROM TCP MEASUREMENTS

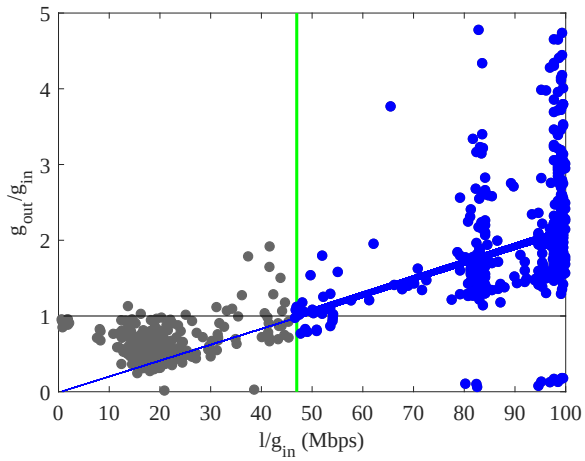
In this section, we investigate the (g_{in}, g_{out}) characteristics of passive TCP measurements and evaluate the available bandwidth estimates that can be obtained thereof. Further, TCP offers a unique opportunity to estimate the available bandwidth from sender-side measurements only, using the feedback that is provided by the spacing of the acknowledgments. We extend the estimation method to include multiple packet gaps as well as acknowledgment gaps. For these, we also develop a technique that deals with packet loss.

6.4.1 *TCP (g_{in}, g_{out}) Characteristics*

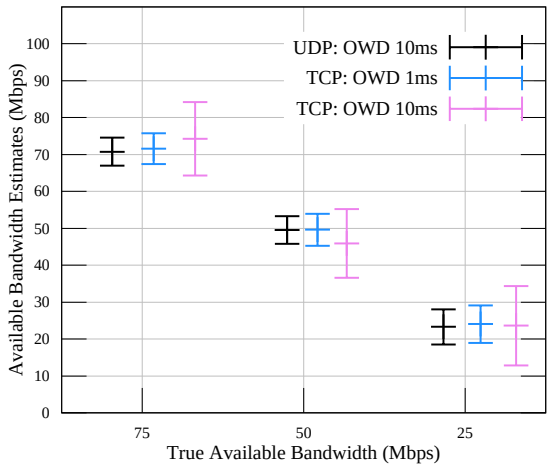
The (g_{in}, g_{out}) characteristics of TCP traffic are largely affected by TCP congestion control and related parameters such as the OWD. Since the input gap is not fixed as in the case of active probing, we extend the notation by superscript b and write $g_{in}^b = t_{in}^{b+1} - t_{in}^b$ whenever we refer to a specific input gap. Above, t_{in}^b is the send timestamp of packet b. In Fig. 6.5, we show two characteristic sets of (g_{in}, g_{out}) samples obtained from TCP traffic. The OWD is 1 ms in Fig. 6.5a and 10 ms in Fig. 6.5b. The remaining parameters are as in Fig. 6.3. For an OWD of 1 ms, the values of g_{in} are spread more or less evenly over a wide range, similar to Fig. 6.3. If the OWD is increased to 10 ms, we observe, however, a clustering of the g_{in} values. Roughly three clusters are formed: in the left part,



(a) OWD = 1 ms, λ = 50 Mbps



(b) OWD = 10 ms, λ = 50 Mbps



(c) Bandwidth Estimates

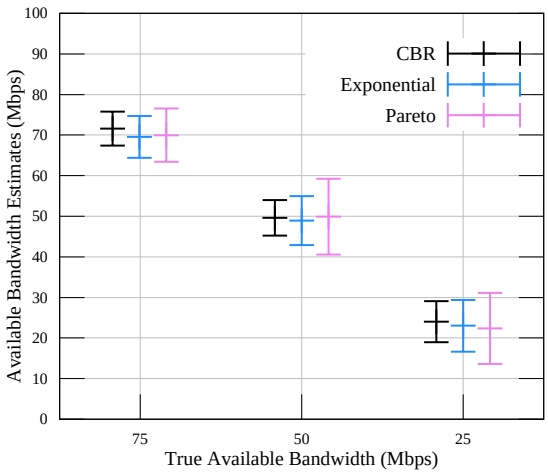
Figure 6.5: (g_{in}, g_{out}) samples obtained from passive TCP measurements and bandwidth estimates obtained thereof. The values of g_{in} are a result of TCP congestion control and depend on network parameters such as the OWD. With increasing OWD a clustering of g_{in} samples is observed and the variability of bandwidth estimates increases.

large g_{in} in the range of up to two OWD occur if the sender has to wait for acknowledgments after transmitting a full CWND; in the middle part, the self-clocking of TCP by the acknowledgments causes g_{in} that correspond roughly to the available bandwidth; and in the right part, back-to-back packets can be found that are triggered, e.g., by cumulative acknowledgments. Bandwidth estimates for CBR cross traffic in the range $\lambda \in \{0, 25, 20, 75\}$ Mbps are summarized in Fig. 6.5c, where we show mean and the standard deviation obtained from 100 repeated measurements each. As before the capacity is $C = 100$ Mbps and the chunk size 1 MB. We compare the case of TCP measurements with an OWD of 1 and 10 ms, respectively, with the UDP measurements presented in Fig. 6.4, before. The results confirm that it is possible to estimate the available bandwidth from passive TCP measurements using the PGM. Moreover, we are able to obtain rational available bandwidth estimates in those scenarios of non-negligible OWDs where TCP throughput as bandwidth estimator is limited. While in the case of small OWDs, TCP and UDP measurements perform comparably, the bandwidth estimates show more variability as well as a certain underestimation if the intensity of the cross traffic is low and the OWD is increased. In the case of low cross traffic intensity, fewer samples pass the threshold test and contribute to the regression line.

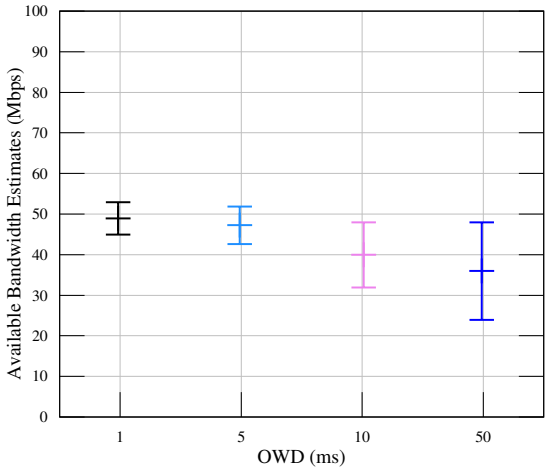
6.4.2 *Parameter Evaluation*

We proceed with an evaluation of the effects of relevant parameters, including the intensity and distribution of cross traffic, the OWD, and the chunk size, on the quality of bandwidth estimates that are obtained from passive TCP measurements. We use cross traffic of different burstiness: CBR as assumed by the PGM; a moderate burstiness due to exponential inter-arrival times; and a strong burstiness due to Pareto inter-arrival times with infinite variance, caused by a shape parameter of $\alpha_s = 1.5$. The packet size is $l = 1514$ B and the average rate of the cross traffic is $\lambda \in \{25, 50, 75\}$ Mbps. The chunk size is 1 MB, the capacity $C = 100$ Mbps, and the OWD is 1 ms. Fig. 6.6a shows the mean and the standard deviation of 100 repeated experiments. We notice that the mean of the estimates corresponds well with the true available bandwidth, regardless of the type of cross traffic. Effects of the cross traffic can be observed in the variability of the estimates that increases if the burstiness is increased.

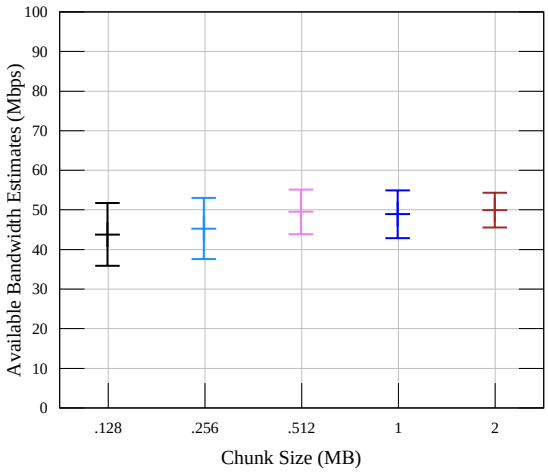
We evaluate the impact of the OWD in a wide range of $\{1, 5, 10, 50\}$ ms for exponential cross traffic with average rate $\lambda = 50$ Mbps in Fig. 6.6b. The results quantify the effects of the OWD on the (g_{in}, g_{out}) characteristics that we observed already in Fig. 6.5. The quality of the bandwidth estimates obtained from passive TCP measurements decreases if the OWD is increased. We note



(a) OWD = 1 ms, chunk size=1 MB



(b) $\lambda = 50$ Mbps exponential, chunk size = 1 MB



(c) $\lambda = 50$ Mbps exponential, OWD = 1 ms

Figure 6.6: Parameter evaluation. Bandwidth estimates for (a) different types of cross traffic burstiness, (b) OWDs, and (c) chunk sizes. The quality of the estimates is good for small to medium OWDs and improves with the chunk size. The burstiness of cross traffic mostly influences the variability of the estimates.

that this effect is specific to TCP congestion control. A fixed increase of the OWD will not alter the (g_{in}, g_{out}) characteristics of UDP traffic.

The impact of the chunk size that determines the number of samples that are obtained for estimation of the available bandwidth, is evaluated in Fig. 6.6c. The cross traffic is exponential with $\lambda = 50$ Mbps and the OWD is 1 ms. Clearly, the quality of the estimates and specifically, the variance of the estimates improves significantly if more samples are available. Moreover, as the chunk size is increased, the quality of the samples changes, too. This is due to the growth of the CWND during the course of the transmission that causes less stalling. Considering the case of a chunk size of 128 kB that corresponds to about 85 packets, we conclude that it is challenging to obtain an estimate of the available bandwidth already during the slow start phase, as HyStart does.

If the assumptions of the fluid model do not hold, e.g., in the case of random cross traffic, the regression technique may occasionally fail. We filter out bandwidth estimates that can be classified as infeasible. This is the case if the slope of the regression line is so small that the intersection with 1 is on the negative $1/g_{in}$ axis in Fig. 2.3, implying the contradiction $A < 0$, or if the slope of the regression line is negative, implying $C < 0$.

6.5 ESTIMATION FROM SENDER-SIDE TCP MEASUREMENTS

TCP offers the option to perform the estimation based only on sender-side measurements of data and acknowledgment packets, i.e., no specific cooperation of the receiver is required. This feature is used, for example, by TCP HyStart. In this section, we will advance the PGM to use acknowledgment gaps. Two aspects have to be considered: TCP uses delayed acknowledgments and typically only every other packet is acknowledged, i.e., the acknowledgment process effectively performs a sub-sampling; and secondly, the cross traffic in the reverse path may alter the spacing of acknowledgments and hence increase the measurement noise. In order to evaluate the impact of the two aspects one at a time, we first investigate the sub-sampling only in Sec. 6.5.1 followed by employing the use of a neural network to deal with measurement noise in acknowledgments gaps in Sec. 6.6.

6.5.1 *Acknowledgment Gaps*

In this section, we address the sub-sampling effect introduced by acknowledgments. For this purpose, we define a multi-gap as $g_{out}^{b,c} = \sum_{i=b}^{c-1} g_{out}^{i} = t_{out}^{c} - t_{out}^{b}$ and $g_{in}^{b,c}$ accordingly. Subsequently, we make the transition from multi-gaps to the corresponding ack-gaps denoted $g_{ack}^{b,c}$. Fig. 6.7 illustrates the concepts of

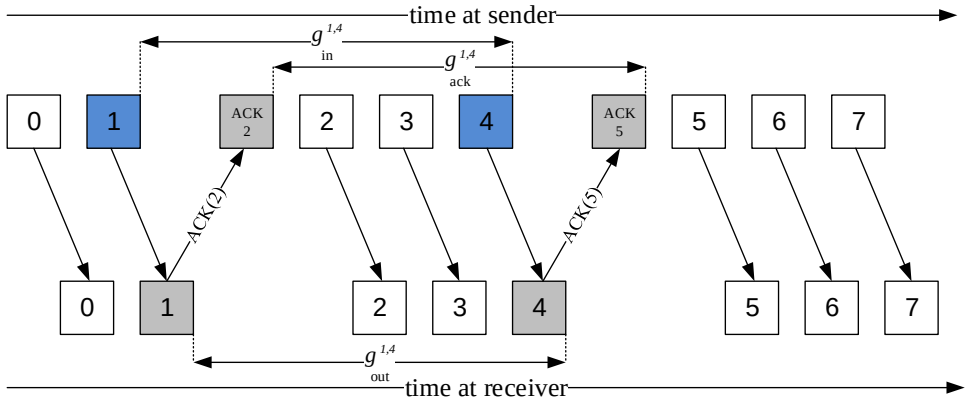


Figure 6.7: Multi-gap and ack-gap models. The ack-gap enables available bandwidth estimation using sender-side measurements only.

multi-gap and ack-gap. The packets are numbered by i and Acknowledgment (ACK) i denotes a cumulative acknowledgment of all packets up to and including packet $i-1$.

We note that combining several gaps to form a multi-gap does not imply constant rate packet train models, since the individual input gaps from passive measurements are random. In fact, the derivation of a multi-gap response curve by repeated application of Eq. (2.2) requires a condition for each individual input gap. Considering only the relevant, upward segment of the gap response curve, we use Eq. (2.2) to derive

$$\frac{g_{out}^{b,c}}{g_{in}^{b,c}} = \frac{\lambda}{C} + \frac{(c-b-1)l}{C g_{in}^{b,c}}, \tag{6.3}$$

if $l/g_{in}^{i} > C-\lambda$ for all $i \in \{b, c-1\}$. The multi-gap response curve shows the same characteristic slope as the gap response curve with one difference: the average input gap $\overline{g}_{in}^{b,c} = g_{in}^{b,c}/(c-b-1)$ and average output gap $\overline{g}_{out}^{b,c} = g_{out}^{b,c}/(c-b-1)$ take the place of g_{in} and g_{out}, respectively.

In order to estimate the available bandwidth from the multi-gap response curve, we use the method defined in Sec. 6.3.1. A slight modification is required to identify samples that satisfy the condition of Eq. (6.3). Given the average input gaps $\overline{g}_{in}^{b,c}$ we find the minimal average input gap $\overline{g}_{in}^{min}$ and the corresponding average output gap $\overline{g}_{out}^{min}$. We select $g_{in}^{max} = \overline{g}_{out}^{min}$ as a threshold for g_{in}^{i} to test that $g_{in}^{i} < g_{in}^{max}$.

To verify that $\overline{g}_{out}^{min}$ is a valid threshold with respect to Eq. (6.3), i.e., $\overline{g}_{out}^{min} < l/(C-\lambda)$, we apply Eq. (6.2) repeatedly to obtain

$$\overline{g}_{out}^{min} = \frac{\overline{g}_{in}^{min}\lambda + l}{C}, \tag{6.4}$$

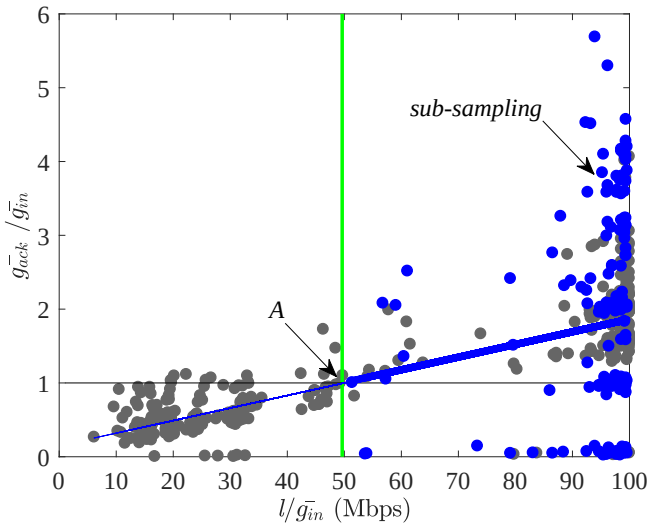


Figure 6.8: Gap response curve. The acknowledgment process effectively performs a sub-sampling and the cross traffic in the reverse path may alter the spacing of acknowledgments as can be seen by $\overline{g}_{ack}/\overline{g}_{in} < 1$.

where we assumed that all g^i_{in}, that are part of $\overline{g}^{min}_{in}$, satisfy $g^i_{in} < l/(C-\lambda)$. By insertion of $\overline{g}^{min}_{in} < l/(C-\lambda)$ it follows that $\overline{g}^{min}_{out} < l/(C-\lambda)$.

The estimation method applies in the same way if acknowledgments are used to avoid receiver-side measurements. In this case, $g^{b,c}_{ack}$ takes the place of $g^{b,c}_{out}$. Fig. 6.8 shows the estimate obtained from 1 MB chunk size in the presence of exponential random cross traffic of average rate $\lambda = 50$ Mbps in the tight link of capacity $C = 100$ Mbps using ack-gap measurements. However, acknowledgment process effectively performs a sub-sampling and random cross traffic in the reverse path distorts the acknowledgment gaps as can be seen by $\overline{g}_{ack}/\overline{g}_{in} < 1$.

We present bandwidth estimates obtained from multi-gap and ack-gap measurements compared to the use of individual gaps in Fig. 6.9. The cross traffic is exponential with average rate $\lambda \in \{25, 50, 75\}$ Mbps. Cross traffic is generated both in the downstream and upstream direction. The chunk size is 1 MB, the capacity $C = 100$ Mbps, and the OWD is 1 ms. The results of 100 repeated measurements are shown. In all cases, the mean of the estimates closely matches the true available bandwidth. The variability is, however, increased if multi-gaps or ack-gaps are used. The reason is due to a smaller number of samples that pass the threshold test. Compared to the multi-gap results, we do not notice a significant change of accuracy when ack-gaps are used.

We note that the estimation may be enhanced using a weighted regression that takes the number of individual gaps that are comprised of a multi-gap into account. We did not use this option since TCP typically acknowledges every

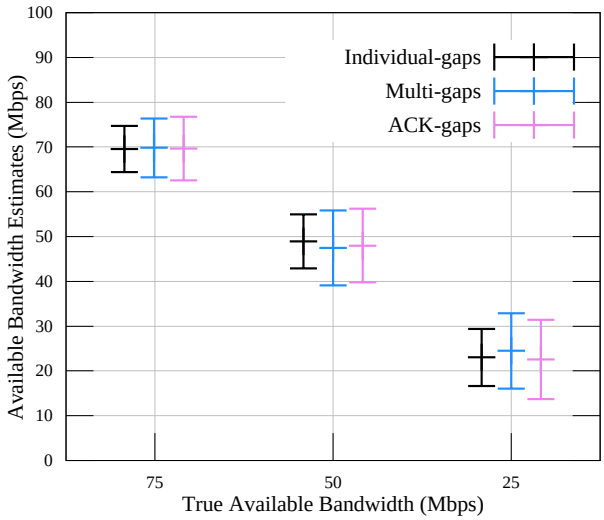


Figure 6.9: Available bandwidth estimates obtained from individual gaps, multi-gaps, and ack-gaps, respectively. The use of multi-gaps and ack-gaps results in increased variability of the estimates. The estimates are reasonable in all three cases.

other packet so that most of the multi-gaps or ack-gaps are of the same size, i.e., they comprise two individual gaps.

6.6 NEURAL NETWORK-BASED METHOD

In this section, we consider the second aspect of performing the estimation using only sender-side measurements of data and acknowledgment packets that the cross traffic in the reverse path may alter the spacing of acknowledgments and hence increase the measurement noise. To estimate the available bandwidth from distorted acknowledgment gaps g_{ack} we present our neural network-based implementation. We describe the training data sets and show a comparison of available bandwidth estimates for a range of different network parameters.

6.6.1 *Data-binning Implementation*

We propose a NN-based method that takes a p-dimensional feature vector of the relative probability of samples of $(\bar{g}_{ack}, \bar{g}_{in})$ as input. Since we cannot control the range of samples of $(\bar{g}_{ack}, \bar{g}_{in})$ that depends upon the OWDs, burstiness and intensity of cross traffic, packet loss and multiple tight links, we use Min-Max normalization to scale the range of samples to $[0, 1]$. We further use a data-binning approach where we partition scaled samples of $l/\bar{g}_{in}$ and $\bar{g}_{ack}/\bar{g}_{in}$ into bins of equal-width. We set bin width as $1/N_b$, where N_b are number of

bins. Since the underlying density of input and acknowledgment gaps is usually unknown, selecting the number of bins is not a trivial task. Therefore, for our approach to avoid bins with zero samples, we choose the number of bins following Rice rule [66] as $N_b = 2N_s^{1/3}$, where N_s is the total number of samples of $(\bar{g}_{ack}, \bar{g}_{in})$. Further to obtain integer number of bins, we apply ceiling function to N_b, i.e., $\lceil N_b \rceil$. The total number of samples N_s, however, also varies depending upon network scenarios in which TCP measurements are taken, e.g., packet loss results in fewer samples of $(\bar{g}_{ack}, \bar{g}_{in})$ as compared to lossless system. Hence, to have consistent number of bins independent of network configurations, N_s is chosen according to the smallest number of $(\bar{g}_{ack}, \bar{g}_{in})$ samples obtained from experiments done to generate data sets for NN. We calculate the relative probability of occurrence of samples in each bin as a ratio of number of samples in each bin and total number of $(\bar{g}_{ack}, \bar{g}_{in})$ samples. To provide input to the neural network, we reshape a matrix representing the relative probabilities of samples into a p-dimensional vector.

6.6.2 *Training Data: Exponential Cross Traffic, Single Tight Link*

We generate different data sets for training and evaluation by setting TCP congestion control protocols as CUBIC and BBR in a controlled network testbed as explained in Sec. 6.2 and OWD of 1 ms. In data set (i) the capacity of tight link is $C = 100$ Mbps and TCP protocol is CUBIC. The capacity of the access links is also configured to have $C = 100$ Mbps. The exponential cross traffic with an average rate $\lambda \in \{25, 50, 75\}$ Mbps is used to generate different available bandwidths. The TCP chunk size is 1 MB. For each configuration, 100 repeated experiments are performed. For training of the neural network, we first implement an autoencoder for each layer separately and then fine-tune the network using SCG. Given a regression network, we optimize the L2-error requiring approximately 1000 epochs until convergence is achieved. Due to the limited amount of training data (300 experiments overall in the first training data set), the shallow network with a small number of hidden neurons allows training without much over-fitting. Both the methods generate available bandwidth estimates from the same measurement data.

6.6.3 *Evaluation: Exponential Cross Traffic, Single Tight Link*

We train the neural network using the (i) training data set and generate additional data sets for testing using the same network configuration as the (i) training data set, i.e., using exponential cross traffic $\lambda \in \{25, 50, 75\}$ Mbps at a single tight link of 100 Mbps capacity. The testing results of the neural network-

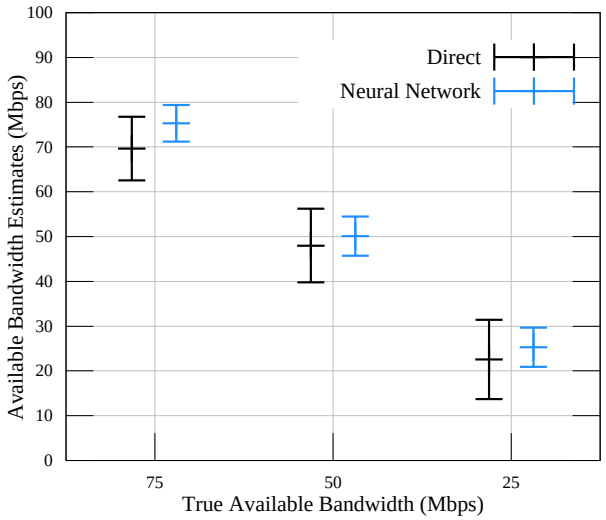


Figure 6.10: Available bandwidth estimates obtained from ack-gaps for different exponential cross traffic rates that have been included in the training data set.

based method compared to the results of the direct method are summarized in Fig. 6.10. We show the average of 100 available bandwidth estimates with error bars depicting the standard deviation of the estimates.

The variability of the available bandwidth estimates of the direct method is comparably large, and the average underestimates the true available bandwidth. Though variability of estimates could be due to a number of reasons as explained in the previous chapters, particularly in this case, the exponential cross traffic deviates from the fluid model and causes random fluctuations of the measurements of g_{ack}. The neural network-based method improves bandwidth estimates significantly. The average matches the true available bandwidth, and the variability is low. The good performance of the neural network is not unexpected as it has been trained for the same network parameters.

6.6.4 *Network Parameter Variation beyond the Training Data*

We investigate the sensitivity of the neural network with respect to a variation of network parameters that differ substantially from the training data set. We consider cross traffic with high burstiness and the networks with multiple tight links as in chapter 4. In this section, we evaluate the variability of cross traffic. Multiple tight links are discussed in the following section.

6.6.4.1 *Burstiness of Cross Traffic*

To evaluate how the neural network-based method performs in the presence of cross traffic with an unknown burstiness, we consider three different types of

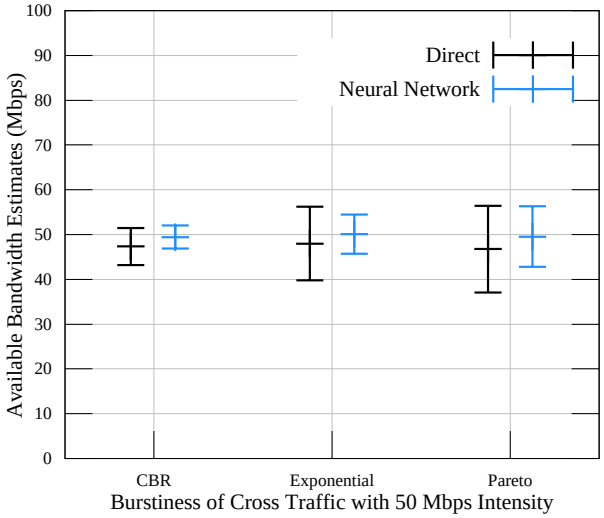


Figure 6.11: Bandwidth estimates for different types of cross traffic burstiness. An increase of the burstiness causes a higher variability of the bandwidth estimates.

cross traffic: CBR that has no burstiness as assumed by the probe rate model, moderate burstiness due to exponential packet inter-arrival times and heavy burstiness due to Pareto inter-arrival times with infinite variance caused by a shape parameter of $\alpha_s = 1.5$. The average rate of cross traffic is $\lambda = 50$ Mbps in all cases. As before, the tight link and access links capacities are $C = 100$ Mbps.

Fig. 6.11 shows the mean and the standard deviation of 100 repeated experiments using the direct and the neural network-based method. The average of the estimates shows a slight underestimation bias for direct method due to the queueing effect caused by bursty traffic at the tight link even if the TCP flow rate is below the average available bandwidth, i.e., if $l/g_{in} < C - \lambda$. More pronounced is the effect of the cross traffic burstiness on the standard deviation of the bandwidth estimates. While for CBR cross traffic, the estimates are close to deterministic, whereas the variability of the estimates increases significantly if the cross traffic is bursty. The neural network that has been trained for exponential cross traffic only, performs almost perfectly in the case of CBR cross traffic and shows good results with less variability compared to the direct technique also for the case of Pareto cross traffic.

6.6.4.2 *One Way Delays*

We evaluate the impact of OWDs in a wide range of $\{1, 5, 10\}$ ms on the performance of neural network for exponential cross traffic with average rate $\lambda =$ 50 Mbps. As before, the tight link and access links capacities are $C = 100$ Mbps.

Fig. 6.11 shows the mean and the standard deviation of 100 repeated experiments using the direct and the neural network-based method. The neural

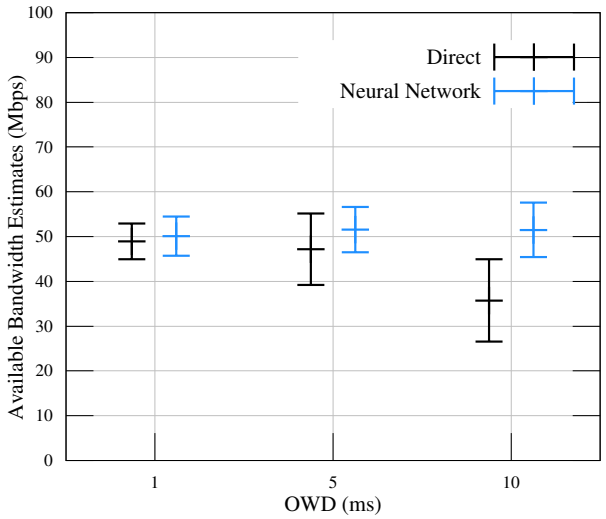
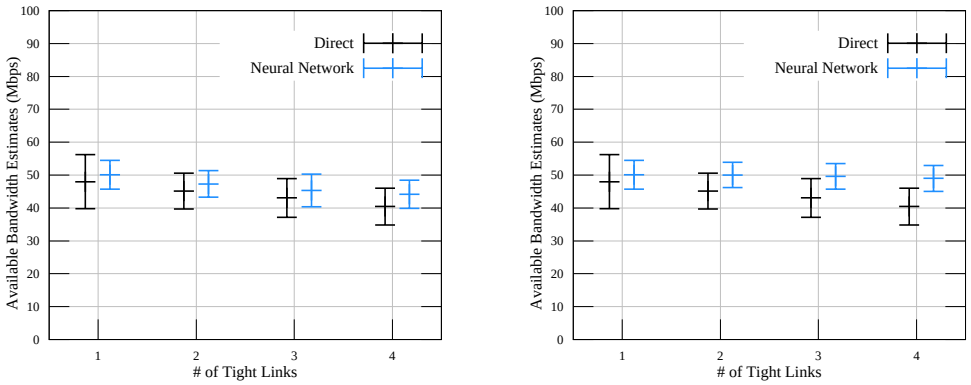


Figure 6.12: Bandwidth estimates for different OWDs. An increase of the OWD causes a higher variability of the bandwidth estimates.

network that has been trained for OWD of 1 ms only, performs almost perfectly in the case of higher OWDs and shows good results with less variability compared to the direct technique. In direct method, due to TCP congestion control, the quality of the bandwidth estimates obtained from passive sender-side TCP measurements decreases if the OWD is increased.

6.7 MULTIPLE TIGHT LINKS

In the case of multiple tight links, the chaotic input gaps g_{in} of TCP get further distorted in the following links as they are the output gaps of the preceding links. At each additional link, the TCP stream interacts with new, bursty cross traffic and its output rate reduces. This results in underestimation of the available bandwidth in multi-hop networks [8–10, 12]. To test the neural network with multiple tight links, we extend our network from single-hop to multi-hop as shown in Fig. 6.2. The path-persistent TCP flow experiences single hop-persistent exponential cross traffic with average rate $\lambda = 50$ Mbps in upstream and downstream while traversing multiple tight links of capacity $C = 100$ Mbps. The capacity of the access links is 100 Mbps. In Fig. 6.13a, we show the results from 100 repeated measurements for networks with 1 up to 4 tight links. The direct as well as the neural network-based methods, underestimate the available bandwidth with an increasing number of tight links. The reason for underestimation in the case of model-based techniques is the cross traffic burstiness which potentially reduces the output probe rate at each of the tight links. Though the estimates of our neural network show the least variability, the neu-



(a) Neural network trained for a single tight link

(b) Neural network trained for multiple tight links

Figure 6.13: The training of the neural network for multiple tight links improves the bandwidth estimates significantly.

ral network underestimates the available bandwidth. This is not surprising as it is trained only for a single tight link.

6.7.0.1 *Training for Multiple Tight Links*

Since the neural network that is trained only for a single tight link, underestimates the available bandwidth in the case of multiple tight links, we consider training the neural network for the latter. To generate the training data, a chunk of 1 MB data is transmitted via TCP through a network with two, three, and four tight links, respectively, each with single-hop persistent exponential cross traffic with an average rate $\lambda = 50$ Mbps in uplink and downlink. The capacity of the access links and that of the tight links is 100 Mbps. The neural network is trained with this additional training set (ii) for multiple tight links along with training set (i) for a single tight link.

Fig. 6.13b compares the results of the direct method with the neural network. As can be seen clearly, the results have improved significantly with the neural network after it has been trained for multiple tight links. The mean value matches the true available bandwidth, and the estimates have less variability.

6.8 EVALUATION OF LOSS

The fluid-flow models that are used in available bandwidth estimation assume lossless systems and few methods for bandwidth estimation consider loss. The iterative packet train method Pathload [30] uses loss as an indication that $r_{in} > C - \lambda$. The authors in [12] model lost packets as incurring an infinite delay. Further, it is possible to define the output rate r_{out} of a packet train in the

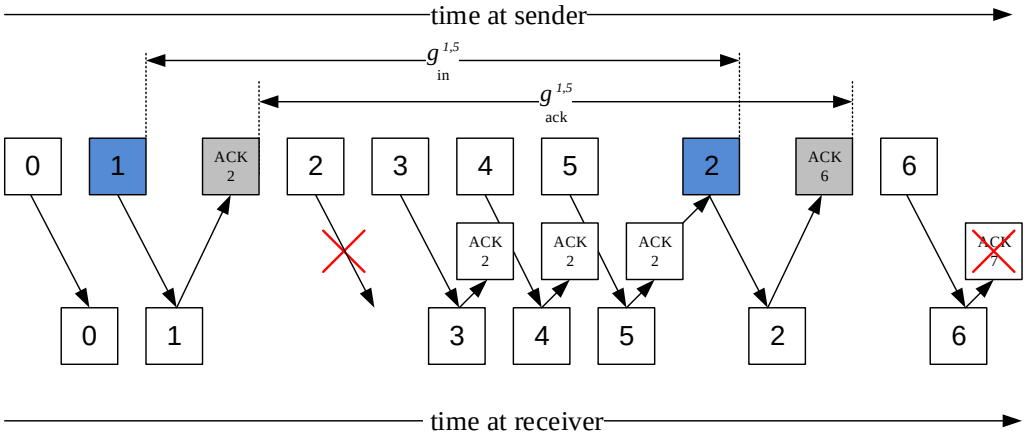


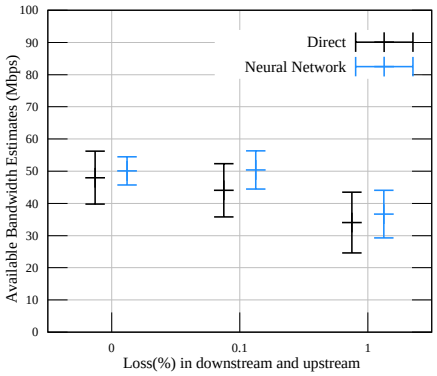
Figure 6.14: Use of the ack-gap model in the presence of packet loss. Duplicate acknowledgments are ignored and the ack-gap is closed by the next higher cumulative acknowledgment. The corresponding input gap has to consider the retransmission that triggered this acknowledgment.

presence of loss, considering only the packets that are received. The output gap g_{out} of a packet pair is, however, void if any of the two packets is lost. This may cause estimation bias, since the packet pairs that encounter congestion have a higher loss probability.

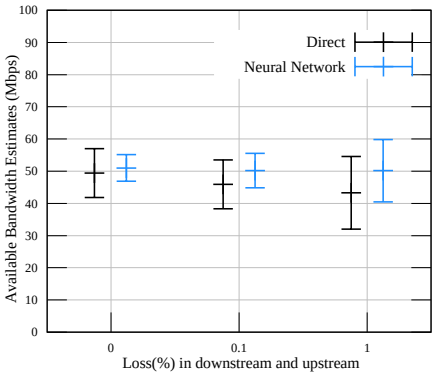
If acknowledgment gaps are used, packet loss has to be taken into account since it causes retransmissions and perturbs the sequence. Fig. 6.14 shows an example, where a packet is lost, and three duplicate acknowledgments trigger a fast retransmit. To deal with this case, we define the following procedure: first, when determining the ack-gaps, duplicate acknowledgments are ignored; also packets that are retransmitted later are ignored; second, the packets that triggered the remaining acknowledgments are identified; these are used to compute the corresponding input gaps. In the example in Fig. 6.14, ACK 2 and ACK 6 remain, resulting in $g^{1,5}_{ack}$. The acknowledgments have been triggered by packet 1 and by the retransmission of packet 2, respectively. Hence, the corresponding $g^{1,5}_{in}$ is determined as the difference of the send timestamps of these two transmissions.

While the procedure can deal with single packet losses, we note that burst losses can result in more intricate constellations that may not be resolvable unambiguously. The loss of acknowledgments, on the other hand, is less of an issue as it is typically resolved by the next cumulative acknowledgment.

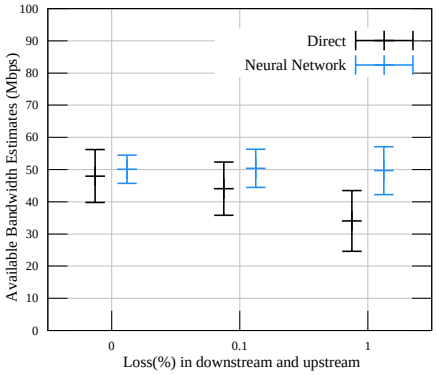
In Fig. 6.15a and Fig. 6.15b, we show the comparative results of the neural network-based and direct method for exponential cross traffic with an average rate $\lambda = 50$ Mbps using CUBIC and BBR, respectively. Loss rates of 0%, 0.1% and 1% are evaluated. For TCP CUBIC in Fig. 6.15a, though the estimates of the neural network-based method show less variability, both the methods underestimate the available bandwidth with increasing loss rate. This is due to



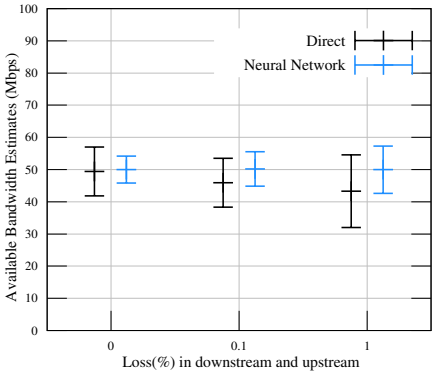
(a) Bandwidth estimates for TCP CUBIC trained for lossless network



(b) Bandwidth estimates for TCP BBR trained for lossless network



(c) Bandwidth estimates for TCP CUBIC trained for loss in the network



(d) Bandwidth estimates for TCP BBR trained for loss in the network

Figure 6.15: Bandwidth estimates for TCP CUBIC and BBR in the presence of loss with a neural network trained for (a) & (b) lossless network and for (c) & (d) loss in the network. Training improves the estimates in the presence of higher loss.

the fact that TCP CUBIC is a loss-based congestion control algorithm and adjust its congestion window when packet loss occurs which affects the distribution of input and acknowledgment gaps. Therefore, the neural network trained for a lossless network does not perform well in the presence of loss. In the case of the direct method with higher packet loss, few samples remain that pass the threshold test for the regression step to estimate the upward segment of the gap response curve.

In contrast to traditional loss-based congestion control algorithms like CUBIC, BBR is designed to respond to actual congestion rather than packet loss. This can be seen from the results of the direct method in Fig. 6.15b which are better in the case of BBR as compared to CUBIC. In addition to that, though the variation in estimate increases with increase in loss rate, the neural network trained for TCP BBR lossless link can estimate the bandwidth reliably in the

presence of loss. To improve the results for both TCP CUBIC and BBR further, we consider training the NN for packet loss.

6.8.1 *Training for Loss*

We generate (iii) training set in the presence of a loss in the uplink and downlink, and exponential cross traffic with an average rate $\lambda = 50$ Mbps. The capacities of the tight link and the access link are set to 100 Mbps. The neural network is trained with this additional (iii) training set for loss in the link along with the (i) training set for a lossless single tight link. As can be seen in Fig. 6.15c for TCP CUBIC, the results have improved significantly after training of neural network for loss. The mean value matches the true available bandwidth even in the higher loss. TCP BBR that is able to perform well in the presence of loss, however, the training results in reduced variability in estimates as shown in Fig. 6.15d.

7

CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

In this thesis, we tackled the challenges that the state-of-the-art bandwidth estimation tools face due to non-conformance with the assumptions of the fluid-flow model such as (i) random cross traffic, (ii) packet interference, (iii) packet loss, (iv) multiple tight links, (v) tight link differs from bottleneck link, and (vi) measurement inaccuracies. For these network scenarios, which do not comply with assumptions of the fluid-flow model, we investigated how to benefit from machine learning techniques for end-to-end active and passive available bandwidth estimation. While using standard packet train probes we proposed three methods for active available bandwidth estimation; the first method is based on regression and the other two methods are based on multi-class classification.

To estimate the available bandwidth using regression-based machine learning techniques, we proposed a method that is motivated by the characteristic rate response curve of a network. We used ratios of data rates and a suitable normalization to achieve an implementation that is scale-invariant with respect to the network capacity. To reduce the amount of probe traffic, we implemented an iterative method that varies the probe rate adaptively. The selection of probe rates is performed by a neural network that acts as a recommender. The recommender effectively selects the probe rates that reduce the estimation error most quickly. To compare with state-of-the-art, we formulated available bandwidth estimation as a classification problem. We compared our full classifier based on our proposed method using the information of all previous probe rates with the state-of-the-art individual classifier based on only the current probe rate. We evaluated our method with other supervised machine learning techniques.

We formulated available bandwidth estimation as a multi-class classification problem and proposed two different classification-based methods. The first method is based on reinforcement learning which learns through network path's observations without having a training phase. We treated available bandwidth estimation as a single-state MDP multi-armed bandit problem and de-

fined a reward parameter as a function of the input and output rates, which is maximized when the input probe rate is equal to the available bandwidth in the network. We implemented ε-greedy algorithm that followed exploration-exploitation mechanism to find the input rate that maximized a cumulative reward function and report it as available bandwidth.

To estimate the available bandwidth in the networks, where capacity and cross traffic changes over time randomly, we proposed another classification-based method employing supervised learning. We have shown that using supervised learning-based techniques, we can improve available bandwidth estimates significantly with six packet trains and reduce the overhead in the network caused by active probing. We have found out that although there is a trade-off between the amount of injected probe traffic and the error rate of an estimator, we are able to obtain significantly good results even by using four packet trains. Furthermore, by applying a median filter to estimation results, we have been able to track the changes in the available bandwidth.

We investigated how techniques from active probing can benefit TCP bandwidth estimation. The difficulty is due to the uncontrollable traffic patterns emitted by TCP that do not match typical active probes such as packet trains. To solve the issue, we used individual packet gaps and applied a linear regression technique to estimate the gap response curve. Taking advantage of the feedback that is provided by TCP acknowledgments, we enhanced the estimation method to use sender-side measurements only. Since the cross traffic in the reverse path introduces noise in the acknowledgment gaps, we investigated that how neural networks can be used to estimate available bandwidth from noise-afflicted acknowledgment gaps.

For all the proposed machine learning-based methods to estimate available bandwidth using active and passive measurements, we conducted a comprehensive measurement study in a controlled network testbed. Our results showed that machine learning techniques can significantly improve available bandwidth estimates by reducing bias and variability. This holds true also for network configurations that have not been included in the training data set such as different types and intensities of cross traffic and randomly generated networks where the capacity and cross traffic intensity can vary substantially. We also included network scenarios with multiple tight links and multi-hop networks where the tight link differs from the bottleneck link. For the passive measurements, we evaluated our proposed method in the presence of both data and acknowledgment losses.

7.2 FUTURE WORK

There are several key issues in the bandwidth estimation which lead to some open research questions and possible future work directions. A potential research direction lies in improving the measurement infrastructure to improve the accuracy of estimates.

There exists certain networks which have limited capabilities in term of power resources, processing and memory, e.g., Wireless Sensor Networks (WSN)s. The future work is to tailor the proposed machine learning-based bandwidth estimation techniques for these networks.

A single state MDP that we have proposed can be extended to multi-state MDP in future work to estimate the available bandwidth in networks such as vehicular communications, where it is not possible to create training data sets that can represent the complete dynamics of network conditions and the only possible solution is to use reinforcement learning but a single state MDP is not sufficient.

Since the Internet relies on congestion control protocols and adaptive applications that adjust their data rate to achieve good performance while avoiding network congestion, the scope of future work is to utilize the knowledge gained from available bandwidth estimation in rate-adaptive applications. Furthermore, the information provided by short-lived TCP flows can be used for the improvement of recent TCP's variants such as CUBIC and BBR, and for the development of new protocols.

Part II

APPENDIX

DATA SET

We provide data sets generated in a controlled network testbed located at Leibniz University Hannover for training and evaluation purposes of our machine learning-based proposed methods. The data sets can be downloaded from [22]. In all the data sets, to investigate the effect of packet interference, the packet size of the cross traffic is $l = 1514$ B including the Ethernet header. The data sets are generated considering different network parameters correspond to the following scenarios known to be difficult in the bandwidth estimation:

1. Cross traffic intensities: The first data set consists of samples for a single tight link with the capacity $C = 100$ Mbps and exponential cross traffic with an average rate of $\lambda \in \{25, 50, 75\}$ Mbps.

2. Cross traffic burstiness: We consider three different types of cross traffic for the second data set: CBR, exponential, and Pareto traffic. The average rate of cross traffic is $\lambda = 50$ Mbps in all cases.

3. Multiple tight links: The third data set is generated for multiple tight links. The path-persistent probe streams experience single hop-persistent exponential cross traffic with the average rate $\lambda = 50$ Mbps while traversing multiple tight links of capacity $C = 100$ Mbps. The capacity of the access links is 1 Gbps.

4. Tight link differs from bottleneck link: We consider two scenarios for the fourth data set: first, in which the tight link follows the bottleneck link and second, in which it precedes the latter. In both scenarios, the tight link capacity is $C = 100$ Mbps and the bottleneck capacity is $C_b = 50$ Mbps with cross traffic intensity of $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps, respectively.

REGRESSION-BASED ACTIVE AVAILABLE BANDWIDTH ESTIMATION

For the active available bandwidth estimation method proposed in chapter 4, we provide benchmark data sets consisting of p-dimensional matrices of ratios of r_{in}/r_{out} which are input to the neural network and r_{in} to calculate δ_r, with respect to which the available bandwidth and bottleneck capacity are normalized. The first row of data sets has true bottleneck capacity and available bandwidth values as output for training the neural network. In addition to

the above-mentioned data sets, an additional data set is generated for a single tight link but with the different capacity $C = 50$ Mbps and the exponential cross traffic that has an average rate of $\lambda \in \{12.5, 25, 37.5\}$ Mbps to evaluate the scale-invariance approach of our proposed method.

CLASSIFICATION-BASED ACTIVE AVAILABLE BANDWIDTH ESTIMATION USING REINFORCEMENT LEARNING

For the first classification-based method employing reinforcement learning proposed in chapter 5, we provide all the above-mentioned four data sets consisting of 1000 repeated experiments. We provide p-dimensional matrices of input rate r_{in} and output rate r_{out}.

CLASSIFICATION-BASED ACTIVE AVAILABLE BANDWIDTH ESTIMATION USING SUPERVISED LEARNING

We provide the first three data sets for the second classification-based method using supervised learning described in chapter 5. In addition to that, we include data set generated from random networks with arbitrarily chosen capacity and cross traffic intensity. We provide p-dimensional matrices of input rate r_{in} and output rate r_{out}. The first row of data sets has true bottleneck capacity and available bandwidth values as output for training the learning framework.

PASSIVE AVAILABLE BANDWIDTH ESTIMATION

We provide the first three data sets for both TCP variants, i.e., CUBIC and BBR for passive bandwidth estimation method described in chapter 6. In addition to that we include data set generated in the presence of data and acknowledgment packet loss. Loss rates of 0%, 0.1% and 1% are considered. The data sets are generated from the transmission of DASH video chunks of the size of about 1 MB. We provide matrices of the ratio of input and output gaps, i.e., g_{in}/g_{out} and quotient of packet size and input gap, i.e., l/g_{in}. We note the latter is in Mbps. The first row of data sets has true bottleneck capacity and available bandwidth values as output for training the learning framework.

BIBLIOGRAPHY

[1] Jin Cao, William S Cleveland, and Don X Sun. "Bandwidth estimation for best-effort internet traffic." In: *Statistical science* (2004), pp. 518–543 (cit. on p. 3).

[2] Heung Ki Lee, Varrian Hall, Ki Hwan Yum, Kyoung Ill Kim, and Eun Jung Kim. "Bandwidth estimation in wireless LANs for multimedia streaming services." In: *Advances in Multimedia* 2007 (2007) (cit. on p. 3).

[3] Nadeem Aboobaker, David Chanady, Mario Gerla, and MY Sanadidi. "Streaming media congestion control using bandwidth estimation." In: (2002), pp. 89–100 (cit. on p. 3).

[4] Richard Süselbeck, Gregor Schiele, Patricius Komarnicki, and Christian Becker. "Efficient bandwidth estimation for peer-to-peer systems." In: *2011 IEEE International Conference on Peer-to-Peer Computing*. IEEE. 2011, pp. 10–19 (cit. on p. 3).

[5] Erol Gelenbe, Xiaowen Mang, and Raif Onvural. "Bandwidth allocation and call admission control in high-speed networks." In: *IEEE Communications Magazine* 35.5 (1997), pp. 122–129 (cit. on p. 3).

[6] Yong Zhu, Constantinos Dovrolis, and Mostafa Ammar. "Dynamic overlay routing based on available bandwidth estimation: A simulation study." In: *Computer Networks* 50.6 (2006), pp. 742–762 (cit. on p. 3).

[7] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy. "Bandwidth estimation: metrics, measurement techniques, and tools." In: *IEEE Network* 17.6 (2003), pp. 27–35 (cit. on p. 4).

[8] Xiliang Liu, Kaliappa Ravindran, and Dmitri Loguinov. "A queueing-theoretic foundation of available bandwidth estimation: single-hop analysis." In: *IEEE/ACM Transactions on Networking* 15.4 (2007), pp. 918–931 (cit. on pp. 4, 5, 7, 9, 13–15, 18, 19, 21, 32, 38, 64, 96, 110).

[9] Manish Jain and Constantinos Dovrolis. "Ten Fallacies and Pitfalls on End-to-End Available Bandwidth Estimation." In: *ACM Internet Measurement Conference*. 2004, pp. 272–277 (cit. on pp. 5, 9, 15, 32, 38, 64, 92, 110).

[10] Li Lao, Constantinos Dovrolis, and M.Y. Sanadidi. "The Probe Gap Model can Underestimate the Available Bandwidth of Multihop Paths." In: *ACM SIGCOMM Computer Communication Review* 36.5 (2006), pp. 29–34 (cit. on pp. 5, 15, 38, 110).

[11] Xiliang Liu, Kaliappa Ravindran, and Dmitri Loguinov. "A stochastic foundation of available bandwidth estimation: Multi-hop analysis." In: *IEEE/ACM Transaction on Networking* 16.1 (2008), pp. 130–143 (cit. on pp. 5, 7, 15, 18, 19, 21, 96).

[12] Ralf Lübben, Markus Fidler, and Jörg Liebeherr. "Stochastic Bandwidth Estimation in Networks with Random Service." In: *IEEE/ACM Transactions on Networking* 22.2 (2014), pp. 484–497 (cit. on pp. 5, 12, 15, 21, 38, 64, 110, 111).

[13] Vern Paxson. "End-to-End Internet Packet Dynamics." In: *IEEE/ACM Transactions on Networking* 7.3 (1999), pp. 277–292 (cit. on pp. 6, 12, 16, 21).

[14] Kathleen Nichols and Van Jacobson. "Controlling queue delay." In: *Queue* 10.5 (2012), p. 20 (cit. on p. 6).

[15] Hui Zhou, Yongji Wang, Xiuli Wang, and Xiaoyong Huai. "Difficulties in estimating available bandwidth." In: *2006 IEEE International Conference on Communications*. Vol. 2. IEEE. 2006, pp. 704–709 (cit. on pp. 6, 7).

[16] Guojun Jin and Brian L Tierney. "System capability effects on algorithms for network bandwidth measurement." In: *Proceedings of 3rd ACM SIGCOMM conference on Internet measurement*. ACM. 2003, pp. 27–38 (cit. on pp. 6, 7).

[17] Dan Tsafrir. "The context-switch overhead inflicted by hardware interrupts (and the enigma of do-nothing loops)." In: *Proceedings of 2007 workshop on Experimental computer science*. ACM. 2007, p. 4 (cit. on p. 6).

[18] Minsu Shin, Mankyu Park, Deockgil Oh, Byungchul Kim, and Jaeyong Lee. "Clock synchronization for one-way delay measurement: A survey." In: *Advanced Communication and Networking* (2011), pp. 1–10 (cit. on p. 7).

[19] Ravi Prasad, Manish Jain, and Constantinos Dovrolis. "Effects of interrupt coalescence on network measurements." In: *Passive and Active Measurement Workshop*. 2004, pp. 247–256 (cit. on pp. 7, 16).

[20] Qianwen Yin and Jasleen Kaur. "Can machine learning benefit bandwidth estimation at ultra-high speeds?" In: *Passive and Active Measurement Conference*. 2016, pp. 397–411 (cit. on pp. 7, 8, 16, 23–25, 44, 46, 49).

[21] Sukhpreet Kaur Khangura and Sami Akın. "Machine Learning for Measurement-based Bandwidth Estimation." In: *ITC 31- Networked Systems and Services* (2019) (cit. on pp. 7, 53).

[22] *Bandwidth Estimation Traces for training and for evaluation*. `https://www.ikt.uni-hannover.de/bandwidthestimationtraces.html` (cit. on pp. 9, 53, 121).

[23] Jörg Liebeherr, Markus Fidler, and Sharokh Valaee. "A System Theoretic Approach to Bandwidth Estimation." In: *IEEE/ACM Transactions on Networking* 18.4 (2010), pp. 1040–1053 (cit. on pp. 11, 21, 22, 25).

[24] Klaus Mochalski and Klaus Irmscher. "On the use of passive network measurements for modeling the internet." In: *Kommunikation in Verteilten Systemen (KiVS)*. Springer. 2003, pp. 371–382 (cit. on p. 11).

[25] S. Keshav. "A control-theoretic approach to flow control." In: *Proc. ACM SIGCOMM*. Sept. 1991, pp. 3–15 (cit. on pp. 12, 16, 21).

[26] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. "What do packet dispersion techniques measure?" In: *IEEE INFOCOM*. 2001, pp. 905–914 (cit. on pp. 12, 16, 21).

[27] Vinay Joseph Ribeiro, Rudolf H Riedi, Richard G Baraniuk, Jiri Navratil, and Les Cottrell. "pathChirp: Efficient available bandwidth estimation for network paths." In: *Passive and Active Measurement Workshop*. 2003 (cit. on pp. 12, 21, 25).

[28] Alok Shriram and Jasleen Kaur. "Empirical evaluation of techniques for measuring available bandwidth." In: *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE. 2007, pp. 2162–2170 (cit. on pp. 12, 22).

[29] Bob Melander, Mats Bjorkman, and Per Gunningberg. "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks." In: *IEEE Globecom*. 2000, pp. 415–420 (cit. on pp. 15, 18, 21, 41).

[30] Manish Jain and Constantinos Dovrolis. "Pathload: A measurement tool for end-to-end available bandwidth." In: *Passive and Active Measurement Workshop*. 2002 (cit. on pp. 17, 21, 25, 44, 48, 111).

[31] Ningning Hu and Peter Steenkiste. "Evaluation and characterization of available bandwidth probing techniques." In: *IEEE Journal on Selected Areas in Communications* 21.6 (2003), pp. 879–894 (cit. on pp. 17, 21, 26).

[32] Jacob Strauss, Dina Katabi, and Frans Kaashoek. "A measurement study of available bandwidth estimation tools." In: *ACM Internet Measurement Conference*. 2003, pp. 39–44 (cit. on pp. 18, 21).

[33] Andreas Johnsson, Bob Melander, and Mats Björkman. "Diettopp: A first implementation and evaluation of a simplified bandwidth measurement method." In: *Swedish National Computer Networking Workshop*. 2004 (cit. on pp. 18, 21, 26, 96).

[34] 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: *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. 2006, pp. 73–84 (cit. on pp. 18, 21, 53–56, 68).

[35] Marcia Zangrilli and Bruce B Lowekamp. "Applying principles of active available bandwidth algorithms to passive TCP traces." In: *International Workshop on Passive and Active Network Measurement*. Springer. 2005, pp. 333–336 (cit. on p. 21).

[36] Cao Le Thanh Man, Go Hasegawa, and Masayuki Murata. "A merged inline measurement method for capacity and available bandwidth." In: *International Workshop on Passive and Active Network Measurement*. Springer. 2005, pp. 341–344 (cit. on p. 21).

[37] Manish Jain and Constantinos Dovrolis. "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput." In: *IEEE/ACM Transactions on Networking* 11.4 (2003), pp. 537–549 (cit. on pp. 21, 39, 92).

[38] Ling-Jyh Chen. "A machine learning-based approach for estimating available bandwidth." In: *TENCON*. 2007, pp. 1–4 (cit. on pp. 23–25).

[39] Natsuhiko Sato, Takashi Oshiba, Kousuke Nogami, Anan Sawabe, and Kozo Satoda. "Experimental comparison of machine learning-based available bandwidth estimation methods over operational LTE networks." In: *IEEE Symposium on Computers and Communications (ISCC)*. 2017, pp. 339–346 (cit. on pp. 23–25).

[40] Sukhpreet Kaur Khangura, Markus Fidler, and Bodo Rosenhahn. "Neural Networks for Measurement-based Bandwidth Estimation." In: *IFIP Networking Conference* (2018) (cit. on p. 24).

[41] Sukhpreet Kaur Khangura, Markus Fidler, and Bodo Rosenhahn. "Machine Learning for Measurement-based Bandwidth Estimation." In: *Elsevier's Computer Communications Journal* (2019) (cit. on p. 24).

[42] Alaknantha Eswaradass, X-H Sun, and Ming Wu. "A neural network based predictive mechanism for available bandwidth." In: *Parallel and Distributed Processing Symposium*. 2005 (cit. on p. 24).

[43] Péter Hága, Sándor Laki, Ferenc Tóth, István Csabai, József Stéger, and Gábor Vattay. "Neural Network Based Available Bandwidth Estimation in the ETOMIC Infrastructure." In: *TRIDENTCOM*. 2007, pp. 1–10 (cit. on pp. 24, 25, 38).

[44] David S Anderson, Mike Hibler, Leigh Stoller, Tim Stack, and Jay Lepreau. "Automatic online validation of network configuration in the emulab network testbed." In: *IEEE International Conference on Autonomic Computing*. 2006, pp. 134–142 (cit. on pp. 28, 94).

[45] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. "MoonGen: A Scriptable High-Speed Packet Generator." In: *ACM Internet Measurement Conference*. Oct. 2015 (cit. on p. 28).

[46] Stefano Avallone, S Guadagno, Donato Emma, Antonio Pescape, and Giorgio Ventre. "D-ITG distributed internet traffic generator." In: *Quantitative Evaluation of Systems*. 2004, pp. 316–317 (cit. on pp. 28, 94).

[47] J Laine, S Saaristo, and R Prior. *Real-time UDP data emitter (rude) and collector for rude (crude)*. 2000. URL: `https://sourceforge.net/projects/rude/` (cit. on pp. 29, 94).

[48] Carl Edward Rasmussen and Christopher KI Williams. "Gaussian processes for machine learning. 2006." In: *The MIT Press, Cambridge, MA, USA* 38 (2006), pp. 715–719 (cit. on p. 50).

[49] Sukhpreet Kaur Khangura and Sami Akın. "Online Available Bandwidth Estimation using Multiclass Supervised Learning Techniques." In: *IEEE Journal on Selected Areas in Communications* (2019 manuscript submitted) (cit. on p. 53).

[50] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cit. on p. 57).

[51] Erik Bergfeldt. "Available-bandwidth estimation in packet-switched communication networks." PhD thesis. Linköping University Electronic Press, 2010 (cit. on p. 64).

[52] Corinna Cortes and Vladimir Vapnik. "Support-vector networks." In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 71).

[53] Thomas G Dietterich and Ghulum Bakiri. "Solving multiclass learning problems via error-correcting output codes." In: *Journal of artificial intelligence research* 2 (1994), pp. 263–286 (cit. on p. 71).

[54] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks." In: *Information processing & management* 45.4 (2009), pp. 427–437 (cit. on p. 73).

[55] Sangtae Ha and Injong Rhee. "Taming the elephants: New TCP slow start." In: *Computer Networks* 55.9 (2011), pp. 2092–2110 (cit. on pp. 91, 95).

[56] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. "BBR: Congestion-based congestion control." In: (2016) (cit. on p. 91).

[57] Sukhpreet Kaur Khangura and Markus Fidler. "Available Bandwidth Estimation from Passive TCP Measurements using the Probe Gap Model." In: *IFIP Networking Conference* (2017) (cit. on p. 91).

[58] Sukhpreet Kaur Khangura. "Neural Network-based Available Bandwidth Estimation from TCP Sender-side Measurements." In: *The 8th IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks* (2019) (cit. on p. 91).

[59] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis. "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP." In: *ACM Conference on Multimedia Systems*. 2011, pp. 157–168 (cit. on p. 92).

[60] Qi Lin, Yitong Liu, Yun Shen, Hui Shen, Lin Sang, and Dacheng Yang. "Bandwidth estimation of rate adaption algorithm in DASH." In: *IEEE Globecom Workshops*. 2014, pp. 243–247 (cit. on p. 92).

[61] Sangtae Ha, Injong Rhee, and Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant." In: *ACM SIGOPS Operating Systems Review* 42.5 (2008), pp. 64–74 (cit. on p. 93).

[62] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs. "Iperf: The TCP/UDP bandwidth measurement tool." In: *https://iperf.fr/* (2005) (cit. on p. 94).

[63] Kurt Wagner. *Short evaluation of linux's Token-Bucket-Filter (TBF) queuing discipline*. 2001 (cit. on p. 94).

[64] *ethtool*. `://linux.die.net/man/8/ethtool`. Accessed: 05-01-2017 (cit. on p. 94).

[65] Foivos Michelinakis, Gunnar Kreitz, Riccardo Petrocco, Boxun Zhang, and Joerg Widmer. "Passive mobile bandwidth classification using short lived TCP connections." In: *IFIP Wireless and Mobile Networking Conference (WMNC)*. 2015, pp. 104–111 (cit. on p. 95).

[66] David M Lane, David Scott, Mikki Hebl, Rudy Guerra, Dan Osherson, and Heidi Zimmer. *An Introduction to Statistics*. Citeseer, 2017 (cit. on p. 107).

PUBLICATIONS

Sukhpreet Kaur Khangura and Sami Akın. "Online Available Bandwidth Estimation using Multiclass Supervised Learning Techniques." In: submitted to *IEEE Journal on Selected Areas in Communications*.

Sukhpreet Kaur Khangura "Neural Network-based Available Bandwidth Estimation from TCP Sender-side Measurements." In: *The 8th IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks*, Paris, November 2019.

Sukhpreet Kaur Khangura, and Sami Akın. "Measurement-based Online Available Bandwidth Estimation employing Reinforcement Learning." In: *ITC 31-Networked Systems and Service*, Budapest, Hungary, August 2019.

Sukhpreet Kaur Khangura, Markus Fidler, and Bodo Rosenhahn. "Machine learning for Measurement-based Bandwidth Estimation." In: *Computer Communications Journal*, Elsevier, August 2019.

Sukhpreet Kaur Khangura, Markus Fidler, and Bodo Rosenhahn. "Neural Networks for Measurement-based Bandwidth Estimation." In: *Proc. IFIP Networking*, Zürich, Switzerland, June 2018.

Sukhpreet Kaur Khangura, and Markus Fidler. "Available Bandwidth Estimation from Passive TCP measurements using the Probe Gap Model." In: *Proc. IFIP Networking Conference* Stockholm, Sweden, June 2017.

CURRICULUM VITAE

Name	Sukhpreet Kaur Khangura
Day of birth	22 October 1983

Education

July 2014 to present	**Ph.D.** student Leibniz Universität Hannover Thesis Title: Machine Learning-based Available Bandwidth Estimation for Future Networks
Oct. 2011 to Sept. 2013	**M.Sc.** in Computer and Communications Technology Universität des Saarlandes Thesis Title: Hybrid Model-based Coding Enhancement for MPEG
June 2009 to May 2011	**MBA(CC)** in Human Resource Management Punjabi University, India
Aug. 2006 to July 2009	**M.Tech** in Electronics and Communication Engineering Punjab Technical University, India
Aug. 2002 to July 2006	**B.Tech** in Electronics and Communication Engineering Punjab Technical University, India

Work Experience

July 2014 to present	**Leibniz Universität Hannover**, Institute of Communications Technology *Research Assistant*
Oct. 2013 to July 2014	**ASC Software Solutions GmbH**, Saarbrücken *Web application developer*

Oct. 2013 to July 2014	**Universität des Saarlandes, Saarbrücken**, Institute for Computer Architecture and Parallel Computing *Research Assistant*
July 2013 to Sept. 2013	**Universität des Saarlandes, Saarbrücken**, Institute for Computer Architecture and Parallel Computing *Student Assistant*
June 2012 to Aug 2012	**Universität des Saarlandes, Saarbrücken**, Intel Visual Computing Institute *Student Assistant*
July 2008 to Oct. 2011	**RIMT-Institute of Engineering and Technology**, Mandi-Gobindgarh, India *Teaching Assistant*
July 2006 to June 2008	**Bhai Gurdas Polytechnic College**, Sangrur, India *Teaching Assistant*

Teaching Experience

July 2014 to present	**Leibniz Universität Hannover**, Faculty of Electrical Engineering and Computer Science
Academic year of 2018 to 2019	Exercises on "Mobile Communications"
Academic year of 2014 to 2016	Exercises on "Networks and Protocols"

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*".

Final Version as of December 17, 2019 (`classicthesis`).