Study of Available Bandwidth Estimation Techniques to be applied in Packet-Switched Mobile Networks

Department of Communication Technology
Aalborg University

May 10, 2006

Dimas López Villa
Carlos Úbeda Castellanos

AALBORG UNIVERSITY
INSTITUTE OF ELECTRONIC SYSTEMS
DEPARTMENT OF COMMUNICATION TECHNOLOGY

Fredrik Bajersvej 7 ▪ DK-9220 Aalborg East Phone 96 35 80 80

TITLE:	Study of Available Bandwidth Estimation Techniques to be applied in Packet-Switched Mobile Networks
THEME:	Mobile Communications
PROJECT PERIOD:	10th Semester, September 2005 to May 2006
PROJECT GROUP:	1116

PARTICIPANTS:
Dimas López Villa
Carlos Úbeda Castellanos

SUPERVISORS:
Oumer M. Teyeb
Jan Elling

COPIES: 9
PAGES: 186
FINISHED: May 10, 2006

Abstract

*The increasing trend in the mobile air-interface data rates means that the relative contribution of the transport network towards the per-user capacity is becoming very important. Admission, load and handover control can make use of information regarding the Available Bandwidth (ABw) in the transport network, as it could end up being the bottleneck rather than the air interface. This report provides a comparative study of three ABw estimation techniques (*TOPP*,* SLoPS *and* pathChirp*) taking into account the statistical conditions of the ABw, several improvements of* pathChirp *in terms of accuracy and efficiency, and a performance evaluation of* pathChirp *under a DiffServ environment. Simulation-based studies show that* pathChirp *outperforms* TOPP *and* SLoPS*, both in terms of accuracy and efficiency. By combining the optimized linear profile with a linear least squares fitting, it is possible to reduce the average relative error from 23.6% to 3.2%. From this work, it is drawn that the use active probing has to be mainly focused on ABw trend detection for load control rather than for real-time adaptive QoS management.*

Preface

This report is written during the project period of the 10th semester as members of the Cellular Systems (CSYS) Division at the Department of Communication Technology, Institute of Electronic Systems, Aalborg University.

Acknowledgements

We want to thank our supervisors, Oumer M. Teyeb and Jan Elling, for their good suggestions and guidance during this project. We are also grateful to Jeroen Wigard, who came out with the initial idea of this project. We would like to acknowledge Nokia Danmark A/S for setting up the scholarship and providing support for the research. Finally, we would like to thank CSYS and Nokia staff for the friendly working environment.

Dedico mi trabajo en este informe a mis padres y a mi novia, que siempre me han apoyado, a la gente de Teleco de la UMH y a los Erasmus de Aalborg, que tan buenos momentos me han dado, y finalmente, a Carlos, con quien fue un placer trabajar.

Dimas López Villa
dimas@kom.aau.dk

Dedico este PFC a mis padres y a mi hermano Andrés, a los que he echado mucho de menos todo este tiempo, a mi tía Inés, que me enseñó a montar en bici, entre otras muchas cosas, a la gente de Teleco UMH 2000-2005, con la que he pasado cinco años inolvidables, a los Erasmus de Aalborg, que me han hecho sentirme como en casa, y a Dimas, amigo y compañero de proyecto, que me ha aguantado durante estos últimos nueve meses.

Carlos Úbeda Castellanos
cubeda@kom.aau.dk

Aalborg University, 10th of May 2006

Table of Contents

List of Figures

List of Tables

List of Abbreviations

8PSK	8 Phase Shift Keying
ABw	Available Bandwidth
ABwE	Available Bandwidth Estimation
AccSig	Accumulation Signature
ACK	ACKnowledgement
AF	Assured Forwarding
AP	Active Probing
BE	Best Effort
BS	Base Station
BSC	Base Station Controller
BTS	Base station Transceiver Subsystem
CBR	Constant Bit Rate
CBS	Committed Burst Size
CDF	Cumulative Density Function
CIR	Committed Information Rate
CS	Circuit Switching
DiffServ	Differentiated Services
DP	Direct Probing
DSCP	DiffServ CodePoint

EBS	Excess Burst Size
ECN	Explicit Congestion Notification
EDGE	Enhanced Data rates for GSM Evolution
EF	Expedited Forwarding
E-UTRAN	Evolved UTRAN
FDMA	Frequency Division Multiple Access
FIFO	First-In/First-Out
GERAN	GPRS/EDGE Radio Access Network
GGSN	Gateway GPRS Support Node
GMSC	Gateway MSC
GMSK	Gaussian Minimum Shift Keying
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HSDPA	High Speed Downlink Packet Access
ICMP	Internet Control Message Protocol
IGI	Initial Gap Increasing
IID	Independent and Identically Distributed
IP	Internet Protocol
ItP	Iterative Probing
LRD	Long Range Dependency
MAC	Medium Access Control
MH	Multi-Hop
MSC	Mobile services Switching Center
MTU	Maximum Transmission Unit
MWM	Multifractal Wavelet Model
OP	One Packet
OSI	Open System Interconnection

OTcl	Object Tool command language
OWD	One-Way Delay
PBS	Peak Burst Size
PCU	Packet Control Unit
PDF	Probability Density Function
PHB	Per-Hop Behavior
PING	Packet InterNet Groper
PIR	Peak Information Rate
PP	Packet Pair
PPP	Point-to-Point Protocol
PQ	Packet Quartets
PRI	Priority
PS	Packet Switching
PSD-CBR	Packet Size Distribution CBR
PSTN	Public Switched Telephone Network
PT	Packet Train
QoS	Quality of Service
RAN	Radio Access Network
RED	Random Early Detection
RIO	RED routers with In/Out bit
RIO-C	RIO Coupled
RIO-D	RIO De-coupled
RNC	Radio Network Controller
RR	Round Robin
RRM	Radio Resource Management
RTT	Round-Trip Time
SGSN	Serving GPRS Support Node

SH	Single-Hop
SLA	Service Level Agreement
SLoPS	Self-Loading Periodic Streams
SORTT	Shortest Observed RTT
sr3CM	single-rate Three-Color Marker
STM-1	Synchronous Transport Module - Level 1
TB	Token Bucket
TBM	TB Marker
TCA	Traffic Conditioning Agreement
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TOPP	Train Of Packet Pairs
ToS	Type of Service
tr3CM	two-rate Three-Color Marker
TSW	Time-Sliding Window
TSW2CM	TSW Two-Color Marker
TSW3CM	TSW Three-Color Marker
TTL	Time-To-Live
UE	User Equipment
UMTS	Universal Mobile Telecommunication System
UTRAN	UMTS Terrestrial Radio Access Network
VoIP	Voice over IP
VPS	Variable Packet Size
WCDMA	Wideband Code Division Multiple Access
WRED	Weighted RED
WRR	Weighted Round Robin

List of Symbols

The next list includes the main symbols used along this report:

α_A	Offset of the ABwE error linear regression
β_A	Slope of the ABwE error linear regression
γ	Chirp spread factor
δ	Gap between the two closest packers of a chirp
ε_A	ABwE error
η	Inter-PP ratio
$\rho_{x,c}$	Cross-traffic rate percentage of traffic class c
σ	Standard deviation
τ	Time-scale
χ	Gray-region resolution
ω	Method resolution
Θ	Inflection point of the cubic profile
Ω	OWD
d	Latency
f	Increasing trend fraction
p_{max}	Maximum dropping probability
q	Queuing delay
q^r	Queuing delay of the round-trip
u	Utilization of a hop

x_s	Transmission delay of a packet
A	End-to-end ABw
$\hat{A}$	End-to-end ABwE
A_L	ABw lower error bound
A_{TH}	ABw threshold
A_U	ABw upper error bound
C	End-to-end capacity
C_0	Joining links capacity
C_t	Capacity of the tight-link
F	Decrease factor
F_A	ABw decrease factor
I	Number of iterations of the method
K	Number of packets of a stream
L	Excursion length threshold
L_p	Probing load
M	Number of streams per fleet
N	Number of samples
P	Packet size
P_{max}	Maximum probing packet size
P_{min}	Minimum probing packet size
P_r	Packet size of the response
P_x	Cross-traffic packet size
Q	Router queue length
Q_{max}	Maximum queue size threshold
Q_{min}	Minimum queue size threshold
R_A	Asymptotic rate for the exponential profile
R_{in}	Input rate

R_k	Instantaneous rate
R_{max}	Maximum input rate
R_{min}	Minimum input rate
R_{out}	Output rate
R_{ov}	Overloading rate
R_p	Average probing rate
R_x	Cross-traffic rate
$R_{x,c}$	Cross-traffic rate of traffic class c
S_{PCT}	Pairwise Comparison Test
S_{PDT}	Pairwise Different Test
T_{in}	Input time gap
T_{in}^{max}	Maximum input gap
T_{in}^{min}	Minimum input gap
T_k	Instantaneous gap
T_{NI}	Non-intrusiveness gap
T_{out}	Output time gap
T_p	Probing time
T_{pp}	Inter-PP gap
W_c	Priority weight for traffic class c

Chapter 1

Introduction

The growth in the number of Internet users, illustrated in Figure 1.1, has been always a key factor in the technology development, either hardware or software, to improve the user's interconnection to the huge amount of available services. That would not have been possible without previous studies on network performance, e.g. *Available Bandwidth (ABw)* measurements.

Understanding the dynamic properties of the end-to-end ABw is beneficial for a proper resource management in existing and emerging mobile communication systems. The increasing trend in the wireless interface data rates means that the requested data rate for a certain service might not be guaranteed, not only because of the air interface bandwidth limitation, but also due to a limitation in the transport network's ABw.

1.1 Overview

In the last 10 years, mobile communications have quickly developed turning into an essential technology in our society. Such development means new services, which require more and more higher data rates. Appendix A.2 includes a description of the mobile network architectures to which this chapter makes reference.

In GERAN, the maximum throughput rate is 384Kbps[1], whereas newer systems, such as UTRAN and HSDPA[2], can achieve up to 2Mbps [1] and 14Mbps [2] respectively for downlink. The current transport networks are able to handle many users at these relatively low rates, so the wireless inter-

[1]The maximum throughput in theory is 471Kbps, but that is never implemented in practice due to timing constraints in the system.

[2]High Speed Downlink Packet Access.

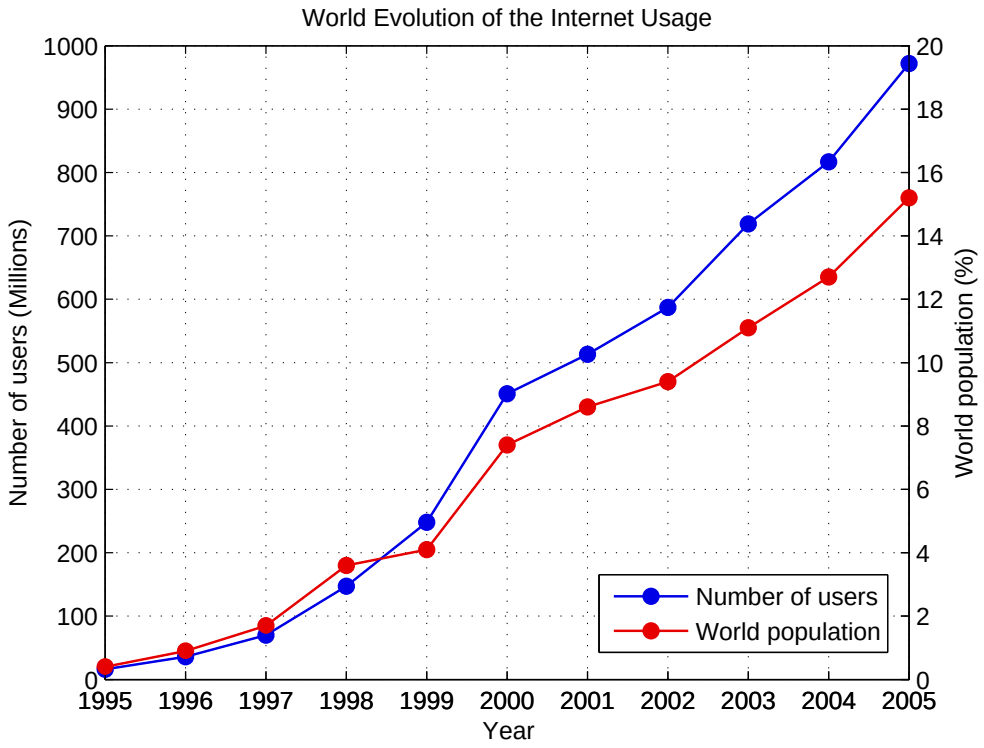

Figure 1.1: The Internet Usage has grown considerably in the last ten years. Source: IDC for 1995-97, C.I. Almanac for 1998, Nua Ltd for 1999-2001 and Internet World Stats for 2002-2005.

face is to blame for the capacity limitation. When it comes to future systems, E-UTRAN maximum downlink rate is targeting to 100Mbps [2]. This more and more increasing trend in the wireless interface data rates means that the relative contribution of the transport network towards the per-user capacity is becoming very important. It implies that the requested rate for a certain service could not be guaranteed for a given user if the path interconnecting the BS to the service provider has not enough ABw.

In general, an accurate *ABw Estimation (ABwE)* is essential in monitoring if the different flows are living up to the required Quality of Service (QoS). For instance, streaming applications could adapt their sending rate to improve the QoS depending on a real-time knowledge of the end-to-end ABw. Moving to a mobile communications core network, the ABw could also be used as an input to take decisions concerning issues such as load control, admission control, handover and routing. However, the scale of the different systems, the different traffic characteristics and the diversity of network technologies make this characterization of the end-to-end ABw a very

challenging task.

One possible way to meet this need would be the deployment of special software or hardware on each router of the network, but the cost in time and money of new equipment, maintenance of new nodes and software development makes it not practical. Moreover, this wide-scale deployment of specialized routers, which are continuously reporting bandwidth properties, might overwhelm the network. Another limitation is the impossibility to control hosts and routers outside the mobile network infrastructure, as Figure 1.2 represents.

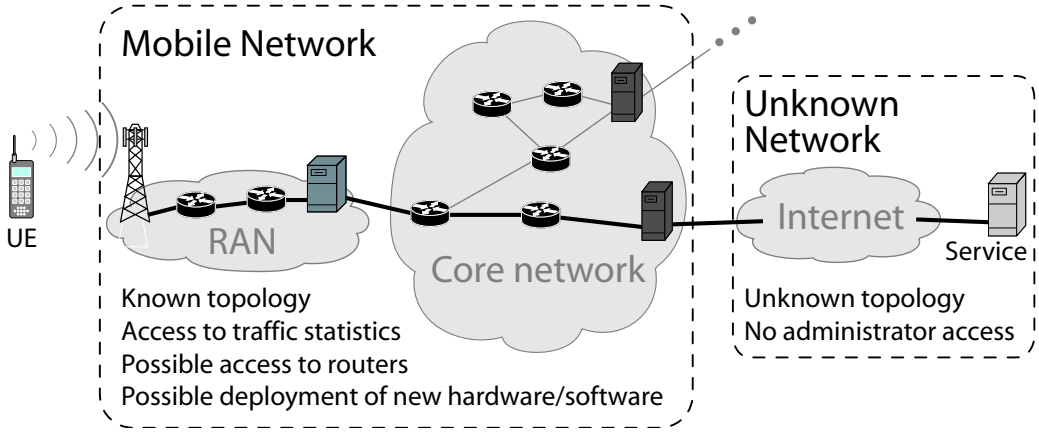

Figure 1.2: A mobile operator may not have administrator rights to the network infrastructure of a service provider out of its bounds.

An alternative is to use software run on the end hosts, which is usually called *Active Probing (AP)*. This approach means an inference of the ABw, not a direct measure of it. An ideal probing scheme should provide an accurate estimate as quickly as possible, while not placing any more load on the network than absolutely necessary. The hindrances of measuring the ABw by means of AP are that, first, the ABw is a time varying metric, second, it exhibits variability depending on the observing time-scale, and third, more and more intelligent devices are being placed on the current networks performing traffic prioritization.

The main goal of the work presented in this report is to provide the mobile systems with a tool to accurately estimate the ABw of any path of the packet-switched transport network in real-time. The tool should not need previous knowledge of the network, since part of the path might be outside the bounds of the mobile network infrastructure. For the same reason, this tool should be able to run only in the source host, avoiding the deployment of another tool at the destination host. Finally, the proposed tool ought to

work properly under traffic prioritization since current mobile systems make use of Differentiated Services, i.e. DiffServ [3], which allows various types of applications and different QoS levels to be supported.

1.2 Related Work

There has been a lot of research on this area over the last two decades. Many techniques have been proposed, and some performance and comparative studies have been done. In [4], performance comparison of *pathChirp* and *pathload* in a single-hop network environment is given among others. In their experiments all the tools showed an accuracy within 30%.

In [5], a series of ABwE experiments conducted on a high-speed testbed is presented. ABwE tools including *pathChirp*, *pathload* and others based on packet pairs are evaluated. Their results show that packet pair techniques perform worse than *pathChirp* and *pathload*.

Comparison of *pathChirp*, *pathload* and *TOPP* in single and multi-hop real network is given in [6]. *PathChirp* is shown to perform better in terms of both accuracy and efficiency.

In [7], the difference between ABw measurements in wired and wireless networks is discussed. *DietTopp* is compared in performance with *pathload* in a wired testbed. The tool is also evaluated in a wireless environment. It is shown that packet size is critical for ABw measurements.

There is some controversy on the validation of the aforementioned comparisons between the different ABwE techniques. First, some of the previous work gives very low accuracy results that are used to propose a method as the best choice. Second, some conclusions are drawn from very few results under very specific scenarios. Third, most of the method parameters used in the simulations are not clearly stated. Fourth, most of the studies compare different methods under different time-scales and using different number of samples of the ABw, which can vary the statistical properties from one estimation to another [8]. Finally, the variability of estimations is not studied.

1.3 Usage Scenario

The goal is to develop a method for E-UTRAN, but being also useful for current systems. As described in Appendix A.2, the resource management at the air interface is already handled by the RRM in the BS and the UE. So, it is only necessary to estimate the ABw of the transport network. It is proposed to deploy an agent that would continuously estimate the ABw of

the packet-switched transport network for resource management. The results of the estimations could be used as an input for the RRM.

As Figure 1.3 illustrates, the source agents could be located on the BSC of GERAN, on the RNC of UTRAN and on the BS of E-UTRAN, whereas the destination agents could be at the service provider. The dashed line means that the interface does not handle a TCP/IP connection, which limits the application of techniques that require feedback from the destination.

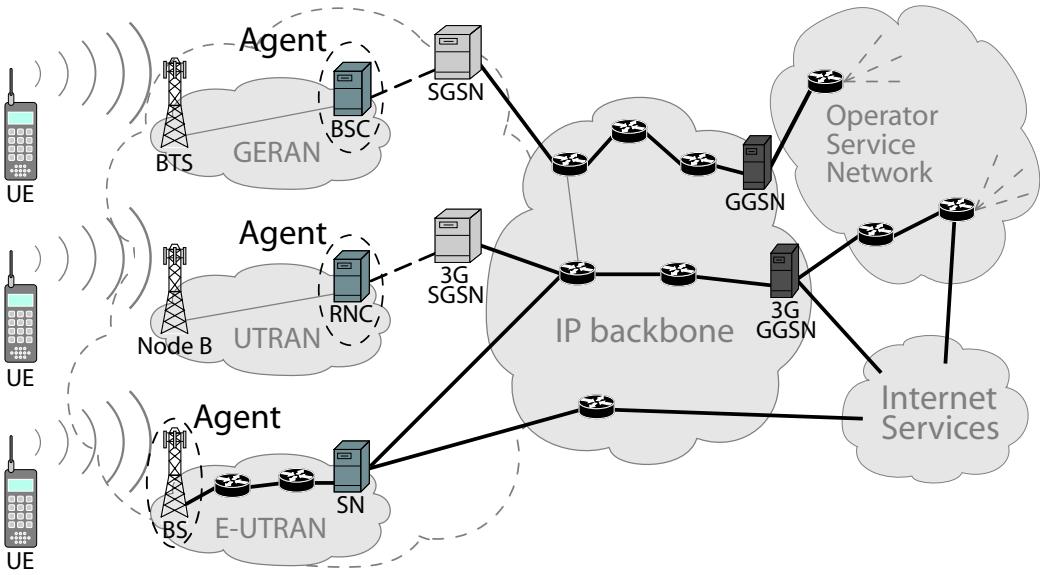

Figure 1.3: The agent could be located on the BSC of GERAN, on the RNC of UTRAN, or in the BS of E-UTRAN. The dashed lines indicate that the interface could not handle a method requiring an IP-based interface.

As far as the measurable ABw is concerned, should the agent be able to deal with different systems, the proposed method should be able to measure an ABw range as wide as possible. Other aspects that ought to be considered are the speed of the tool to give an estimate of the ABw and its performance with real cross-traffic and multi-hop paths.

1.4 Research Methodology

The research will be based on network simulations. The steps that will be followed to accomplish the goal of the report can be separated into two main phases. The first phase, which consists of the analysis and comparison of different techniques, can be divided into the following tasks:

1. Initial analysis of the state of the art in ABwE so as to choose the most suitable methods to be applied in a packet-switched mobile transport network.

2. Detailed analysis of the suggested techniques for their implementation in the simulation and adaptation of their parameters so as to keep the same statistical conditions for the three methods.

3. Choice of the appropriate network topology that allows controlling the ABw and making use of multi-hop paths, and study of the different kinds of cross-traffic to use in the simulations.

4. Comparison of the suggested methods under the proposed cross-traffic models in terms of accuracy, variability, intrusiveness, efficiency and cross-traffic properties dependency.

5. Selection of one of these techniques in terms of accuracy and efficiency according to the requirements of this work.

The second phase consists of an evaluation of the chosen technique. Three main studies will be done:

- Study of a possible improvement of the technique in terms of accuracy and/or efficiency, and adaptation of its parameters for the different requirements of current and future systems.

- Study of the possibility to work only with a source agent not to require access to hosts outside the mobile network infrastructure.

- Study of such technique with DiffServ, analyzing the method performance under different levels of priority of both the cross-traffic and the probing packets.

Figure 1.4 represents the simulation block diagram, which makes use of the next tools:

- *Network Simulator (NS-2)* [9] provides support for simulation of TCP, routing, and multicast protocols over wired and wireless networks. It is an object-oriented simulator with an OTcl[3] interpreter as a frontend. It is used to build the wired network topology, to implement the scheduling of each method, i.e. the sending of the probe packets, and to timestamp the probe packets arrival. The different kinds of cross-traffic are simulated making use of the NS-2 traffic libraries or by our own scripts.

[3]Object Tool command language.

- *MATLAB* is a high-level technical computing language, more powerful than traditional programming languages for mathematical issues. It is used for data analysis and for simulation results visualization.
- *C Language* is one of the most widely used programming languages. It is used to schedule the different simulations and to analyze real-time data during the simulations.

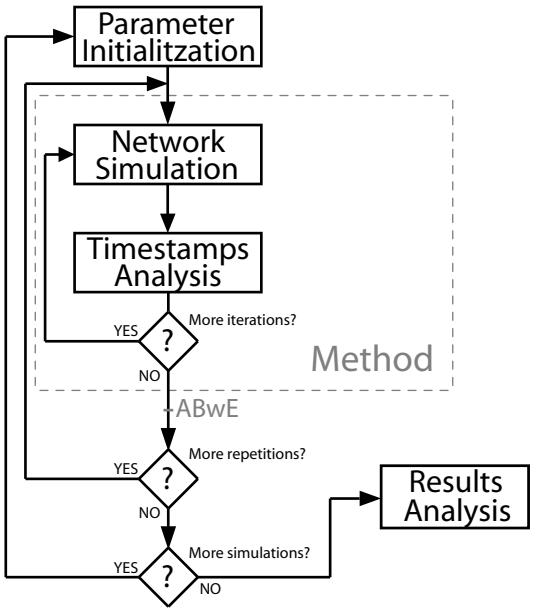

Figure 1.4: An ABwE is obtained every time the *method block* is executed. Each ABwE is repeated several times in order to study the variability of the estimator. Different simulations are scheduled so as to analyze the performance under different ABw rates and cross-traffic packet sizes.

1.5 Report Outline

The report is divided into the following chapters:

- Chapter 2 sets the foundations of ABwE techniques defining the different bandwidth metrics and providing a survey of the most common tools with some indication as to which method is relevant for this report.
- Chapter 3 describes a practical implementation of *TOPP*, *SLoPS* and *pathChirp* studying the unification and adaptation of their parameters for an easier comparison between them.

- Chapter 4 establishes the network topology, the cross-traffic models and the parameters of *TOPP*, *SLoPS* and *pathChirp*, and shows the simulation results together with a discussion of which of the three techniques should be more useful to be implemented in a packet-switched mobile transport network.

- Chapter 5 is focused on improving *pathChirp*, which includes an analysis of profiles other than the exponential structure described in [6], the optimization of different parameters, a study of two iterative approaches, a statistical treatment based on least square fitting, and a research on the effect of RTT measurements.

- Chapter 6 studies the effects of a DiffServ environment to a *pathChirp* enhanced version proposed in Chapter 5, focusing on the different scheduling modes.

- Chapter 7 summarizes the main contributions of this report and proposes ideas for future research.

Chapter 2

Available Bandwidth Estimation

Some of the current Internet services require monitoring of the ABw but, as explained in Chapter 1, a direct measure by deploying hardware or software in every router of the network would be neither efficient nor profitable. For that reason, the research in AP has attracted a lot of interest recently, resulting a wide variety of methods. By adapting these techniques, a real-time estimation of the ABw on a packet-switched mobile transport network could be implemented.

In order to reduce the mathematical complexity, all these techniques set several assumptions related to the network model, such as store-and-forward routers, First-In/First-Out (FIFO) queuing and a fluid cross-traffic model. Appendix A.1 includes a description of the networking architecture these methods are based on, and Appendix C explains the different traffic models.

This chapter sets the foundations of ABwE techniques. First, different concepts of bandwidth metrics are defined. Second, a mathematical model for the delays that packets suffer on a network, which are directly related to the ABw, is described. Finally, a survey of the most common ABwE methods is given, with some indication as to which method is relevant for this report. As a complement of this chapter, a review of Capacity Estimation techniques can be found in Appendix B.

2.1 Bandwidth Metrics

In digital communications, the concept of *bandwidth* is essential, as it is associated to the amount of information a link can handle per unit of time, i.e. information rate. However, the term bandwidth is often imprecisely used. It is necessary to distinguish between the maximum rate a single connection

could achieve and its instantaneous rate, lower than the maximum one due to the network load. It is also important to differentiate between the bandwidth related to a single hop or to a sequence of hops. Therefore, in the following sections, both bandwidth metrics are explained.

2.1.1 Capacity

The *capacity* of a link can be defined for both individual links and end-to-end paths. The rate at which a network segment can normally transfer data is the *transmission rate* or the *capacity* of the segment.

Extending the previous definition to a network path, the capacity C of a path is the maximum possible IP layer rate the path can transfer from the source to the receiver [10]. Hence, if we define C_h as the maximum layer-3 transfer rate at hop h, the hop with the minimum capacity, which sets an upper bound for the capacity of the whole path, is called the *narrow-link* of the path. The capacity of the path will then be

$$C = \min_{h=1,\ldots,H} \{C_h\} \tag{2.1}$$

where the hop capacities are assumed to be constant.

Link Layer Effects

The last definition has one constraint since the IP layer has always a lower rate than its nominal transmission rate due to the link layer encapsulation and framing. If C_{L2} is the nominal capacity of a segment, the transmission time for an IP packet is

$$\Gamma_{L3} = \frac{P_{L3} + O_{L2}}{C_{L2}} \tag{2.2}$$

where P_{L3} is the size of the layer-3 packet and O_{L2} is the size of the layer-2 overhead. If the capacity C_{L3} of that segment at the IP layer is P_{L3}/Γ_{L3}, the fraction of capacity delivered to the network layer is

$$\frac{C_{L3}}{C_{L2}} = \frac{P_{L3}}{P_{L3} + O_{L2}} = \frac{1}{1 + \frac{O_{L2}}{P_{L3}}} \tag{2.3}$$

The overhead for an Ethernet transmission is 38 Bytes, and the encapsulation for Point-to-Point Protocol (PPP) is 8 Bytes [10]. By substituting in Equation 2.3, Figure 2.1 can be obtained. The minimum layer-3 packet size to achieve an error under 20% is 150 Bytes for Ethernet. For this reason, it is suggested not to use packet sizes under 200 Bytes for calculating the transfer rate at the IP layer. On the other hand, there is an upper bound of

1500 Bytes, fixed by the Maximum Transmission Unit (MTU) [11], to avoid layer-2 fragmentation in Ethernet networks.

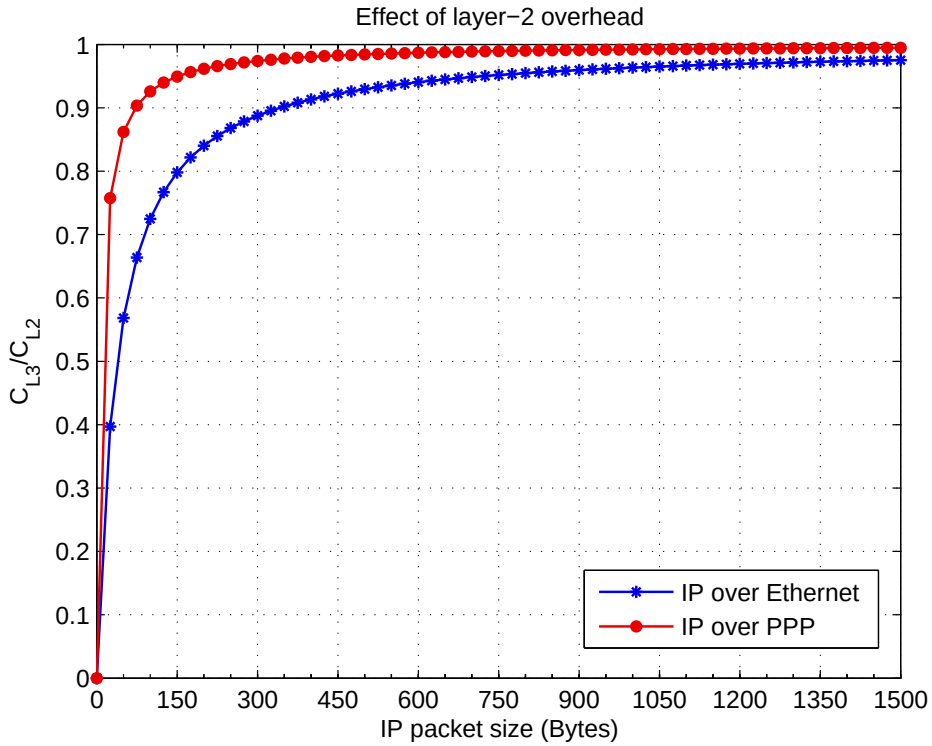

Figure 2.1: The IP layer capacity depends on the size of the IP packet relative to the layer-2 overhead. Packet sizes below 150 Bytes produce errors greater than 20% when the transfer rate is calculated at the IP-layer.

2.1.2 Available Bandwidth

The ABw of a link is described as the unused capacity of the link for a given time interval. It depends on the traffic load, also known as *cross-traffic*, hence, it is a time-varying metric. The instantaneous utilization of the link can be either 0 or 1. Therefore the average utilization [10] of the link, which refers to the cross-traffic, is

$$u^{\tau}(t) = \frac{1}{\tau} \int_{t-\tau}^{t} u(x)dx \tag{2.4}$$

where $u(x)$ is the instantaneous utilization of the link at time x and τ is the *averaging time-scale*. If C_h is the capacity of a certain hop and u_h is the

average utilization of that hop within a given time-scale, the average ABw is defined as

$$A_h^\tau(t) = (1 - u_h^\tau(t))\, C_h \tag{2.5}$$

Let consider now a path with H hops. The end-to-end ABw is the minimum ABw of all H hops,

$$A^\tau(t) = \min_{h=1,\ldots,H}\{A_h^\tau(t)\} = \min_{h=1,\ldots,H}\{(1 - u_h^\tau(t))C_h\} \tag{2.6}$$

The link with the minimum ABw is called the *tight-link* of the path. Figure 2.2 shows a *pipe network model* where each pipe represents a different link. The capacity of each link is proportional to the pipe width. The gray area refers to the used part of the link, so the white one is the ABw. In the example, the link with the lowest capacity is the third one, whereas the link with the lowest ABw is the first one. Note then that the *tight-link* and the *narrow-link* may not be the same.

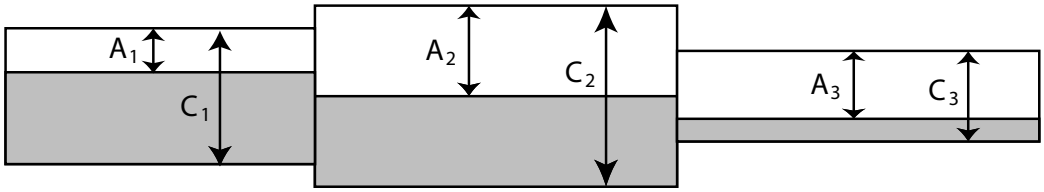

Figure 2.2: Three consecutive links with their respective capacity and ABw. The *tight-link* and the *narrow-link* may not be the same.

Statistical analysis

Since the ABw varies in time, it can be considered as a random process $A^\tau(t)$. The traffic is assumed as stationary for at least several minutes [12]. Therefore, $A^\tau(t)$ is completely defined by its mean and variance. The mean $\mu_A = E[A^\tau(t)]$, does not depend on the averaging time-scale, due to the stationarity of the random process. However, the variance $Var[A^\tau(t)] = E\{A^\tau(t) - E[A^\tau(t)]\}^2$ shows a strong dependence on the averaging time-scale and on the correlation structure of the random process $A^\tau(t)$.

To show the effect of the variance variability, a trace using CBR traffic with random noise has been analyzed. In Figure 2.3(b), the variance of the ABw decreases as the averaging time-scale increases, due to the fact that the random process is observed during a longer time period reducing its variability. As the Central Limit Theorem [13] says, the variance decreases

with the number of samples (see Figure 2.3(a)). Depending on the correlation of the random process [8], the decreasing rate changes:

- If $A^{\tau}(t)$ is a IID (independent and identically distributed) process, the variance decreases inversely proportional with the length of the averaging time-scale,
$$Var[A^{N\tau}(t)] = \frac{Var[A^{\tau(t)}]}{N} \tag{2.7}$$
where N is the number of samples of the random process $A^{\tau}(t)$.

- If $A^{\tau}(t)$ is a self-similar process (see Appendix C.4.3), the variance decreases slower,
$$Var[A^{N\tau}(t)] = \frac{Var[A^{\tau(t)}]}{N^{2(1-\varrho)}} \tag{2.8}$$
where N is the number of samples of the random process $A^{\tau}(t)$ and ϱ is the Hurst parameter $0.5 < \varrho < 1$.

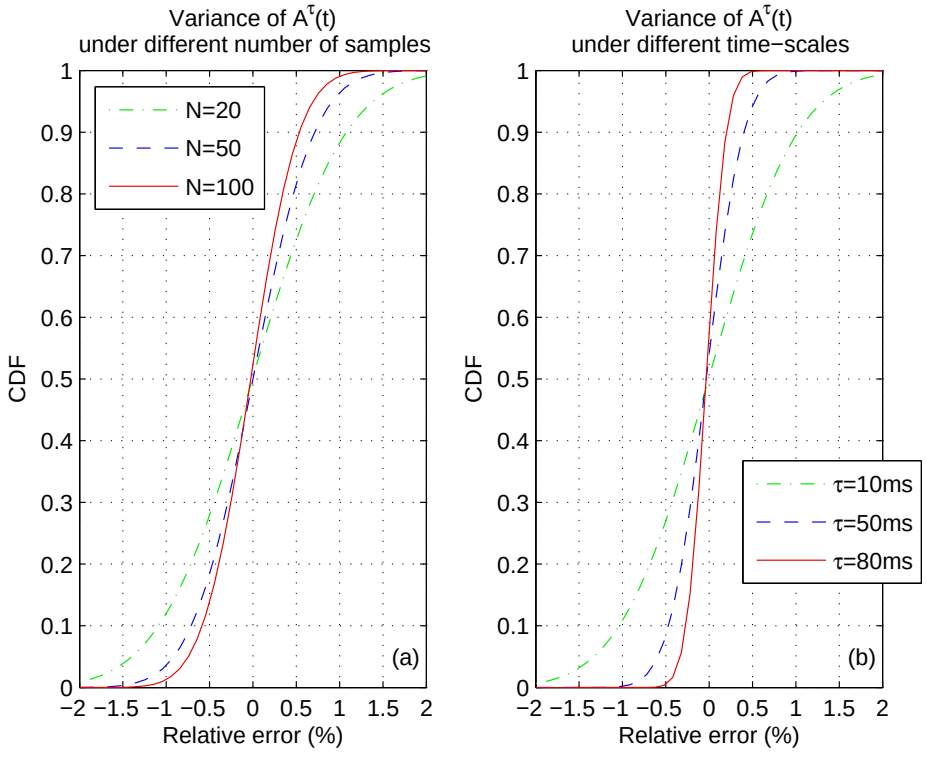

Figure 2.3: Analysis of a trace using CBR traffic with random noise. The variance of $A^{\tau}(t)$ decreases with the number of samples (a) and with the time-scale (b).

2.2 Delay Model

With the aim of understanding the different bandwidth estimation methods, it is necessary to develop a mathematical description of the different delays a packet experiences from the source to the destination.

Let define the *One-Way Delay (OWD)* as the time a packet k spends to reach a certain hop h. This time depends on the transmission delay, the latency and the queuing delay. The *transmission delay* is the time the router needs to transmit a packet on the link, which is a function of the packet size and the capacity of the link. The *latency* is the time the signal needs to travel through the link, determined by physical characteristics of the link. The *queuing delay* is the time a packet spends on the link due to the cross-traffic. The two first terms are deterministic, whereas the latter is random. Therefore, the OWD can be expressed as

$$\Omega_k^h = \sum_{s=1}^{h} (x_s + d_s + q_s) = \sum_{s=1}^{h} \left(\frac{P_k}{C_s} + d_s + q_s \right) \tag{2.9}$$

where x_s is the transmission delay of a packet of size P_k, d_s is the latency and q_s is the queuing delay.

To measure the OWD, it is necessary to take timestamps on both the source and the destination. For some applications it might be interesting to measure only from the source by using the *Round-Trip Time (RTT)*, which is defined as the time the packet spends on reaching a certain hop and returning back to the source. So, the RTT can be expressed as

$$RTT_k^h = \sum_{s=1}^{h} \left(\frac{P_k}{C_s} + d_s + q_s \right) + \sum_{s=h}^{1} \left(\frac{P_r}{C_s} + d_s + q_s^r \right) \tag{2.10}$$

where P_r is the packet size of the response and q_s^r is the queuing delay of the round-trip.

2.3 Analysis of ABwE Techniques

ABwE techniques are classified depending on the way they get the samples and how they analyze them. Assuming that the amount of cross-traffic is constant for a given time interval, i.e. a cross-traffic fluid model, the ABw, from Equation 2.6, can be rewritten as follows,

$$A = C_t - R_x \tag{2.11}$$

where C_t is the capacity of the tight-link and R_x is the cross-traffic rate. As Figure 2.4 points out, if a stream of packets is sent at $R_{in} > A$, the link will be overloaded. Let define the overloading rate R_{ov} as the fraction of the input rate the link cannot handle,

$$R_{ov} = R_{in} - A \tag{2.12}$$

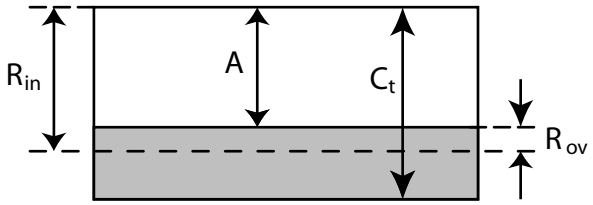

Figure 2.4: When the input rate R_{in} is larger than the ABw A, the link will be overloaded. This overloading rate R_{ov} will be proportional to amount of information the link cannot process.

On the other hand, the initial input gap between two consecutive packets of the probing stream is

$$T_{in} = \frac{P}{R_{in}} \tag{2.13}$$

where P is the packet size. During this time,the link will not process $R_{ov}T_{in}$ bytes, which will produce an increasing queuing delay Δq as

$$\Delta q = \frac{T_{in}R_{ov}}{C_t} = \frac{P}{C_t}\left(\frac{R_{in} - A}{R_{in}}\right) \qquad \text{if } R_{in} > A \tag{2.14}$$

otherwise, there will be no delay ($q = 0$). Therefore, the output rate, which is related to the gap at the destination T_{out}, can be expressed as,

$$R_{out} = \frac{P}{T_{out}} = \frac{P}{T_{in} + \Delta q} = \frac{R_{in}C_t}{C_t + R_{in} - A} \qquad \text{if } R_{in} > A \tag{2.15}$$

otherwise, the output and the input rate will be the same ($R_{out} = R_{in}$).

ABwE techniques can be classified into two main groups: *Direct Probing* and *Iterative Probing Techniques*, depending on whether they sample the ABw or they iteratively check if the input rate is larger than the ABw.

2.3.1 Direct Probing Techniques

The main characteristic of *Direct Probing (DP)* is that every probing stream provides an estimation of the ABw. If the tight-link capacity C_t is known

or can be estimated, DP methods can obtain an estimation of the ABw by measuring the output rate R_{out} or estimating the cross-traffic rate R_x. By using Equation 2.15, the ABw is

$$A = C_t - R_{in}\left(\frac{C_t}{R_{out}} - 1\right) \tag{2.16}$$

Multifractal Cross-Traffic Estimation [14], which uses the *Delphi Algorithm*, is a good example of DP. The term *multifractal* comes from the fact this method makes use of the Multifractal Wavelet Model (MWM) [15] for the cross-traffic. This model is based on the idea that Internet traffic exhibits self-similarity, since its behavior observed during short time periods is very similar to the one observed during longer durations (see Appendix C.4.3).

The MWM characterizes the cross-traffic with an expected value and a set of B parameters. These B parameters can be estimated from the current traffic conditions when there is no previous information about the network traffic properties [14].

As Figure 2.5 shows, the method sends a stream of exponentially spaced packets, called *chirp*, and computes the queuing delay of each packet. The first three packets ensure a fine resolution for the estimation while the exponential spacing of the probing packets avoids overwhelming the network.

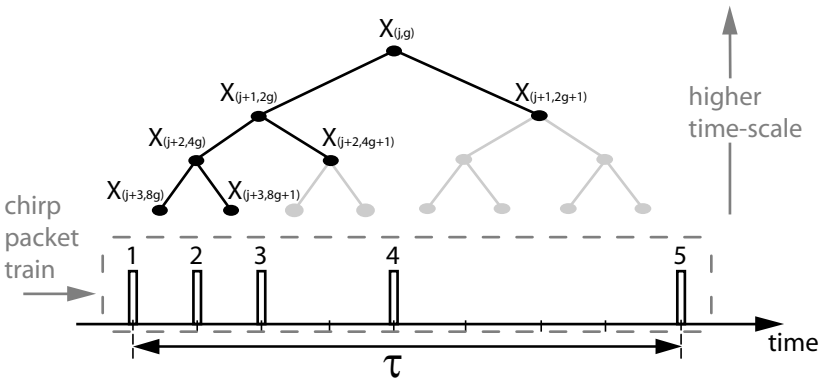

Figure 2.5: The chirp fits the binary tree structure of the MWM, where $X_{(j,g)}$ is the cross-traffic in bytes at the time-scale (j, g).

With the measured queuing delay, and making use of the MWM, it develops an approximate maximum likelihood estimate of the cross-traffic between the first two packets of the chirp, which determine the minimum time-scale. In order to perform the estimation of the cross-traffic for higher scales, the model uses a binary tree structure. The tree coefficients $X_{(j,g)}$ (parents) represent the number of cross-traffic bytes between the first and last packets of

a chirp of time length τ, and are the sum of their children (see Figure 2.5),

$$X_{(j,g)} = X_{(j+1,2g)} + X_{(j+1,2g+1)} \tag{2.17}$$

and hence, the cross-traffic rate is $R_x = X_{(j,g)}/\tau$. By substituting R_x in Equation 2.11, the ABw can be calculated.

The main problem of this technique is the need of an estimation or measurement of the capacity of the tight-link, which makes it unsuitable for a field of application where there is no previous knowledge of the network.

2.3.2 Iterative Probing Techniques

Iterative Probing (ItP) techniques are based on the fact that rates greater than the ABw increase the queuing delay and, hence, reduce the output rate. Specifically, ItP techniques sample wether the input rate exceeds the ABw or not, by sending a stream of packets and studying the behavior of the queuing delay or measuring the output rate. The stream is sent iteratively, sometimes changing some of its parameters, in order to be more accurate. There are several ItP techniques, but this section focuses only on three of them, namely *Train Of Packet Pairs*, *Self-Loading Periodic Streams* and *pathChirp*.

Train Of Packet Pairs

Train Of Packet Pairs (TOPP) [16], implemented on the publicly available tool *DietTopp*[17], is the ideal example of It. It can be considered as an extension to PP capacity estimation technique explained in Appendix B.2 as the probing stream consists of $K/2$ packet pairs, where K is the number of packets per stream (see Figure 2.6). For this reason, *TOPP* and PP share the same assumptions, advantages and drawbacks.

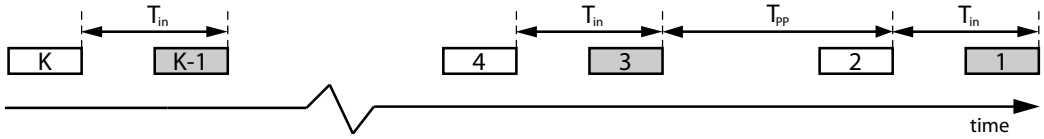

Figure 2.6: Train of $K/2$ packet pairs, where T_{in} is the time between the two packets of a pair and T_{pp} is the time between two consecutive packet pairs.

Given that all packets have the same size P, if the initial time gap between the packets of each pair is T_{in} and using Equation 2.13, each pair will be sent at an input rate $R_{in} = P/T_{in}$. So that, by substituting Equation 2.11 in 2.15, the output rate is obtained as follows

$$R_{out} = \frac{R_{in}C_t}{R_{in} + R_x} \tag{2.18}$$

Equation 2.18 is not a linear function with the input rate, but it can be modified so as to solve that limitation,

$$\frac{R_{in}}{R_{out}} = \frac{R_{in} + R_x}{C_t} = \frac{1}{C_t} R_{in} + \frac{R_x}{C_t} \tag{2.19}$$

To estimate the ABw, *TOPP* sends trains of packet pairs, uniformly increasing their input rates by changing T_{in}, from the source to the destination. The time gap between two packet pairs T_{pp} should be chosen for the pairs not to share the same queue at a router. Then, by measuring the received rate and making use of Equation 2.19, at the turning point of the resulting graph, R_{in} is equal to A (see Figure 2.7(a)). Moreover, from the slope, the capacity of the tight-link can be obtained.

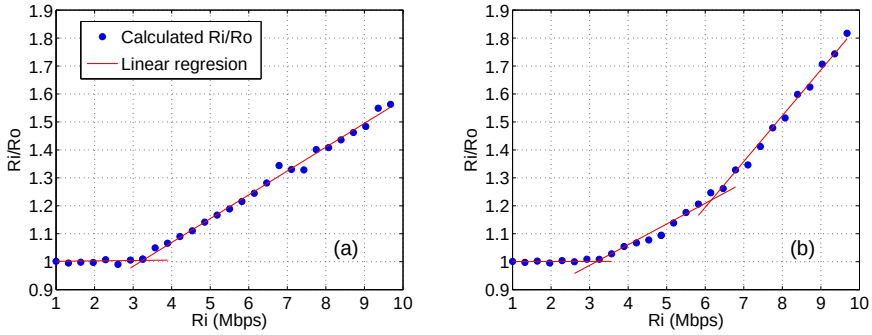

Figure 2.7: (a) is a simulation example of *TOPP* method in a path with three links of ABw 12, 3 and 6 Mbps respectively. (b) is the same example but changing the links order ($A = \{12, 6, 3\} Mbps$).

If the network has a second link with an ABw A_2 and the increase of R_{in} continues beyond A_2, a second turning point is likely to appear. This fact can easily be derived from Equation 2.18. Suppose a path with two links of ABw $A_1 > A_2$. If an input rate $R_{in} > A_1$ is inserted into the first link, the input rate at the second link would be R_{out}. Introducing again this result in Equation 2.18, the output rate of the whole path is

$$R'_{out} = \frac{R_{out} C_{t_2}}{R_{out} + R_x} = \frac{\frac{R_{in} C_t}{R_{in} + R_x} C_{t_2}}{\frac{R_{in} C_t}{R_{in} + R_x} + R_x} \tag{2.20}$$

where C_{t_2} denotes the capacity of the second link (secondary tight-link). From this equation, and following the same methodology used to obtain the linear function 2.19, this technique is not only able to detect the ABw and the capacity of the tight-link, but it can also estimate secondary tight-links (see

Figure 2.7(b)) and their respective capacities (with previous transformations on the different slopes [16]).

However, the ability to detect secondary tight-links depends on the sequence that different links have on the path as the pipes analogy of Figure 2.8 shows for the cases of Figure 2.7. The width of the pipe illustrate the ABw of each link, while the gray area illustrates the fraction of the ABw occupied by the probe traffic. In (a), the secondary tight-link remains invisible to the method, but in (b), due to the ordering of the links in the path, *TOPP* is able to detect it. All the tight links are detected only if they are ordered in descending ABw order.

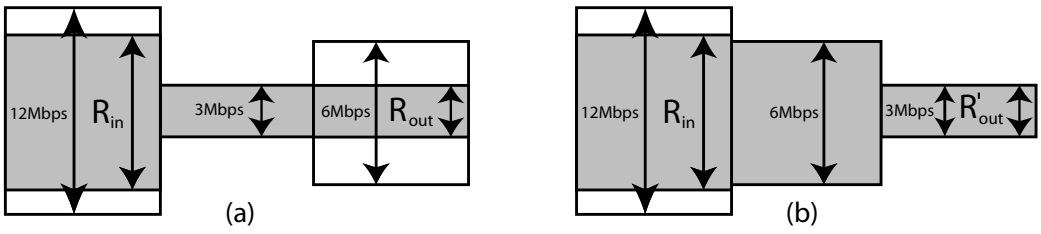

Figure 2.8: (a), related to 2.7(a), shows that the secondary tight-link remains invisible. In (b), related to 2.7(b), it is shown that *TOPP* is able to detect secondary tight-links if they have the appropriate order.

Due to the fact that *TOPP* is based on *PP*, its main problem is the dependence on the cross-traffic characteristics. As shown in Appendix B.2, the packet size P should be large enough to mitigate this drawback. Nevertheless, it can still have low accuracy in highly utilized paths since it is more likely that cross-traffic packets queue in front of the first packet of each pair and make it wait for the second packet, decreasing the separation between them.

The detection of the turning points of the graph is also critical. Many statistical methods have been proposed, such as inflection point obtained from the second derivative [18] or point with minimum variance [19], but their accuracy always depends on the noise of the measured cloud of points.

Self-Loading Periodic Streams

Self-Loading Periodic Streams (SLoPS) [20] is another good example of ItP, implemented on the publicly available tool *pathload* [21]. The source sends a stream of K packets of size P and with a time separation T_{in} between them, resulting in a constant input rate $R_{in} = P/T_{in}$ (see Figure 2.9). The queuing delay of all packets is measured and statistically analyzed at the destination.

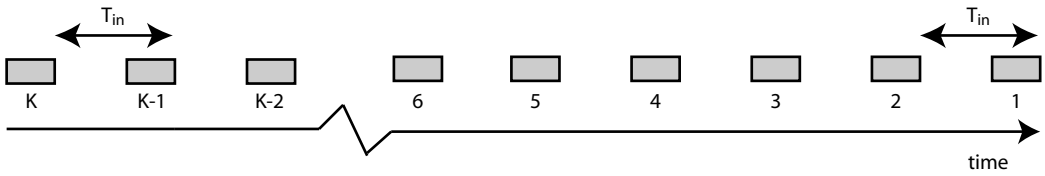

Figure 2.9: *SLoPS* stream formed by K packets separated a constant time gap T between them.

If R_{in} is greater than the ABw of the path, the tight-link will be momentarily overloaded and the queuing delay of the probing packets will begin to increase. When $R_{in} < A$, the queuing delay trend remains around zero. As a special feature, *SLoPS* detects a variation of the ABw within a stream when the queuing delay trend does not show a clear increasing or non-increasing trend, reporting it as a *gray region*. Figure 2.10 is a simulation example of these three cases.

The method sends a fleet of streams with the same rate so as to make sure it has taken the right decision. The iterative algorithm performs a *binary search* of the ABw [21] starting from a given interval and adapting the rate of each stream of the next fleet, by changing P, according to the last detected trend.

With regard to the advantages, *SLoPS* takes into account the variability of the ABw during the probing time by giving an interval of variation. Moreover, a binary search is more efficient than a linear one. Nonetheless, the input rate adjustment requires feedback from destination, which for example can be done via a TCP/IP connection.

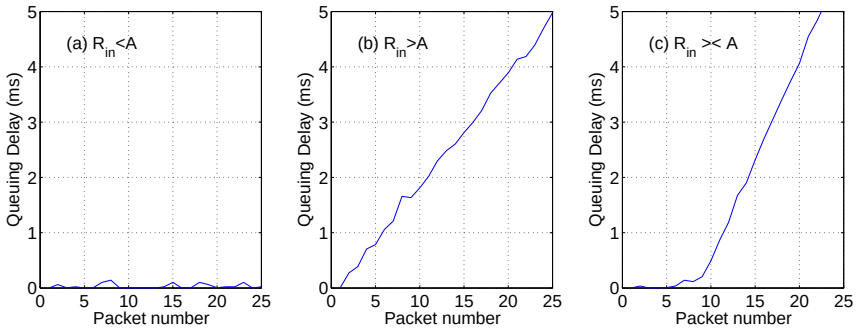

Figure 2.10: *SLoPS* simulation example of a non-increasing trend (a), an increasing trend (b) and a grey region (c).

PathChirp

PathChirp [6] is a publicly available tool to estimate the end-to-end ABw. The method consists of sending streams of exponentially spaced packets, usually called *chirps*, and using the queueing delay for estimating the ABw. Figure 2.11 illustrates a chirp with a spread factor γ, which sets the time gap between two consecutive packets. The instantaneous rate of a certain stream can be defined as

$$R_k = \frac{P}{T_k} = \frac{P}{\delta\gamma^k} \qquad k = 0, \ldots, K-2 \tag{2.21}$$

where T_k is the instantaneous gap and δ is the gap between the two closest packets of the stream. Note that the instantaneous rate changes within each stream.

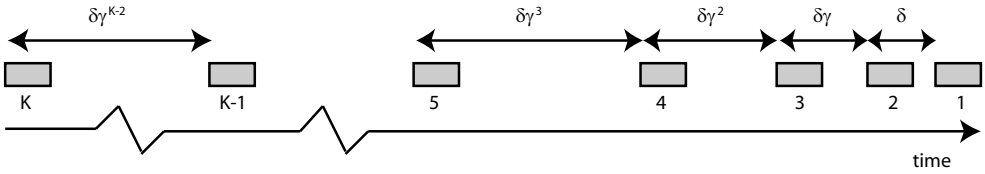

Figure 2.11: *PathChirp* stream, usually called *chirp*.

Assuming a fluid cross-traffic model, the queue delay will increase if the instantaneous rate is greater than the ABw, else there will be no queuing delay,

$$\begin{cases} q_k < q_{k+1} & \text{if } R_k > A \\ q_k = 0 & \text{if } R_k \leq A \end{cases} \tag{2.22}$$

Therefore, an ABwE would be the first instantaneous rate that increases the queueing delay. However, the cross-traffic shows a bursty behavior, which produces *excursions* as Figure 2.12 illustrates. These excursions tend to stabilize until the instantaneous rate exceeds the ABw, in which case the queuing delay will monotonously increase. By analyzing each excursion, *pathChirp* determines an estimate of the ABw per packet [6]. Then, the ABw per stream is obtained from a weighted average of the different estimates per packet. In order to get a more accurate estimation, a fleet of streams is sent. The estimate per fleet is calculated as an average of the different estimates per stream.

PathChirp has a fixed probing time set by the spread factor, the stream length and the packet size. Besides, *pathChirp* does not need to iterate as it gives an estimation every fleet. However, giving a single value, instead of a range of variation, is considered as drawback by some authors [8] due to

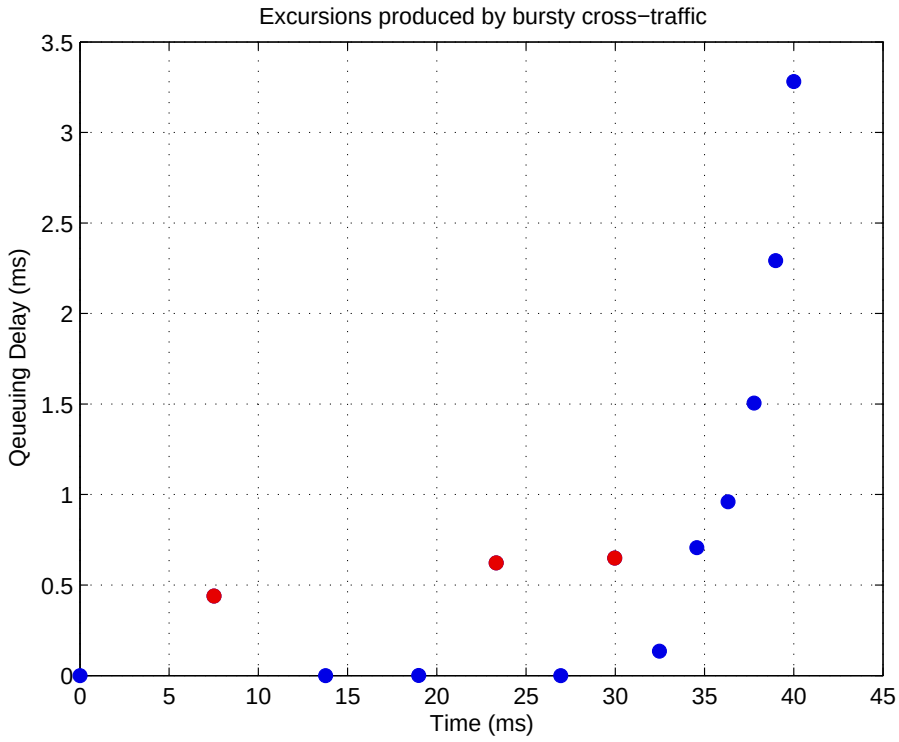

Figure 2.12: Simulation of *pathChirp* excursions. The 2^{nd}, 5^{th} and 7^{th} points (in red) represent the peak of excursions produced by bursty cross-traffic.

the statistical behavior of the ABw. Although *pathChirp* gives an estimation without iterating, it is not considered as DP since it does not need the capacity of the tight-link and it is based on measuring queuing delays produced by self-induced congestion as ItP.

2.3.3 Mixed Techniques

There are some methods that do not clearly fall into either DP or ItP, but use a combination of them instead. One such example is called *Initial Gap Increasing (IGI)* [22], which uses an iterative algorithm to estimate the cross-traffic, so it needs the capacity of the tight-link in order to get an estimation of the ABw.

The method is based on sending a train of packet pairs and measuring the output gap within each pair (see Figure 2.13). The method studies the relationship between this gap variation and the cross-traffic. As a result of the bursty behavior of the cross-traffic, not all the packet pairs will suffer an

increasing gap. Only the packets with increasing gap contribute to estimate the cross-traffic [22]. The amount of cross-traffic during the probing time T_p is calculated as the difference in time between the increasing output gaps and the transmission delay as the next equation shows

$$X = C_t \sum^{k^+} \left(T^+_{out} - \frac{P}{C_t} \right) \tag{2.23}$$

where k^+ is the number of packets with an increasing gap.

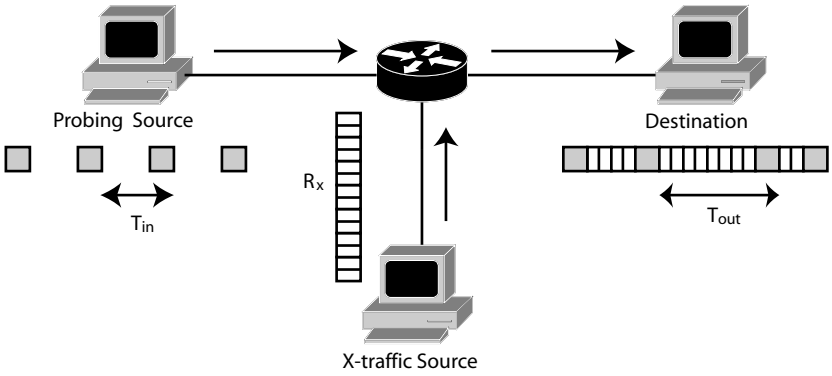

Figure 2.13: Initial Gap Increasing model, where the output gap increases due to cross-traffic.

Therefore, the cross-traffic rate is computed as the ratio between the amount of cross-traffic and the probing time

$$R_x = \frac{X}{T_p} = C_t \frac{\sum^{k^+} \left(T^+_{out} - \frac{P}{C_t} \right)}{\sum^{k^+} T^+_{out} + \sum^{k^=} T^=_{out} + \sum^{k^-} T^-_{out}} \tag{2.24}$$

where $k^=$ is the number of packets whose gap remains constant and k^- is the number of packets showing a decreasing gap.

The key of the method lies in finding the most suitable initial gap. A very small gap means a great input rate, which could overload the network and would not be useful to estimate the cross-traffic. The method uses an iterative algorithm, which consists of increasing the initial gap until the difference between the sum of input and output gaps of the train is as small as the desired accuracy. At that point, Equation 2.24 is applied to estimate the cross-traffic. Finally, the ABw is obtained using Equation 2.11.

The main drawback of this technique is that it is necessary to know the capacity of the tight-link. Besides, it is required feedback from the destination to adapt the initial gap during the probing time. Despite these disadvantages, the method models very intuitively the effects of cross-traffic over a packet pair.

2.4 Summary

The term bandwidth has been quite often imprecisely used. This chapter differentiates between capacity, which is related to the maximum transfer rate of a hop, and *Available Bandwidth (ABw)*, which refers to the unused capacity of a hop. The scope of this chapter is the study of the most representative *ABw Estimation (ABwE)* techniques.

Direct Probing (DP) techniques get an estimation of the ABw every stream by either measuring the output rate or estimating the cross-traffic rate. The canonical example of these techniques is *Delphi*, which assumes a complex multi-fractal model to estimate the cross-traffic. The main advantage is the real-time adaptation of this model to the current traffic. Nevertheless, it requires previous knowledge of the capacity of tight-link.

Iterative Probing (ItP) techniques estimate weather the input rate is larger than the ABw or not in each iteration. The main advantage is that the capacity of the tight-link is not required. The most important methods are:

- *TOPP* sends streams of packet pairs, uniformly increasing their input rates each iteration. The rate is changed by modifying the input gap of each pair. The ABw is estimated as the maximum input rate that is not larger than the output rate. It is also able to estimate the capacity of the tight-link and the ABw of secondary tight-links.

- *SLoPS* sends streams of equally spaced packets. Instead of changing the input rate linearly as *TOPP*, it performs a binary search. The rate varies by modifying the packet size. It takes into account the variability of the ABw by giving a range of variation, rather than a single value.

- *PathChirp* sends streams of exponentially spaced packets called *chirps*, so the instantaneous input rate changes. Only one iteration is needed to get an estimation of the ABw, since it probes the network with different input rates in each stream.

IGI is a technique that shows characteristics from both DP and ItP Techniques. It is based on sending a train of packet pairs, whose gap iteratively increases until the input gap is equal to the output gap. At that point, it estimates the cross-traffic. The main drawback is that the capacity of the tight-link must be known.

In general, all these techniques have some constraints related to their application in real networks:

- A practical implementation means software being installed in both the source and the destination, but the latter may not be accessible.
- Some of the methods, such as *SLoPS* and IGI, require feedback from the destination so as to adapt their parameters during the probing time.
- IGI and Delphi need a prior estimation of the tight-link capacity. Provided that current Capacity Estimation techniques give a value for the capacity of the narrow-link [8], the use of IGI or Delphi implies assuming that the narrow-link and the tight-link are the same. This assumption may not hold, as Figure 2.2 exemplifies.
- Estimating the ABw with Equation 2.11 may lead to an extra error if both the cross-traffic rate and the tight-link capacity have been estimated.

On the other hand, in order to simplify the mathematical complexity of the model that describes the delay packets suffer from the source to the destination, these techniques are based on assumptions that may not hold:

- *First-In/First-Out (FIFO)*: all methods assume FIFO queuing, while actual networks implement traffic prioritization to guarantee a certain QoS, which can lead to estimation errors.
- *Fluid cross-traffic model*: real cross-traffic shows self-similarity and burstiness [15]. The only method of all the studied techniques that takes into account both traffic properties is *Delphi*. The rest of the methods initially assume a fluid cross-traffic model and try to mitigate the effects of burstiness using different statistical procedures.

Chapter 3

Implementation of Iterative Probing Techniques

Techniques that require previous knowledge of the tight-link capacity are not analyzed in this report since it is not possible to determine such capacity without assuming that the tight-link and the narrow-link are the same, which leads to errors [8]. Therefore, the chapter is focused on ItP techniques, specifically *TOPP*, *SLoPS* and *pathChirp*, studying the unification and adaptation of their parameters for an easier comparison between them. First of all, it is necessary to define some structural concepts. As Figure 3.1 illustrates, a set of packets forms a *stream* or *train* and a set of streams constitutes a *fleet*.

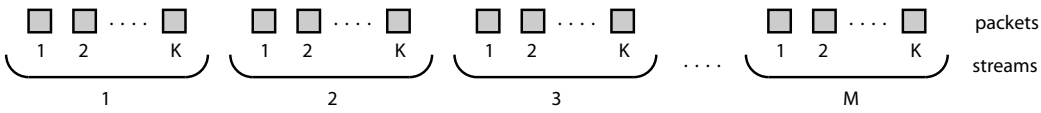

Figure 3.1: A fleet is formed by M streams. Each stream has K packets.

The statistical conditions of the ABw process are fixed by the *time-scale* τ and the *number of samples* N [8]. The time-scale is the time a stream interacts with the cross-traffic in a certain moment, i.e. the stream duration. An input rate is set every two consecutive packets of a stream (or every packet pair in *TOPP*). A sample of the ABw is obtained as a result of checking whether such input rate is larger than the ABw or not. So, the number of samples is related to the number of packets per stream. In order not to vary the statistical characteristics during a probe, both τ and N are established to be constant.

3.1 Train Of Packet Pairs

The objective of this section is to describe how to implement *TOPP* method for the end-to-end ABwE. Additional *TOPP* features, such as the detection of secondary tight-links, may be taken into consideration since they might modify the performance of the model.

3.1.1 TOPP Algorithm

The basics of *TOPP* is very simple, as mentioned in Section 2.3.2. Its main complexity lies in the detection of the turning points of the *segmented linear model* (see Figure 2.7). There are two steps in the *TOPP* algorithm: an iterative stage during which all the probing packets are sent and an analysis state where the measurements are evaluated.

Packets Scheduling

Concerning the iterative algorithm, the method attempts to estimate the ABw within a fixed interval $[R_{min}, R_{max}]$. It sends M trains of packet pairs per iteration and it increases the input rate of the pairs each iteration. Although [16] states that the method sends only one train per iteration, sending $M > 1$ trains could lead to a more accurate estimation by averaging the results.

TOPP sends $K/2$ packet pairs per train and thus gets $N = K/2$ samples of the ABw during a stream. Therefore, the time-scale is determined by the stream duration (see Figure 3.2),

$$\tau = T_{in}\frac{K}{2} + T_{pp}\left(\frac{K}{2} - 1\right) \tag{3.1}$$

As the probe packet size is fixed, the parameter that determines the input rate of each pair is T_{in}. In order to keep τ constant, the value of T_{pp}^{i} must be adapted for every rate,

$$T_{pp}^{i} = \frac{\tau - T_{in}^{i}\frac{K}{2}}{\frac{K}{2} - 1} \qquad i = 0, \ldots, I-1 \tag{3.2}$$

where I is the total number of iterations of the method. Figure 3.2 represents all the scheduling of the packets. A non-intrusiveness gap T_{NI} between trains is inserted to avoid overloading the network and to keep the results obtained from different trains uncorrelated. Also from the figure, it can be distinguished that when T_{in} is maximum, T_{pp} has its minimum value and vice versa.

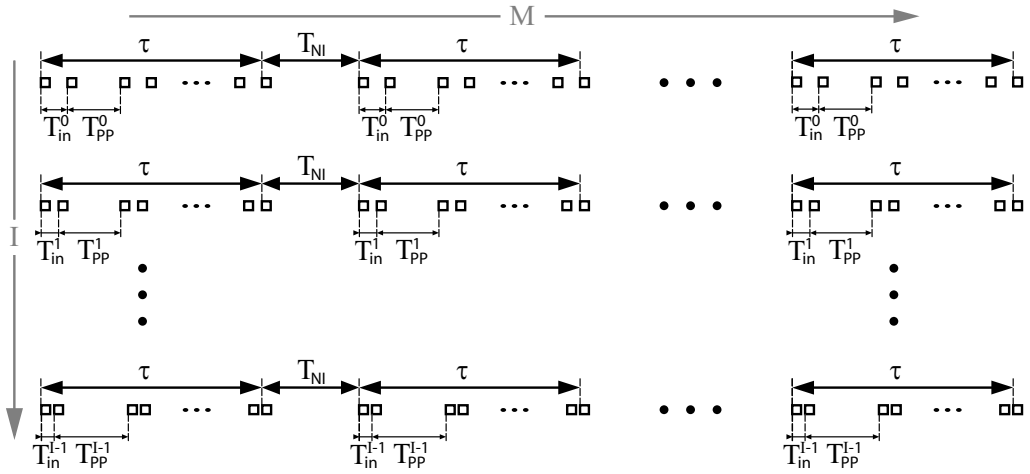

Figure 3.2: *TOPP* sends I fleets of M streams separated a non-intrusiveness gap T_{NI}. T_{pp} is adapted for each T_{in} so as to keep τ constant.

As it is mentioned in Section 2.3.2, T_{pp}^{min} should be chosen for the pairs not to share the same queue. However, it is complex to determine an optimal value independent on the kind of traffic and the network topology. In practice, it is set as a percentage η over the maximum input gap

$$T_{pp}^{min} = (1 + \eta)T_{in}^{max} \tag{3.3}$$

where $\eta > 0$ to keep the rate between two packet pairs, fixed by T_{pp}, below the current input rate. For a given τ and R_{min}, η should be as large as possible to avoid the influence of a packet pair over the next one. However, if η increases, the number of samples per stream decreases as it can be deduced from figure 3.2.

The input rate is increased by a resolution factor ω every iteration, thus the input rate at iteration i is

$$R_{in}^{i} = R_{min} + i\omega \tag{3.4}$$

T_{in}^{i} can be obtained using Equation 2.13. The required number of iterations to achieve a resolution ω in the given interval is obtained from

$$I = \frac{R_{max} - R_{min}}{\omega} \tag{3.5}$$

It is easy to see that the input rate R_{in}^{i} is always below R_{max}. Once all the trains have been received, the destination analyzes all the measured output gaps T_{out}^{i}.

Data Analysis

The data processing of *TOPP* can be divided into three phases, all of them performed by the destination agent:

- The $K/2$ output gaps belonging to each train are averaged to get the time gap per stream. Then, the output rate per stream is obtained by means of Equation 2.15. The M different measured output rates are averaged to get a single value per iteration (R^i_{out}).
- As Figure 3.3 shows, Equation 2.19 is used to draw the cloud of points (in blue), which is statistically analyzed to estimate the segmented function (in red) that best fits the cloud. Different methods exist to calculate the turning point, as described in Appendix D. The minimum variance method is chosen here as it is the most robust.
- Finally, *TOPP* takes the closest R^i_{in} to the calculated turning point as the ABw value.

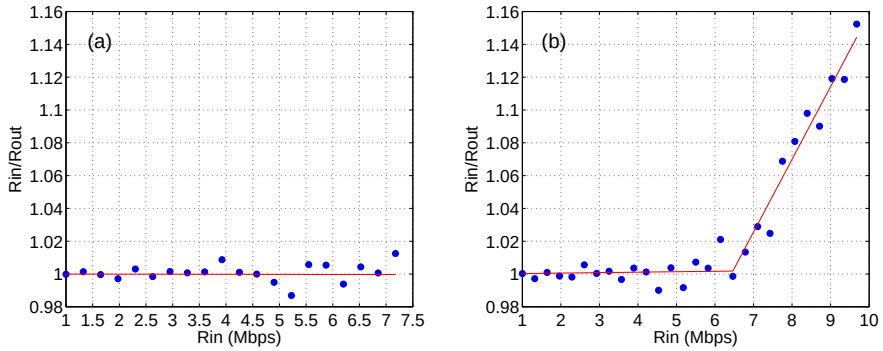

Figure 3.3: *TOPP* simulation example showing the effects of interval borders for $A = 6.7Mbps$ and $\omega = 0.32Mbps$. In (a), $R_{in} \in [1, 7.5]Mbps$ and $I = 20$. In (b), $R_{in} \in [1, 10]Mbps$ and $I = 28$.

3.1.2 TOPP Constraints

TOPP method has some constraints that are related to its performance and efficiency.

Resolution vs Probing Time

As Equation 3.5 denotes, the better the resolution is desired, the larger the number of iterations is required, and so the longer time the whole process

takes. Real-time applications, such as admission control or handover decisions, need a very fast ABwE, so a trade-off should be set depending on the requirements of the implementation.

Interval Borders

The algorithm will not have enough points to estimate properly the right line if turning point is too close to R_{max}. As a consequence, false turning points may be detected if there is enough noise in the measurements resulting in an underestimation of the ABw. Measuring beyond the desired R_{max} could be used to reduce the errors, although it implies to increase the number of iterations to keep the same resolution. Figure 3.3 illustrates a simulation example of this problem.

Cross-traffic Packet Size

This problem is inherited from PP. Its causes, as explained in Appendix B.2, can deeply affect the accuracy. $P \geq 800Bytes$ has shown to give better results in some Internet experiments [23], so large packets should be used. Besides, a statistical analysis using histograms, instead of making a mean to the measured time gaps, could be considered to improve performance of the method.

Secondary Tight-links

The proposed algorithms assume there is only one turning point. As a result, the presence of secondary tight-links may have an impact to *TOPP* performance as Figure 3.4 proves. For the sake of solving this problem, an iterative procedure based on some empirical thresholds for both slopes can be used to determine if a first estimation is reliable or a new estimation should be performed. The new estimation would only use the left side of the previous cloud of points. The algorithm would finish when the first slope is horizontal enough and the second one is positive. This approach, however, may lead to severe underestimation of the ABw since the useful part of the cloud of points decreases every iteration.

3.2 Self-Loading Periodic Streams

In this section, the practical aspects of *SLoPS* are described. It is shown that the method is more complicated than *pathChirp* and *TOPP* due to the

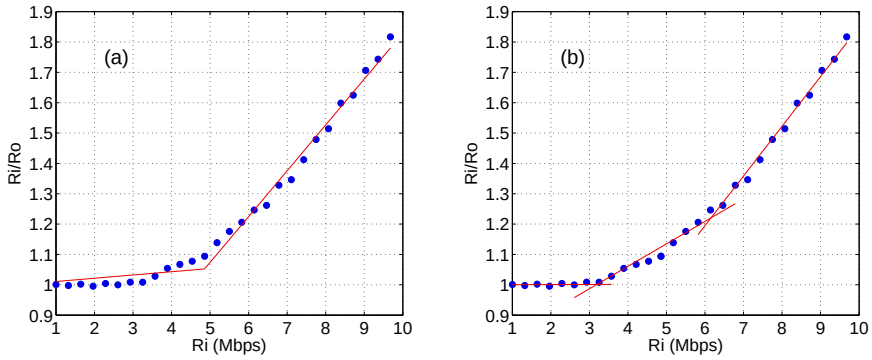

Figure 3.4: *TOPP* simulation examples of secondary tight-links effects. In (a), the ABw is estimated as 5Mbps when it is 3Mbps due to a secondary tight-link at 6Mbps. (b) shows the ideal detection.

fact that it requires feedback from the destination to adapt the sending rate in each iteration.

3.2.1 SLoPS Algorithm

As described in Section 2.3.2, *SLoPS* sends a fleet of streams at a constant rate to estimate whether the input rate is larger than the ABw or not. Depending on this decision, a new fleet with a different rate is sent to give a more accurate range of variation of the ABw. Since the stream duration is the time that the cross-traffic interacts with the probing packets to get an estimation, the time-scale is considered as the stream time length (see Figure 2.9),

$$\tau = (K-1)T_{in} \tag{3.6}$$

The number of samples taken in each stream is related to the different measured queuing delay variations, so $N = K - 1$. In order to detect this queuing delay variation trend of a stream, it is only necessary to measure the OWD, avoiding synchronization between the source and the destination. As Equation 2.9 proves, if and only if the OWD variation from two consecutive equally sized packets increases (or decreases), the queuing delay variation increases (or decreases) as well,

$$\Omega_{k+1} \gtrless \Omega_k \iff q_{k+1} \gtrless q_k \tag{3.7}$$

The K probing packets are divided into $\Lambda = \sqrt{K}$ groups. Then, the averaged OWD $\hat{\Omega}_l$ of each group is calculated and two criteria are applied. The *Pairwise Comparison Test* (S_{PCT}) measures the fraction of consecutive

packets whose OWD variation is increasing and the *Pairwise Different Test (S_{PDT})* measures the strength of the end-to-end OWD variation,

$$S_{PCT} = \frac{\sum_{l=2}^{\Lambda} \Phi(\hat{\Omega}_l > \hat{\Omega}_{l-1})}{\Lambda - 1} \tag{3.8}$$

$$S_{PDT} = \frac{\hat{\Omega}_\Lambda - \hat{\Omega}_1}{\sum_{l=2}^{\Lambda} |\hat{\Omega}_l - \hat{\Omega}_{l-1}|} \tag{3.9}$$

where $\Phi(X) = 1$ if X is true, otherwise $\Phi(X) = 0$.

If the delay variation is independent, then $S_{PCT} = 0.5$ and $S_{PDT} = 0$. In [20], it is proposed, based on empirical results, that if $S_{PCT} > 0.55$ and $S_{PDT} > 0.4$ then the stream is considered as increasing and labeled as type-I, otherwise it is labeled as type-N. In order to make the estimation more accurate, a fleet of M streams with the same rate is sent, with a *non-intrusiveness gap* T_{NI} in between. In general, this gap avoids overloading the network and lets the cross-traffic stabilize after the previous stream. Specifically in *SLoPS*, the source must wait for the destination to analyze the OWD trend and to send the result back.

If a fraction f of the M streams is type-I, the fleet shows an increasing trend, so $R_{in} > A$. Else, if a fraction f of the M streams is type-N, the fleet shows an non-increasing trend, so $R_{in} < A$. Otherwise, the fleet is considered to be in a gray-region, so $R_{in} \bowtie A$. In [20], it is suggested to use $f = 70\%$. The input rate of the next stream is adapted, by changing P and keeping T_{in} constant (see Equation 2.13), according to detected trend. If there were no gray-regions, the iterative algorithm would do a *binary search*:

- If $R_{in} > A$, then $R_{max} = R_{in}$, else $R_{min} = R_{in}$.
- The iterations finish when $R_{max} - R_{min} < \omega$, where ω is the desired resolution.

However, real cross-traffic can produce gray-regions, which makes it more complex to determine the new input rate. Appendix E.2.1 gives a sample implementation of an algorithm that takes the gray-regions into account. Note that if the initial measurable range $[R_{min}, R_{max}]$ is unknown, it can be set with different algorithms. One of them, based on an exponential search, is explained in Appendix E.2.2.

3.2.2 SLoPS Constraints

SLoPS algorithm has not only constraints which are a direct consequence of its performance, but also related to link-2 effects.

Resolution vs Probing Time

As it happens in *TOPP*, the resolution is inversely proportional to the number of iterations and hence, to the probing time. Although, the number of iterations in *SLoPS* is usually quite lower than in *TOPP* as a result of the binary search (see Figure 3.5), it is unpredictable due to gray-regions.

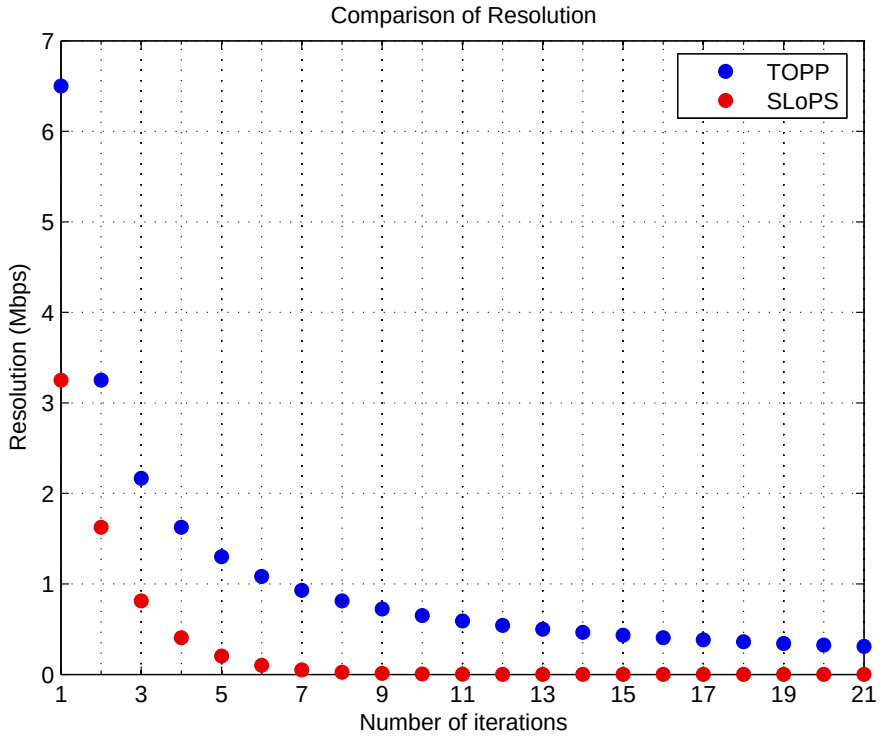

Figure 3.5: Resolution vs iterations for a measurable range of $[1.0, 7.5]Mbps$. *SLoPS* needs much less iterations than *TOPP* to get the same resolution.

In case of no gray-regions, the interval is halved each iteration, so the number of iterations for a desired resolution ω is

$$I \geq \log_2 \left(\frac{R_{max} - R_{min}}{\omega} \right) \tag{3.10}$$

Range of Measurable Available Bandwidth

As it is mentioned in Section 2.1.1, the packet size has certain limitations so as to avoid layer-2 effects (P_{min} set to 200 Bytes) and fragmentation (P_{max}

set to 1500 Bytes). Therefore, the measurable range of ABw is

$$R_{min} = \frac{P_{min}}{T_{in}} \qquad R_{max} = \frac{P_{max}}{T_{in}} \tag{3.11}$$

In order to keep the same statistical conditions during the probing time, the time-scale and the number of samples should not change, and hence, the input gap neither (see Equation 3.6). This fact, together with the packet size constraint, makes the measurable ABw range be as follows

$$R_{max} = 7.5R_{min} \tag{3.12}$$

3.3 PathChirp

This section includes the details of *pathChirp*, whose basics are described in Section 2.3.2.

3.3.1 PathChirp Algorithm

At first sight, *pathChirp* implementation can be considered as similar to *TOPP* implementation, in the sense that the probing phase and the analysis phase are pretty well differentiated. Nonetheless, the exponential structure of the stream allows *pathChirp* to give an estimation on a single iteration [6].

Packets Scheduling

As the previous methods, *pathChirp* attempts to estimate the ABw within a an interval $[R_{min}, R_{max}]$, fixed by the packet size and the scheduling characteristics. The probing phase consists of sending M streams, called *chirps*, formed by K exponentially spaced packets. A non-intrusiveness gap T_{NI} between streams is also inserted in the scheduling for the same reasons as in *TOPP*. Therefore, the time that *pathChirp* interacts with the cross-traffic is the stream duration and is expressed as

$$\tau = \sum_{k=0}^{K-2} T_k = \sum_{k=0}^{K-2} \delta\gamma^k = \frac{1-\gamma^{K-1}}{1-\gamma}\delta \tag{3.13}$$

In [6] the spread factor $\gamma = 1.2$ is set. The number of samples N is determined by the number of different rates sent in a stream (see Equation 2.21), hence, $N = K - 1$. The exponential spacing together with the packet size set the maximum range that can be estimated,

$$R_{min} = \frac{P}{\delta\gamma^{K-2}} \qquad R_{max} = \frac{P}{\delta} \tag{3.14}$$

Analysis Phase

As a consequence of the chirp structure, a characteristic of *pathChirp* is that it gets a value of the ABw from every stream, but as usual, that procedure is repeated to improve its accuracy. The method finishes when M streams have been sent and an estimation of the ABw has been calculated.

As mentioned in Section 2.3.2, *pathChirp* makes use of the excursions to estimate the ABw by measuring the queuing delay. In order to avoid synchronization between the source and the destination, as in *SLoPS*, it is enough to study the OWD trend (see Equation 3.7) so as to determine whether an excursion takes place or not. To explain how an excursion is detected, let define a *starting point* of an excursion as any point that shows an increasing delay, meanwhile an *ending point* is determined by

$$\Delta\Omega = \Omega_t^m - \Omega_s^m < \frac{\Delta\Omega_{max}}{F} = \frac{\max_{s\leq k\leq t}\{\Omega_k^m - \Omega_s^m\}}{F} \tag{3.15}$$

where F is the decrease factor and m denotes the current stream. If $t - s \geq L$, then it is considered as an excursion, where L is the excursion length threshold. Figure 3.6 represents both concepts.

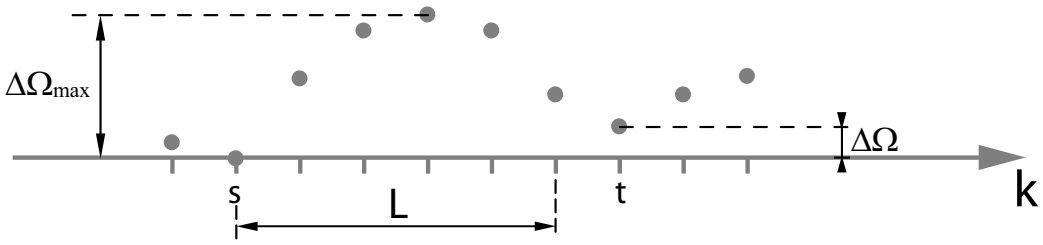

Figure 3.6: If $(\Delta\Omega_{max} > F\Delta\Omega)$ and $(t - s \geq L)$ then an excursion is detected with starting point s and ending point t.

In [6], $F = 1.5$ and $L = 5$ are set by default, and $F = 6$ and $L = 3$ are proposed for Internet traffic. The idea is to detect all the possible excursions from the OWD profile such as the one shown in Figure 2.12. Appendix E.1.1 includes a practical implementation of the excursions detection algorithm.

Once all the excursions have been uncovered, the method, implemented in Appendix E.1.2, estimates the ABw per packet E_k^m. Three different cases may take place:

- If packet k belongs to an excursion and shows an increasing trend, then $E_k^m = R_k^m$.

- Else, if packet k belongs to an excursion that does not finish, then $E_k^m = R_s^m$, where s is the starting point of this excursion.
- Otherwise, packet k does not belong to an excursion or shows a decreasing trend. Then, $E_k^m = R_{s_l}^m$, where s_l is the starting point of the last excursion when such excursion does not finish, and $K-1$ when it does.

If the excursion does not finish, R_k might be greater than C_t and, hence, the ABwE would be greater than C_t. Therefore, the method can be considered as conservative since it decides to take the starting point of the excursion in this case. In the next step of the algorithm, the ABw per stream D^m is obtained from a weighted average of the different estimates per packet,

$$D^m = \frac{\sum_{k=0}^{K-2} E_k^m T_k^m}{\sum_{k=0}^{K-2} T_k^m} \tag{3.16}$$

Lately, in order to get a more accurate estimation, the estimate per fleet is calculated as an average of the different estimates per stream,

$$A = \frac{\sum_{m=0}^{M-1} D^m}{M} \tag{3.17}$$

3.3.2 PathChirp Constraints

Although the use of chirps is an advantage since each of them sample the whole interval $[R_{min}, R_{max}]$, they introduce some limitations in resolution and in the freedom of what to measure. On the other hand, it is hard to find optimal values of F and L for any kind of traffic.

Variable Resolution

Due to the exponential spacing of packets within a stream, the different tested rates R_k^m tend to concentrate around the low part of the measurable ABw range. Therefore the resolution is not the same within the whole range. This fact may result in worst estimations for values near the upper limit of the interval.

Limited Freedom

Another constraint derived from the chirp structure is that the different rates have always an exponential trend, without letting freedom to choose the intermediate rates of the interval. For instance, if the number of samples and

the time-scale were needed to be fixed, then, for a given γ, the resolution and the ratio between R_{max} and R_{min} would be fixed. As a conclusion, *pathChirp* has much less freedom in resolution than *TOPP* and *SLoPS* since the resolution of these two methods only depends on the number of iterations.

Thresholds Choice

In *pathChirp*, F and L are not clearly determined. In fact, depending on the network and cross-traffic characteristics, different parameters are used [6]. As Figure 3.7 illustrates, if L increases or F decreases too much, mostly of the possible excursions are discarded, so there is an overestimation. Otherwise, an underestimation is produced.

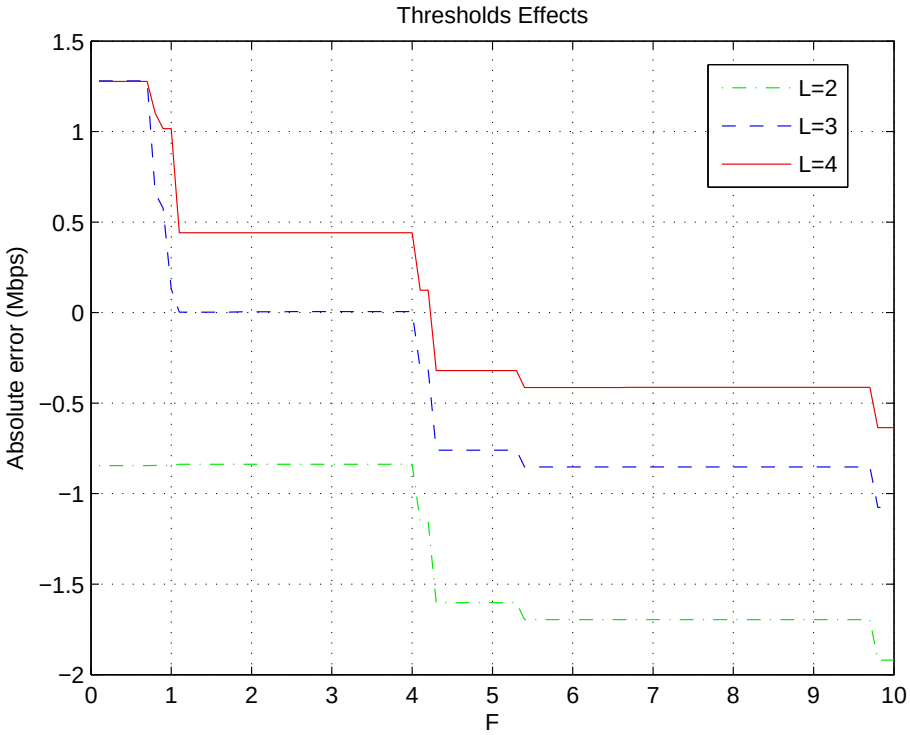

Figure 3.7: Simulation example of a single-hop path with CBR traffic. When F increases or L decreases, *pathChirp* tends to underestimate and vice versa.

3.4 Efficiency Parameters

After having detailed a practical implementation of the techniques of interest, this section describes the different parameters used to characterize such methods in terms of efficiency.

3.4.1 Probing Load

Each method sends M streams composed of K packets. The whole procedure is repeated I iterations. Therefore, the probing load can be defined as

$$L_p = K \cdot M \cdot I \cdot P \tag{3.18}$$

In *pathChirp* and *TOPP*, all the parameters of Equation 3.18 are fixed regardless of the simulation. However, the number of iterations and the packet size in *SLoPS* vary from one simulation to another. One possible criteria for calculating the probing load is to consider P as the average packet size, i.e. 850 Bytes, and I determined by Equation 3.10.

3.4.2 Probing Time

The scheduling structure of *TOPP*, shown in Figure 3.2, can be used to determine the probing time, which mainly depends on the time-scale and the non-intrusiveness gap. Generalizing for any method, the probing time is

$$T_p = M \cdot I(\tau + T_{NI}) - T_{NI} \tag{3.19}$$

3.4.3 Average Probing Rate

The average probing rate is used as a measure of the intrusiveness of a method. It is calculated as the ratio between the load and the probing time,

$$R_p = \frac{L_p}{T_p} \tag{3.20}$$

Chapter 4

Simulation of Iterative Probing Techniques

To the authors' knowledge, there are no publications that compare the different techniques under similar statistical conditions [8]. This fact could make the results of previous work unreliable since the methods were experiencing different cross-traffic statistical characteristics. Simulations are used, not only to verify the functionality of the methods, but also to compare them and draw conclusions in order not to waste resources in a practical implementation.

This chapter is focused on the simulation of the chosen ItP techniques, i.e. *TOPP*, *SLoPS* and *pathChirp*, following the requirements established in Chapter 1. First, the simulation scenario is described. Then, a viability study of statistical comparison is performed and the parameters of the three methods are adjusted. The chapter finishes with the results and conclusions of the simulations, together with a discussion of which of the three techniques should be more useful to be implemented on a packet-switched mobile transport network.

4.1 Simulation Scenario

The simulation scenario consists of the network topology and the cross-traffic modeling. The sooner establishes the relationship between the different nodes of the network, as well as the way the cross-traffic flows through the network, whereas the latter argue the use of certain kinds of traffic.

4.1.1 Network Topology

The networking parameters, such as the ABw, should be under total control in order to compare the three methods in terms of accuracy. For that reason, it is also crucial the way the probe traffic and the cross-traffic are inserted into the simulation network.

Cross-traffic Routing

The objective of the topology is to control the capacity and the ABw of each hop so that the tight-link can be located in the desired place. Let define *path persistent cross-traffic* as the one whose packets follow the same path as the probe traffic (see Figure 4.1(a)), while the *one-hop persistent cross-traffic* is the one whose packets only travel through one hop of the path (see Figure 4.1(b)). In the latter, the cross-traffic rate at a hop h is only fixed by the input rate $R_{x,h}$ of the cross-traffic source connected to it, so the ABw per hop is

$$A_h = C_h - R_{x,h} \tag{4.1}$$

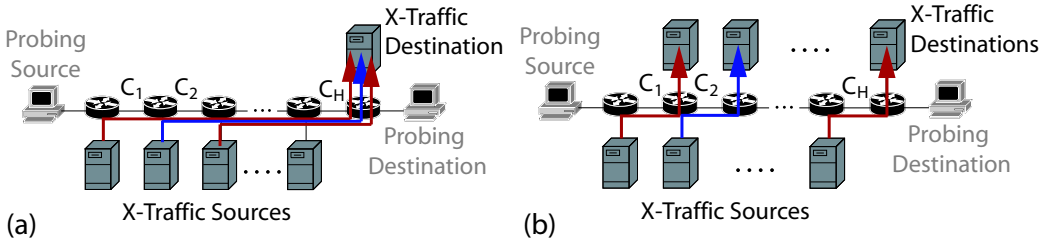

Figure 4.1: Cross-traffic routing examples. In (a), the cross-traffic is path persistent. In (b), the cross-traffic is one-hop persistent.

If path persistent cross-traffic was used, the cross-traffic rate of a hop would depend on the cross-traffic rate of the preceding hops. A certain hop could not have a lower rate than the previous one, which limits the freedom to choose the tight-link. Therefore, one-hop persistent cross-traffic is used in the simulations.

Network Model

Figure 4.2 represents the network model for the simulations, composed of a probing source and a probing destination, i.e. the ABwE agents, joined by an H-hop path. It is based on one-hop persistent cross-traffic routing, so there are H pairs *XS-XD* for the cross-traffic generation. The routers are

set to have an infinite queue size to avoid probing packet losses due to buffer overflow.

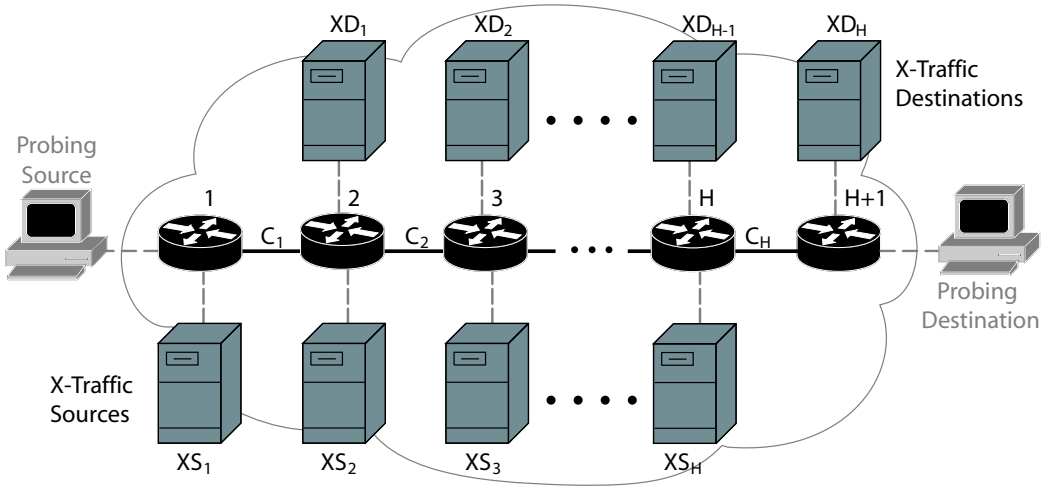

Figure 4.2: Network topology for the simulations. Probing source and destination are joined by an H-hop path. The cross-traffic is one-hop persistent.

Each hop of the path is defined by a capacity C_h, a latency d_h and a cross-traffic source transmitting at a rate $R_{x,h}$, where $h = 1, \ldots, H$. The *joining links* (dashed line), which never carry probe packets and cross traffic at the same time, are defined by a capacity C_0 and a latency d_0. Both the capacity of the hops and the joining links are set using standard values for Ethernet networks, provided in Table 4.1. Note that the latency can be arbitrarily chosen since it does not affect the methods.

Technology	**Capacity**
Ethernet	10Mbps
Fast-Ethernet	100Mbps
Gigabit-Ethernet	1Gbps
10Gigabit-Ethernet	10Gbps

Table 4.1: Standard Ethernet Capacities [24].

Another important criteria to decide the best model is the capability to work in multi-hop paths. The proposed simulation topology allows the selection of the number of hops, so that $H > 1$ for multi-hop. Nevertheless, single-hop paths are also studied to determine properly the effects of the different cross-traffic models that are applied.

Network Model Constraint

The time gaps set by the scheduling at the cross-traffic source vary from the time gaps at the end of the joining link if the cross-traffic packets are not equally sized, due to their different transmission delays. However, such constraint can be erased by adapting the scheduling so as the packets arrive at the targeted hop with the desired rate. Suppose that it is wished to set a rate R_x at a certain hop of the path. If the cross-traffic source sends two packets with sizes P_A and P_B respectively leaving an input gap T_{in} in between, the output gap at the end of the joining link is

$$T_{out} = \frac{P_B}{C_0} - \frac{P_A}{C_0} + T_{in} \tag{4.2}$$

assuming the latency remains constant. To achieve such rate at the end of the joining link, the output gap should be $T_{out} = P_A/R_x$, so the necessary input gap is

$$T_{in} = \frac{P_A C_0 - (P_B - P_A)R_x}{R_x C_0} \tag{4.3}$$

Taking into account that $T_{in} \geq P_A/C_0$ and substituting in Equation 4.3, the capacity constraint of the joining link is described by the next expression

$$C_0 \geq \frac{P_B}{P_A} R_x \tag{4.4}$$

The worst situation takes places when $P_A = 40B$ and $P_B = 1500B$, in which case, the joining link capacity should be

$$C_0 \geq 37.5 R_x \tag{4.5}$$

4.1.2 Cross-traffic Models

The methods should be evaluated under different kinds of cross-traffic. The more realistic the traffic model is used in the simulations, the more representative values will be obtained. Appendix C includes the foundations of the cross-traffic models used in the simulations. Their main characteristics and the reason for their use are given in the following lines.

Constant Bit Rate (CBR)

It is the simplest cross-traffic model, based on sending equally sized packets with a constant inter-arrival time. Although it is not the most realistic traffic, it is the closest to the fluid traffic model initially assumed by all three techniques.

Packet Size Distribution CBR (PSD-CBR)

It is based on CBR traffic, but making use of a random packet size distribution obtained from a study of Internet traffic characteristics [25]. The constant inter-arrival time of the packets at each hop of the path is fixed by the average packet size of the model and the cross-traffic rate. It is used to observe the effect of a more realistic cross-traffic PSD on the different methods.

Poisson

In queuing theory, the arrival of requests in a server is usually assumed to be a Poisson process since it is a well-known distribution, which captures accurately this behavior [26]. Poisson traffic model consists of a Poisson source, which sends equally sized packets with an exponential inter-arrival distribution.

4.2 Viability Study of Statistical Comparison

The statistical conditions are fixed by the time-scale τ and the number of samples N. The methods performance is also dependent on the probing packet size P and the measurable ABw range $[R_{min}, R_{max}]$. So, the time-scale can be expressed as

$$\tau = f(P, N, R_{min}) \qquad \text{or} \qquad \tau = f(P, N, R_{max}) \tag{4.6}$$

An initial study has been conducted to see if it is possible to set the parameters of the three methods in such a way that they are working under the same statistical conditions. Figure 4.3 represents such study for $R_{min} = 1Mbps$. In 4.3(a), the time-scale as a function of the number of samples is shown for the three methods, using their optimum packet sizes [27, 20, 6]. There is an extra plot of *TOPP* for the minimum packet size to avoid link layer effects [20]. In 4.3(b) , the maximum input rate[1] of *SLOPS* and *pathChirp* is compared.

From Figure 4.3(a), it can be seen that *SLoPS* and *TOPP* can not be compared for a significant number of samples. *TOPP* is comparable with *pathChirp*, but only if a probing packet size that is very sensitive to cross-traffic is used [27] and $\eta = 0$, which increases the probability of two PPs to be in the same queue. Figure 4.3(b) shows that *pathChirp* and *SLoPS* are not comparable since they measure very different ranges for the same τ and

[1] *TOPP* is not shown since R_{max} does not depend on N.

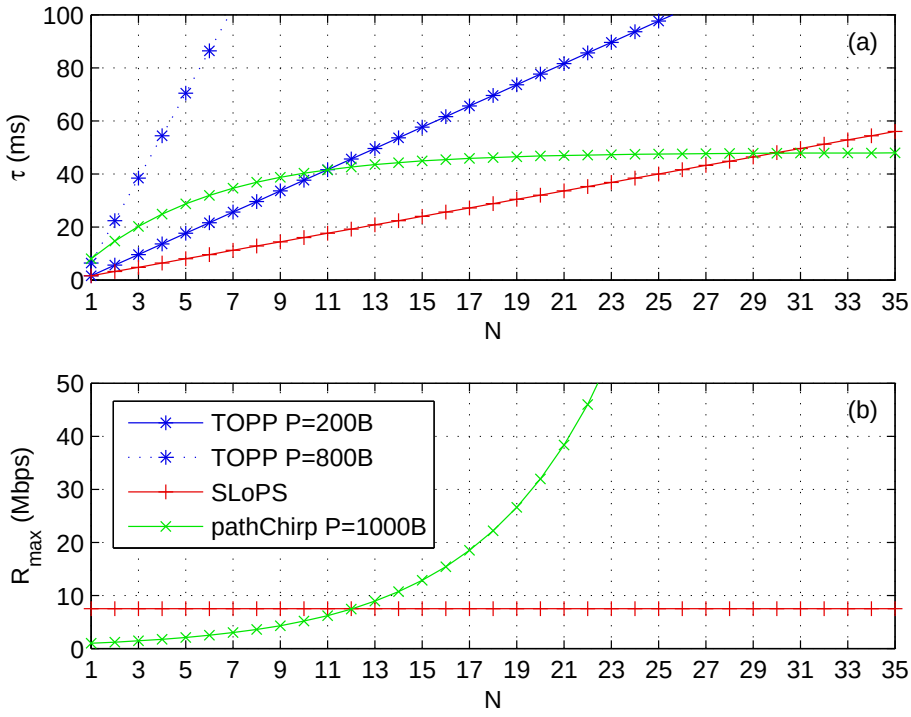

Figure 4.3: Statistical comparison of *TOPP* ($\eta = 0$), *SLoPS* and *pathChirp* for $R_{min} = 1Mbps$.

N. However, this study depends on a given R_{min}. In order to compare the different methods regardless of a given time-scale or a certain measurable ABw range, the previous study can be generalized by using the *High-Low Factor*, explained below.

4.2.1 High-Low Factor Definition

From Equation 4.6, the following relationships, expressed in Bytes, can be worked out for the three methods

$$\begin{aligned} \text{Low-Factor} &\equiv \tau R_{min} = f(P, N) \\ \text{High-Factor} &\equiv \tau R_{max} = f(P, N) \end{aligned}$$

Two methods which measure the same ABw range under the same time-scale have to have the same High-Low Factor. Therefore, a parametric study of both factors with different packet sizes and number of samples enables to decide about the viability of the statistical comparison.

Train of Packet Pairs

As explained in Section 3.1.1, when the input rate is minimum, T_{in} is maximum and, hence, the separation between PPs T_{pp} is also minimum to keep constant the time-scale. Taking into account that the number of samples is $N = K/2$ and $R_{min} = P/T_{in}^{max}$, by using Equations 3.1 and 3.3, the Low-Factor in *TOPP* can be expressed as

$$\tau R_{min} = P\left[(2+\eta)N - (1+\eta)\right] \tag{4.7}$$

where P should be larger than 800 Bytes in order to mitigate the sensitivity to cross-traffic packet size (see Section 3.1.2). Note that the Low-Factor slope is minimum for $\eta = 0$. On the other hand, the maximum input rate does not depend on the time-scale and number of samples, but it is limited by the minimum resolution of the OS. So, there is no upper bound fixed by the High-Factor.

Self-Loading Periodic Streams

In *SLoPS*, the measurable range is fixed by the minimum packet size to avoid layer-2 effects, and the maximum packet size not to produce fragmentation (see Section 3.2.2). Taking into account that the number of samples is $N = K - 1$ and by using Equation 3.6, the Low-Factor and High-Factor are

$$\tau R_{min} = NP_{min} \tag{4.8}$$
$$\tau R_{max} = NP_{max} \tag{4.9}$$

where $P_{min} = 200B$ and $P_{max} = 1500B$.

PathChirp

In Section 3.3.1, the exponential structure of *pathChirp* is described. Considering the number of samples is $N = K - 1$ and by substituting Equation 3.14 in 3.13, the Low-Factor and the High-Factor are obtained as follows

$$\tau R_{min} = P\frac{\gamma - \gamma^{1-N}}{\gamma - 1} \tag{4.10}$$
$$\tau R_{max} = P\frac{\gamma^{N} - 1}{\gamma - 1} \tag{4.11}$$

where $\gamma = 1.2$ and P should be larger than 1000 Bytes in order to get a better performance [6].

4.2.2 High-Low Factor Study

Taking into account the nature of the High-Low Factor equations, which depend on more than one variable, it is required a parametric study. Figure 4.4 shows a graphical representation of such analysis. In 4.4(a), the Low-Factor of *TOPP*, *SLoPS* and *pathChirp* is plotted for their optimum packet sizes. There is an extra plot of *TOPP* using the minimum packet size. In 4.4(b), the High-Factor of *SLoPS* and *pathChirp* is compared.

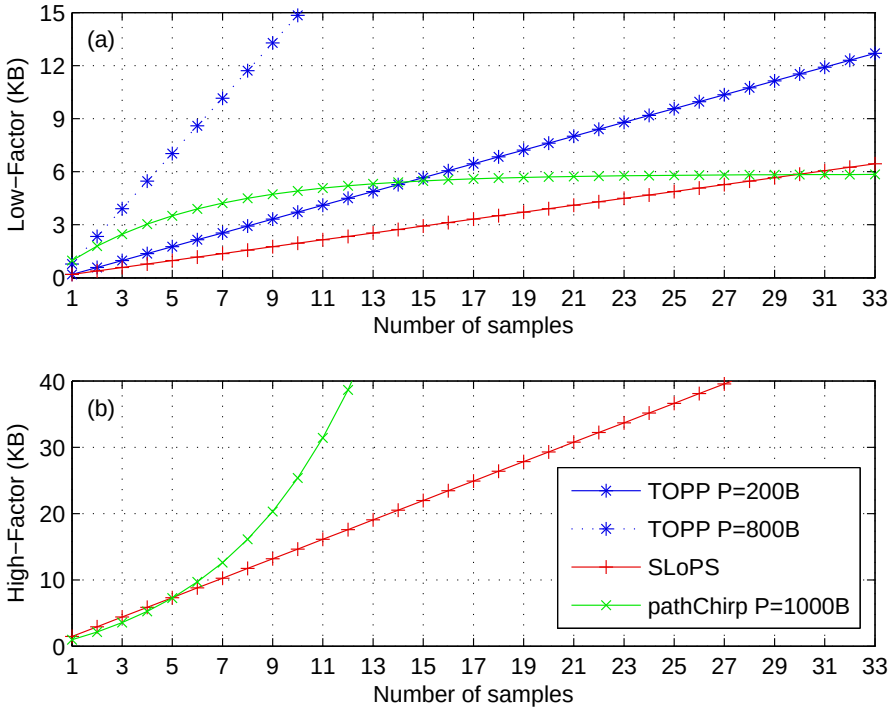

Figure 4.4: *TOPP* ($\eta = 0$), *SLoPS* and *pathChirp* cannot be compared under the same statistical conditions, even changing their optimum parameters.

Figure 4.4 proves that it is not possible to study the performance of *TOPP*, *SLoPS* and *pathChirp* under the same statistical conditions to measure a certain ABw range, even modifying their recommended parameters. Therefore, it is necessary to choose between the time-scale or the number of samples as the common statistical parameter. The time-scale is directly related to probing time, the latter being a determining factor for the field of application of this study. In the following simulations, τ is fixed for all three methods, while N is chosen for each method as large as possible to

measure a similar ABw range without excessively modifying their optimum parameters.

4.3 Adjustment of Parameters

As it is mentioned in the previous section, *TOPP*, *SLoPS* and *pathChirp* are only comparable for a given time-scale if they use a different number of samples. It is desired to measure as wide range as possible to obtain results suitable for different applications and purposes. Nevertheless, *SLoPS* fixes a maximum measurable range (see Equation 3.11) not to modify τ and N during its probing time. Therefore, it is necessary to split the measurable range into a low and a high ABw intervals

$$\begin{aligned} \text{Low Range} &\equiv 1 \text{ to } 7.5\text{Mbps} \\ \text{High Range} &\equiv 8 \text{ to } 60\text{Mbps} \end{aligned}$$

As Section 2.1.2 explains, longer time-scales reduce the variance of the ABw, while shorter ones reduce the probing time, which is a key factor in some applications. In [8], an Internet experiment shows that the variance of the ABw is considerably reduced for time-scales larger than 10ms. As the High-Low Factor determines, a lower ABw require a longer time-scale to keep similar number of samples,

$$\begin{aligned} \text{Low Range} &\equiv \tau = 40ms \\ \text{High Range} &\equiv \tau = 10ms \end{aligned}$$

Each stream is sent M times and the results are averaged so as to improve the accuracy. *SLoPS* [20] sets M to 12 and *TOPP* [16] sends only one stream, whereas in *pathChirp* [6], an optimum M value is not determined. In order to keep the same conditions during the simulations for the three methods, it has been decided to use $M = 10$, although this decision increases *TOPP* probing time.

Another common parameter of all three methods is the non-intrusiveness gap T_{NI}. As a trade-off between the probing time and the network overload, T_{NI} is fixed as large as the time-scale, which reduces the average probing rate by at least 50%,

$$\begin{aligned} \text{Low Range} &\equiv T_{NI} = 40ms \\ \text{High Range} &\equiv T_{NI} = 20ms \end{aligned}$$

Note that T_{NI} in *SLoPS* is fixed as in the other two techniques, without waiting for the timestamps from the destination (see Section 3.2), since the

methods are implemented in a simulator, which instantaneously gives the timestamps.

4.3.1 Low Available Bandwidth Range

Table 4.4 summarizes the values of the simulation parameters for each method for the low ABw range. The reasons for their choice are explained in the following lines.

Train of Packet Pairs

In order to fix the number of samples, it is necessary to select a suitable inter-PP ratio η. Considering that in [16] and [18], there is no clear criteria to set η, it has been decided to use $\eta \geq 0.5$ to keep the packets quite separated without excessively decreasing the number of samples.

As the Low-Factor states in Section 4.2.1, the number of samples and the probing packet size are related for a given time-scale and measurable range. By using Equation 4.7 with $\tau = 40ms$ and $R_{min} = 1Mbps$, Table 4.2 is obtained. The second column shows the range of possible P that can be used for a given number of samples. The third column displays the value of η depending on the chosen packet size. A packet size that is robust to cross-traffic, i.e. larger than 800 Bytes, means very few samples, whereas for a large number of samples, the packet size makes the method too sensitive to cross-traffic. As a trade-off, $P = 500B$ is selected, resulting $N = 4$ and $\eta = 1$.

N	P (Bytes)	η	N	P (Bytes)	η
2	[833,1459]	[2.99,0.50]	6	[314,370]	[0.99,0.50]
3	[589,833]	[1.74,0.50]	7	[271,313]	[0.91,0.50]
4	[456,588]	[1.33,0.50]	8	[239,270]	[0.84,0.50]
5	[371,455]	[1.12,0.50]	9	[214,238]	[0.80,0.50]

Table 4.2: Comparison between the probing packet size and the number of samples for *TOPP* in the low ABw range, where $\eta \geq 0.5$.

The number of iterations, which is essential to control the probing time, depends on R_{max} and the resolution ω for a given R_{min}. As described in Section 3.1.2, it is necessary to increase R_{max} over the maximum desired input rate to mitigate the lack of measurements at the upper bound of the

range. Therefore, R_{max} is set to 9Mbps. By using Equation 3.5, the resolution is 0.32Mbps for 25 iterations.

Self-Loading Periodic Streams

Given a time-scale and a minimum input rate, the number of samples is fixed by Equation 4.8. Considering that $P_{min} = 200B$, the number of samples is set to 25, where $\tau = 40ms$ and $R_{min} = 1Mbps$. By using Equation 3.12, R_{max} is set to 7.5Mbps.

The parameters for trend detection, i.e. the pairwise comparison test, the pairwise difference test and the increasing trend factor, are as set as it is proposed in [20] and hence, $S_{PCT} = 0.55$, $S_{PDT} = 0.4$ and $f = 0.7$. On the other hand, the ABw resolution ω is chosen equal to 0.25Mbps to be similar as in *TOPP*, while the gray-region resolution χ is set to 0.5Mbps as a less restrictive ending condition to reduce the number of iterations due to multiple gray-regions (see Appendix E.2.1).

PathChirp

As the High-Low Factor study shows, a larger probing packet size increases the lower and upper bounds of the measurable ABw range. Taking into account that it is desired to measure a low ABw range, P is set to 1000 Bytes, which is the minimum value recommended in [6] showing better results. By using Equations 4.10 and 4.11, Figure 4.5(a) illustrates that 12 samples is the most suitable number for measuring a range from 1Mbps to 7.5Mbps, where $\gamma = 1.2$ as proposed in [6].

The excursion detection parameters are fixed to the Internet configuration proposed in [6], i.e. $F = 6$ and $L = 3$, since the aim of the following simulations is analyzing the performance of the method under not only CBR, but also other kinds of traffic with burstiness close to Internet traffic behavior. Figure 4.6(a), which simulates a single-hop network with PSD-CBR traffic, shows that Internet parameters work better than the default ones.

4.3.2 High Available Bandwidth Range

Following the methodology stated in the previous section, the most suitable probing packet size and the different parameters of each method are set for the high ABw range, which are summarized in Table 4.4.

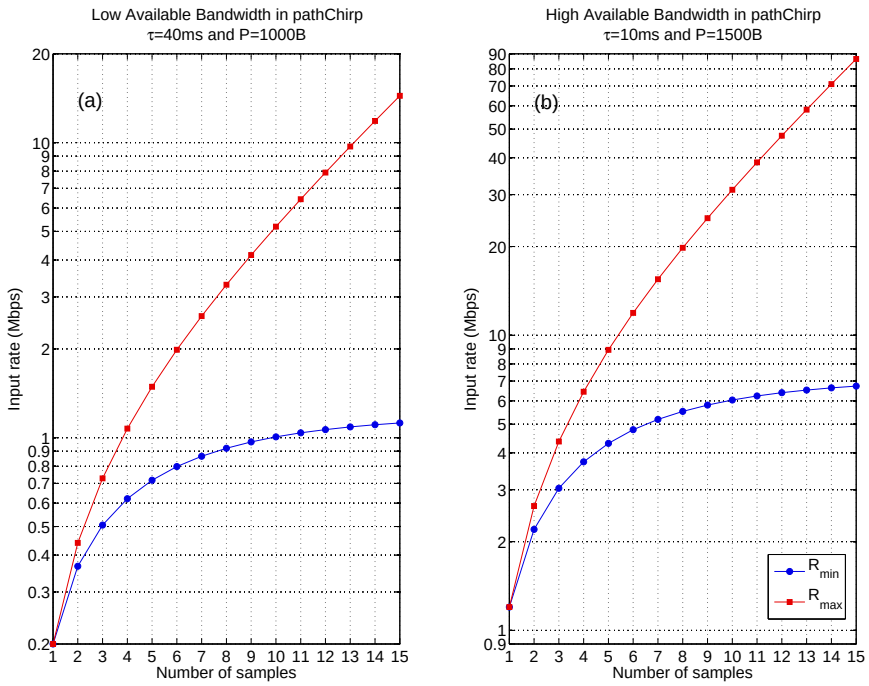

Figure 4.5: Measurable ABw in *pathChirp*. (a) shows the low ABw range [1,7.5]Mbps and (b) shows the high ABw range [8,60]Mbps.

Train of Packet Pairs

For the high ABw range, $R_{min} = 8Mbps$ and $\tau = 10ms$. As in the low ABw case, Table 4.3 shows the different possible packet sizes and their corresponding η for a given number of samples. Following the same criteria, P is set to 700 Bytes, which means 6 samples with $\eta = 0.66$. On the other hand, the maximum input rate is fixed to 70Mbps to mitigate the effect of interval borders described in Section 3.1.2. The number of iterations is set to 31, which implies a resolution of 2Mbps.

Self-Loading Periodic Streams

The limitation of the measurable ABw range explained in Section 3.2.2 leads to 50 samples for $R_{min} = 8Mbps$ and $\tau = 10ms$. With this number of samples, the maximum input rate is limited to 60Mbps. The parameters for trend detection are set as in the previous section and hence, $S_{PCT} = 0.55$, $S_{PDT} = 0.4$ and $f = 0.7$. The ABw resolution is fixed to 1Mbps, whereas the gray-region resolution is set to 2Mbps.

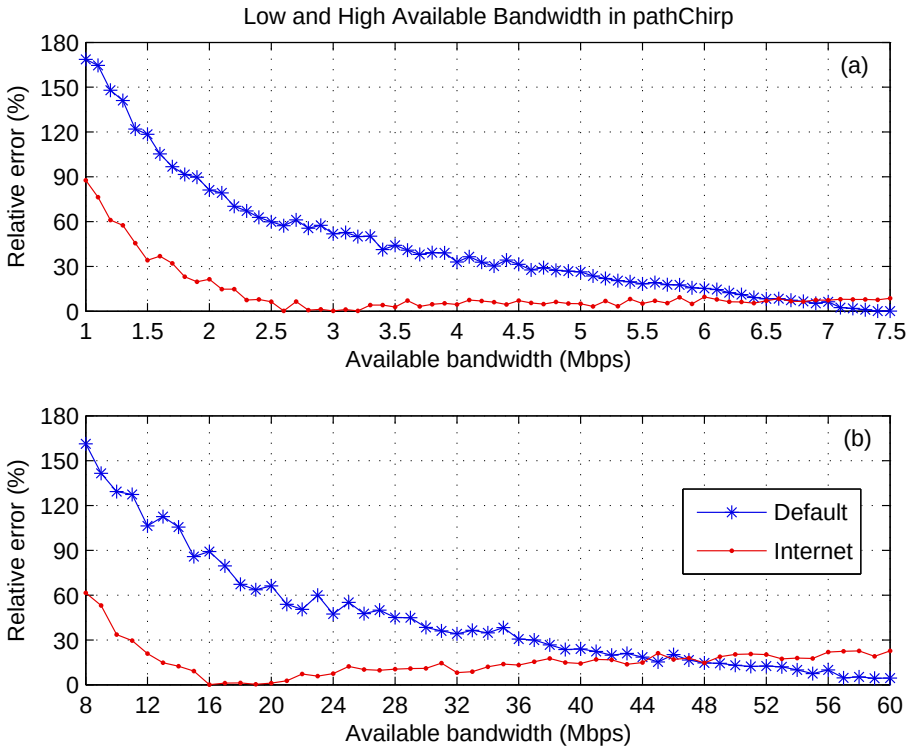

Figure 4.6: Simulation of a single-hop with PSD-CBR, where the Internet configuration of *pathChirp* shows a better performance for both the low (a) and high (b) ABw ranges.

PathChirp

Taking into account that the ABw range is higher than in the previous section, the probing packet size can be increased to make it perform better [6]. Specifically, P is set to 1500 Bytes. Figure 4.5(b) shows that 24 samples fits best the desired ABw range for $\gamma = 1.2$. In the following simulations, the decrease factor and the excursion length are set to 6 and 3 respectively. The values of the Internet configuration [6] are used because they show a better behavior within the whole measured range, as Figure 4.6(b) points out.

4.4 Low Range Simulations Results

This section shows the results of the most representative simulations for the low ABw range. Other results can be found in Appendix F. The parameters of the three different methods are taken from Table 4.4. Measurements of

N	P (Bytes)	η	N	P (Bytes)	η
4	[910,1176]	[1.33,0.50]	12	[324,351]	[0.72,0.50]
5	[742,909]	[1.12,0.50]	13	[300,323]	[0.70,0.50]
6	[626,741]	[1.33,0.50]	14	[279,299]	[0.68,0.50]
7	[542,625]	[0.91,0.50]	15	[261,278]	[0.67,0.50]
8	[477,541]	[0.85,0.50]	16	[245,260]	[0.66,0.50]
9	[427,476]	[0.81,0.50]	17	[231,244]	[0.64,0.50]
10	[386,426]	[0.77,0.50]	18	[218,230]	[0.63,0.50]
11	[352,385]	[0.74,0.50]	19	[207,217]	[0.63,0.50]

Table 4.3: Comparison between the probing packet size and the number of samples for *TOPP* in the high ABw range, where $\eta \geq 0.5$.

the ABw are taken in steps of 0.5Mbps from 1Mbps to 7.5Mbps for different cross-traffic packet sizes $P_x \in \{40, 100, 200, ..., 1500\}$[2] in Bytes. Each estimation is repeated 25 times in order to study the variability of the methods. Considering that *SLoPS* gives an interval of variation of the ABw (see Section 3.2), the center of such interval is used as the estimated value to compare *SLoPS* with the other two methods.

4.4.1 Single-hop

In single-hop simulations ($H = 1.$), the topology is established as stated in Section 4.1.1. The capacity of the tight-link is set to 10Mbps, whereas the latency is fixed to 10ms. The joining links have a latency of 10ms and 100Mbps of capacity, except in PSD-CBR, where the capacity is 1Gbps to avoid the network model constraint explained in Section 4.1.1.

Cross-traffic Packet Size Dependency

Figure 4.7 shows the cross-traffic packet size dependency under CBR traffic. The figure is obtained averaging the relative errors calculated within the whole measurable ABw range for each packet size. From its study, two conclusion are drawn:

- *TOPP* is totally dependent on the packet size. Specifically, it performs well until the cross-traffic packet size is larger than the probing packet

[2]except in PSD-CBR, where a packet size distribution is used.

Low ABw Parameters		TOPP	SLoPS	pathChirp
Time-scale	$\tau(ms)$	40	40	40
Number of samples	N	4	25	12
Number of streams	M	10	10	10
Non-intrusiveness gap	$T_{NI}(ms)$	40	40	40
Number of iterations	I	25	$\star$	1
Probing packet size	$P(Bytes)$	500	[200,1500]	1000
Measurable ABw range	$R(Mbps)$	[1.0,9.0]	[1.0,7.5]	[1.1,7.9]
ABw resolution	$\omega(Mbps)$	0.32	0.25	$\star$
Inter-PP ratio	η	1	-	-
Pairwise comparison test	S_{PCT}	-	0.55	-
Pairwise difference test	S_{PDT}	-	0.4	-
Increasing trend fraction	f	-	0.7	-
Grey-region resolution	$\chi(Mbps)$	-	0.50	-
Spread factor	γ	-	-	1.2
Decrease factor	F	-	-	6
Excursion length threshold	L	-	-	3
High ABw Parameters		**TOPP**	**SLoPS**	**pathChirp**
Time-scale	$\tau(ms)$	10	10	10
Number of samples	N	6	50	14
Number of streams	M	10	10	10
Non-intrusiveness gap	$T_{NI}(ms)$	20	20	20
Number of iterations	I	31	$\star$	1
Probing packet size	$P(Bytes)$	700	[200,1500]	1500
Measurable ABw range	$R(Mbps)$	[8.0,70.0]	[8.0,60.0]	[6.6,71.0]
ABw resolution	$\omega(Mbps)$	2	1	$\star$
Inter-PP ratio	η	0.66	-	-
Pairwise comparison test	S_{PCT}	-	0.55	-
Pairwise difference test	S_{PDT}	-	0.4	-
Increasing trend fraction	f	-	0.7	-
Grey-region resolution	$\chi(Mbps)$	-	2	-
Spread factor	γ	-	-	1.2
Decrease factor	F	-	-	6
Excursion length threshold	L	-	-	3

Table 4.4: Low and high ABw range parameters, where the star ($\star$) means a non-determined value and the hyphen (-) points out a non-required parameter.

size, 500 Bytes in this case.

- The packet size is not a key factor in *SLoPS* and *pathChirp* performance, keeping an average relative error around 10%.

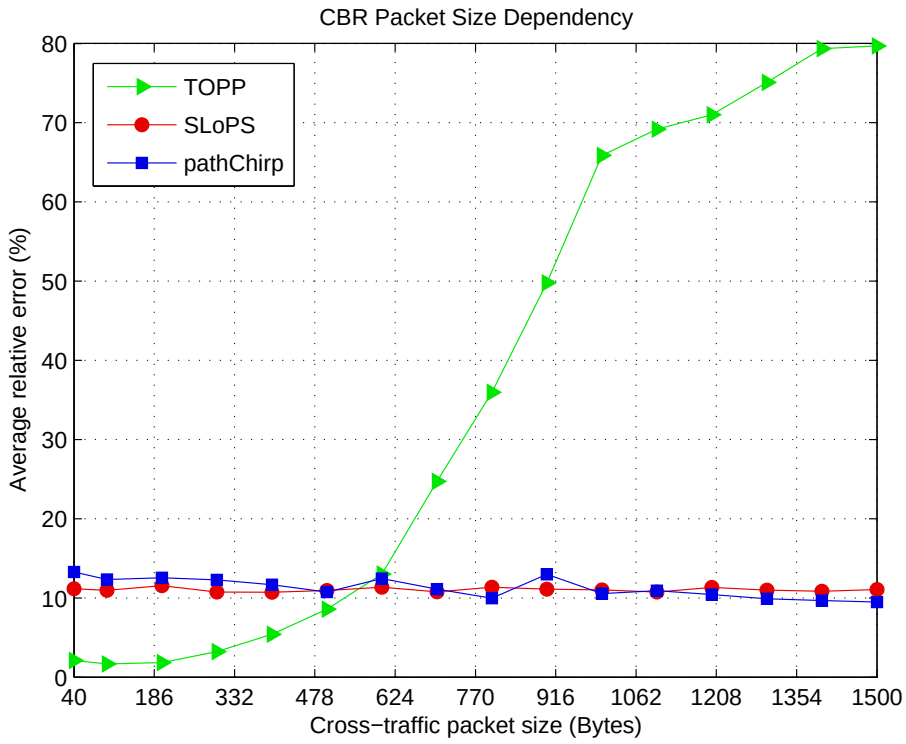

Figure 4.7: Low range simulation in single-hop. Opposite to *SLoPS* and *pathChirp*, *TOPP* shows a great dependency on the packet size. On the other hand, the accuracy of *SLoPS* and *pathChirp* is similar, around 10% of error.

This conclusions are also corroborated by Figure 4.8(a), which shows a PSD-CBR traffic simulation. In such figure, *TOPP* is very sensitive to a packet size distribution, whereas *SLoPS* and *pathChirp* perform similar. Figure F.2 shows a more detailed study for three representative packet sizes, i.e 40, 500 and 1500 Bytes, within the whole range. The *TOPP* dependency is easily explained by the fact that it uses PPs, which are very sensitive to cross-traffic, as it is described in Appendix B.2. This constraint can be mitigated increasing the probing packet size. However, this also means increasing the time-scale and/or the minimum input rate not to reduce the number of samples, as stated by the Low-Factor in Section 4.2.2.

Study of Variability

Figure 4.8(b) shows the variability of the methods under PSD-CBR traffic, obtained by measuring the standard deviation of the 25 repetitions of each estimation. In the figure, *TOPP* experiences a great variability, which is most of the times comparable to the estimation. The variability of *SLoPS* and *pathChirp* is quite smaller in comparison, the latter showing a better behavior at the end of the interval. Due to the random nature of the packet size distribution used in PSD-CBR traffic, *TOPP* interacts with different packet sizes in each estimation. Therefore, the mentioned *TOPP* packet size dependency is also the source of its variability.

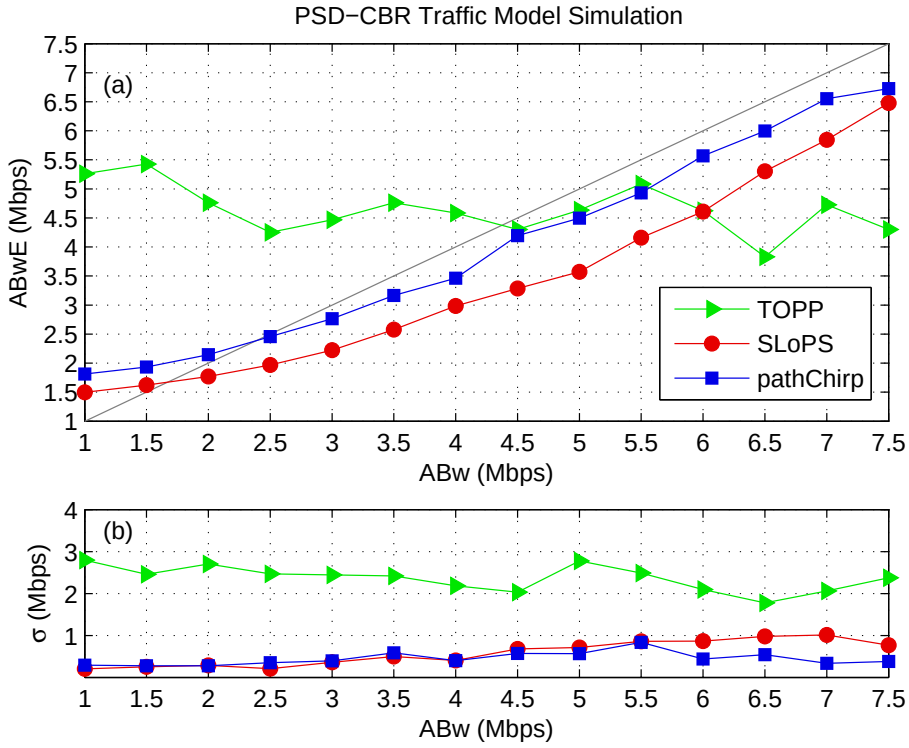

Figure 4.8: PSD-CBR traffic model simulation for low range in single-hop, where (a) shows the average ABwE and (b) the standard deviation obtained from the 25 repetitions of each estimation.

Cross-traffic Pattern Dependency

Figures 4.8(a) and 4.9 compare *TOPP*, *SLoPS* and *pathChirp* under different kinds of cross-traffic. In Poisson and CBR traffic models, it has been chosen

$P_x = 500B$ since it is a representative average cross-traffic size [28]. From the figures, it can be concluded that:

- *TOPP* is very accurate for CBR traffic if $P_x \leq P$, as explained previously. However, *TOPP* does not work for PSD-CBR and Poisson models, which points out that it is sensitive, not only to cross-traffic packet size distributions, but also to random inter-arrivals.
- *PathChirp* performs quite better than *SLoPS* regardless of the kind of cross-traffic. This behavior could be due to the fact that *pathChirp* makes use of the excursions analysis, which takes into account the burstiness.
- *SLoPS* and *pathChirp* tend to underestimate the ABw except for values very close to the lowest bound of the measurable range.

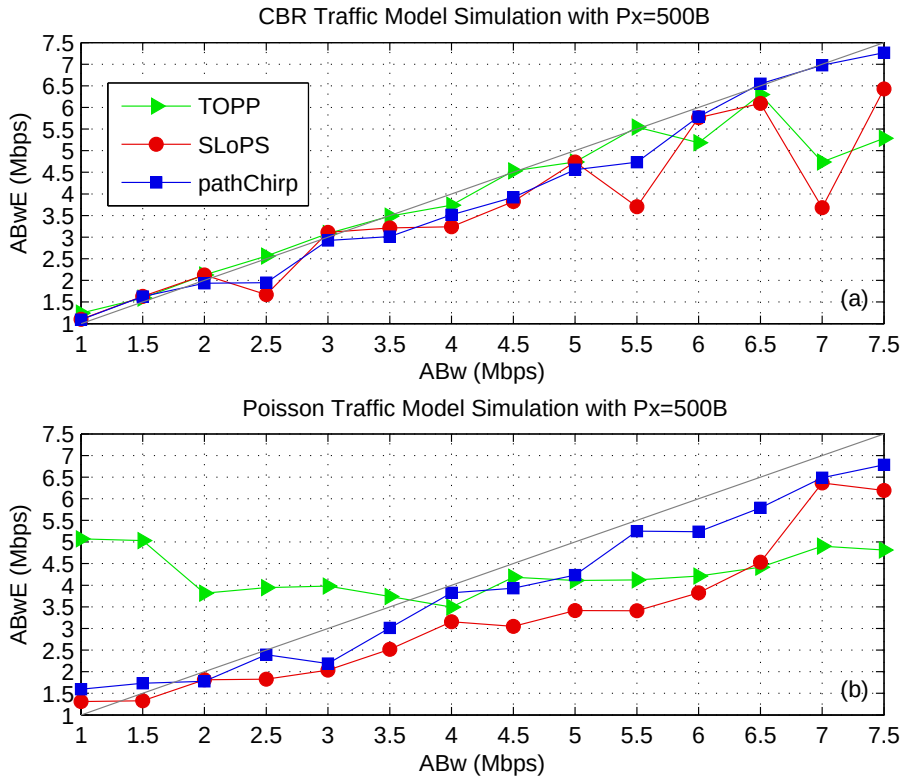

Figure 4.9: The average ABwE is compared for CBR (a) and Poisson (b) traffic models with $P_x = 500B$ for low range in single-hop.

Figure F.3 shows a comparison of the three methods under different cross-traffic models, which reinforces the conclusions drawn in this section.

4.4.2 Multi-hop

After having analyzed different aspects of the performance of the three techniques in a single-hop path, this section focuses on comparing those results with the ones obtained in a multi-hop path. *TOPP* is not considered here because of the poor performance shown in the single-hop case, but the reader can refer to a sample result in Figure F.4, showing *TOPP* performance in a multi-hop setting.

The multi-hop topology used for the following simulations consists of 5 hops with a single tight-link located in the middle hop of the path (see Section 4.1.1). The capacity of all the hops is set to 10Mbps with 10ms of latency. The ABw of the non-tight-links is fixed to 9Mbps, avoiding secondary tight-links. The joining links are set as in Section 4.4.1.

Study of Variability

Figure 4.10(b) shows the variance obtained from single and multi-hop simulation of *SLoPS* and *pathChirp*. The plot points out that the variability of each method has a similar trend in both single and multi-hop.

Cross-traffic Pattern Dependency

Figures 4.10(a) and 4.11 represent a comparison between single and multi-hop performance of *pathChirp* and *SLoPS* under different cross-traffic models. From its analysis, it is drawn that:

- *PathChirp* works worse in multi-hop than in single-hop, since it is more likely that the exponential structure of the chirp is modified by the other hops of the path. As *pathChirp* probes the network with a certain rate once per stream, the variation of such structure is decisive.
- *SLoPS* is less sensitive to multi-hops paths as it repeatedly measures the same rate during the stream, which makes it more robust against variations of the stream structure.
- For the more realistic cross-traffic models, i.e. PSD-CBR and Poisson, *pathChirp* performs as well as *SLoPS*, or even better depending on the interval stretch.

4.5 High Range Simulations Results

This section is analogous to Section 4.4, but for the high ABw range. Appendix F includes more results that complete this study. The followed

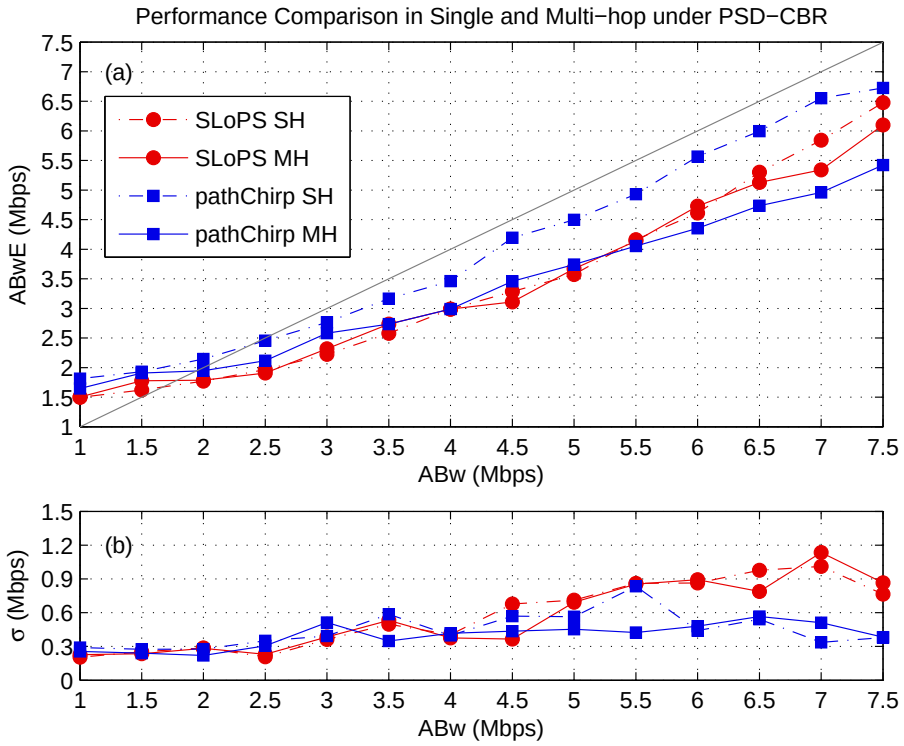

Figure 4.10: Single and multi-hop comparison for low range under PSD-CBR, where (a) shows the average ABwE and (b) the standard deviation obtained from the 25 repetitions of each estimation.

methodology is the same as the previous section, except the fact that the ABw measurements are taken in steps of 4Mbps from 8Mbps to 60Mbps.

4.5.1 Single-hop

In single-hop simulations ($H = 1$), the topology is established as stated in Section 4.1.1. The capacity of the tight-link is set to 100Mbps, whereas the latency is fixed to 10ms. The joining links have a latency of 10ms and 1Gbps of capacity, except in PSD-CBR, where the capacity is 10Gbps to avoid the network model constraint explained in Section 4.1.1.

Cross-traffic Packet Size Dependency

Figure 4.12(a) illustrates a performance comparison under PSD-CBR. From the figure, it is drawn that the three methods show the same behavior as in Section 4.4. It is to say that *TOPP* is strongly dependent on the packet size,

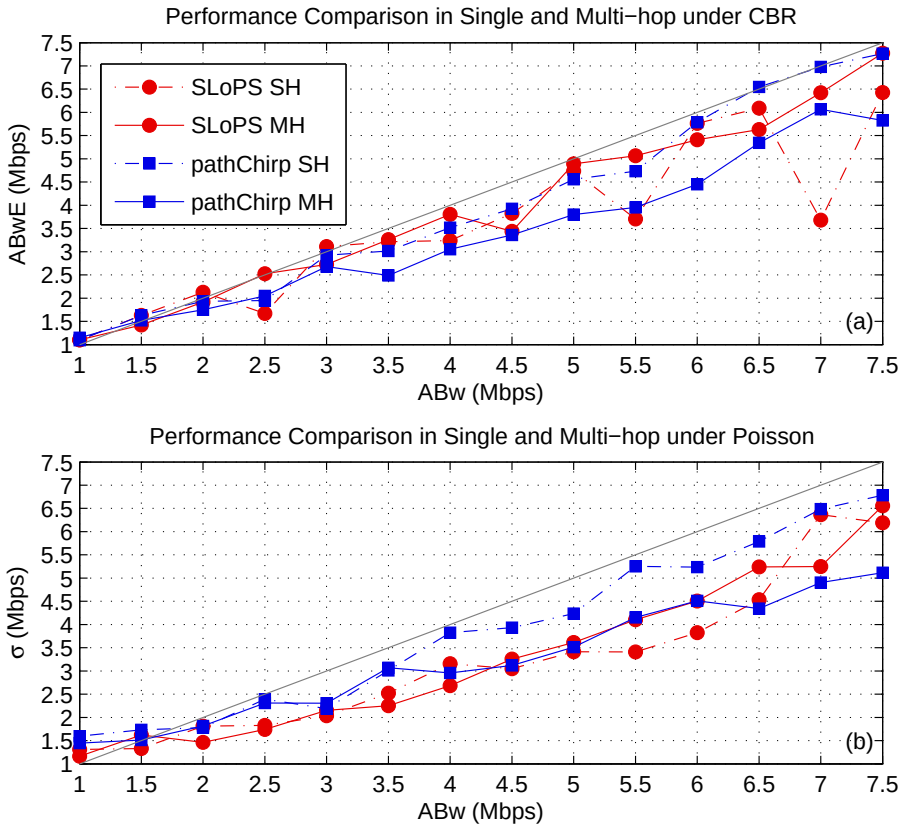

Figure 4.11: Single and multi-hop comparison for low range under CBR (a) and Poisson (b) traffic with $P_x = 500B$.

whereas *pathChirp* and *SLoPS* are much less sensitive to such variation, fact that is also corroborated by Figure F.5(a).

Study of Variability

Figure 4.12(b) shows the variability of the methods under PSD-CBR traffic. As in Section 4.4, *TOPP* is much more variable than *SLoPS* and *pathChirp* due to its cross-traffic packet size dependency. *PathChirp* also shows a better behavior at the end of the interval.

Cross-traffic Pattern Dependency

The accuracy in the high ABw range is alike to Section 4.4. From Figure 4.8(b) and 4.12(b), it is possible to stress that *pathChirp* works slightly worse

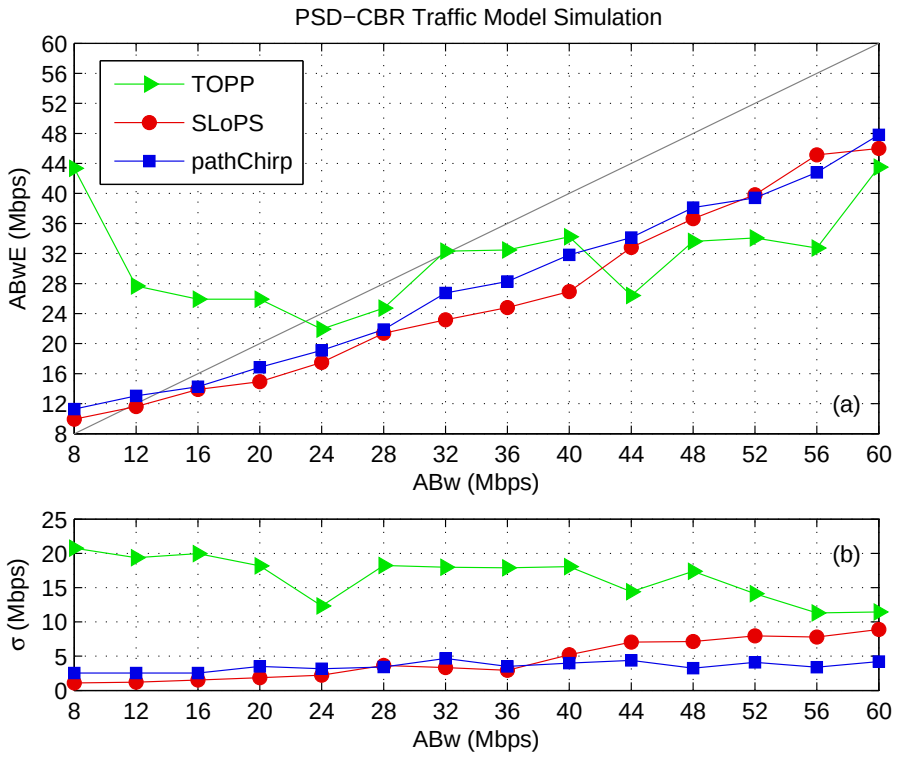

Figure 4.12: PSD-CBR traffic model simulation for the high ABw range in single-hop, where (a) shows the average ABwE and (b) the standard deviation obtained from the 25 repetitions of each estimation.

in the high ABw range, being its accuracy comparable to the results of *SLoPS*. This could be due to the fact that a similar number of samples are used to measure a wider ABw range, leading to a worse resolution. Similar results are obtained for CBR and Poisson traffic, shown in Figures F.5(b) and F.5(c).

4.5.2 Multi-hop

This section provides comparative results between single and multi-hop paths. *TOPP* is not considered here because of its inaccurate results obtained in single-hop. The multi-hop topology used for the following simulations is fixed as in Section 4.4.2. The capacity of all the hops is set to 100Mbps with 10ms of latency. The ABw of the non-tight-links is fixed to 90Mbps, avoiding secondary tight-links. The joining links are set as in Section 4.5.1.

Study of Variability

The variability trend experienced by *pathChirp* and *SLoPS* does not change from single to multi-hop, as Figure 4.13 points out. Such behavior is similar to the one obtained in Section 4.4.

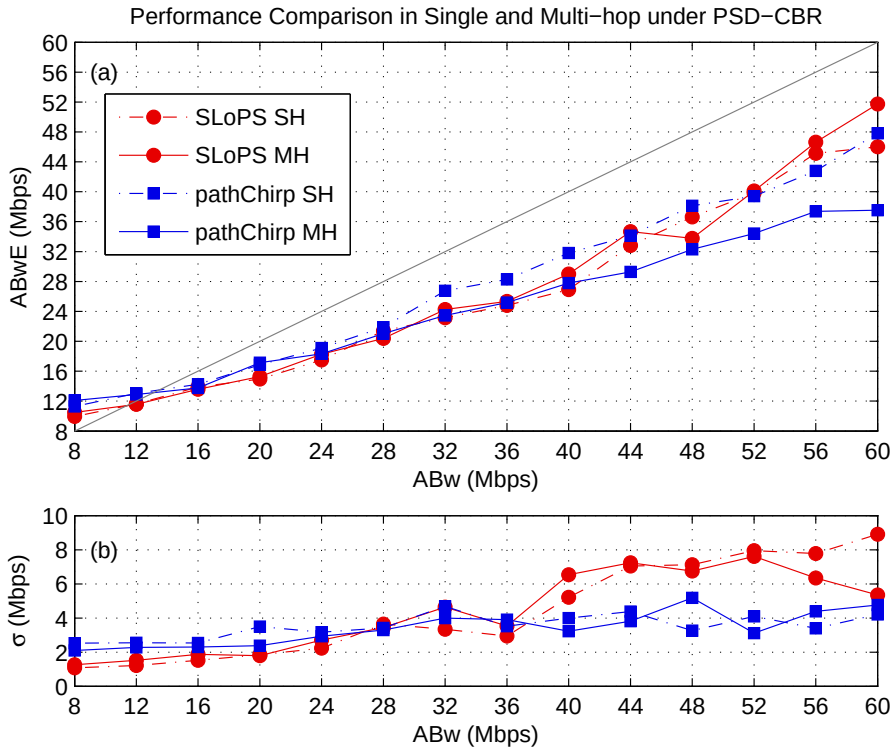

Figure 4.13: Single and multi-hop comparison for high range under PSD-CBR, where (a) shows the average ABwE and (b) the standard deviation obtained from the 25 repetitions of each estimation.

Cross-traffic Pattern Dependency

PathChirp tends to perform worse in multi-hop than in single-hop paths, as described in Figures 4.13, F.6(a) and F.6(b). However, *SLoPS* is less sensitive to a variation in the number of hops. These results are also obtained in the low ABw range study.

4.6 Study of Efficiency

Table 4.5 summarizes the efficiency parameters described in Section 3.4 and obtained using Table 4.4 for the low and high ABw range respectively. From its study, it is drawn that *pathChirp* is the most efficient method since its load is very reduced, as well as its probing time. On the other hand, *TOPP* is the less intrusive method due to its long probing time. Note that the non-intrusiveness gap T_{NI} used in these simulations is not optimized. So, it is likely that a deep study of such parameter would allow the probing time to be considerably reduced. Even so, the measurements of Table 4.5 continue being useful for a comparison study.

Low ABw Parameters		**TOPP**	**SLoPS**	**pathChirp**
Load	$L_p(KB)$	977	1079	127
Probing time	$T_p(s)$	19.96	3.96	0.76
Average rate	$R_p(Mbps)$	0.40	2.23	1.37
High ABw Parameters		**TOPP**	**SLoPS**	**pathChirp**
Load	$L_p(KB)$	2543	2540	220
Probing time	$T_p(s)$	9.28	1.78	0.28
Average rate	$R_p(Mbps)$	2.24	11.69	6.44

Table 4.5: Efficiency parameters for the low and the high ABw ranges, where *pathChirp* is shown as the most efficient method and *TOPP* as the least intrusive one.

4.7 Summary

The aim of this chapter is to compare *TOPP*, *SLoPS* and *pathChirp*, in terms of accuracy, variability and efficiency, but taking into account the statistical properties of the ABw. For this reason, an initial viability study of such statistical comparison is conducted resulting that it is impossible to evaluate the three methods making use of the same number of samples and observable time-scale. Due to the fact that the time-scale is a critical parameter for the probe duration, it is chosen as the common parameter.

Different simulations are carried out in single and multi-hop paths with three different kinds of cross-traffic, i.e. CBR, PSD-CBR and Poisson. From

the results, *pathChirp* is proposed as the tool to be implemented in a packet-switched mobile transport network since it excels in terms of both accuracy and efficiency, despite its slightly worse performance in multi-hop paths. When it comes to the other methods, *TOPP* is strongly dependent not only on the cross-traffic packet size distribution, but also on the cross-traffic random inter-arrivals. In addition, it experiences a great variability and is very slow. *SLoPS* shows a better accuracy than *TOPP*, but worse than *pathChirp* in most of the simulations. It also needs more time to give an estimation. As an advantage, it is less sensitive to multi-hop paths.

Chapter 5

PathChirp Optimization

In Section 4.7, *pathChirp* is stated as the most suitable method, as a trade-off between accuracy and efficiency, to be applied in a packet-switched mobile transport network. As explained in Chapter 1, the suggested tool should be optimized to be applied for admission control and load control in E-UTRAN. Admission control requires an ABw between 128Kbps and 2Mbps in order to guarantee the QoS to a new user [29], whereas load control needs a higher interval, for instance up to 60Mbps, to warn about possible transport network congestion.

Assuming the capacity of an E-UTRAN cell is fixed by the STM-1 backhaul transmission line of 155Mbps [30], very low ABw rates mean very high levels of utilization. In this situation, bursty cross-traffic may overwhelm the network. Figure 5.1 shows the percentage of time the cross-traffic rate is equal to the capacity against the level of utilization under different time-scales. The network is more likely to be overloaded as the utilization increases, whereas longer time-scales reduces the observed network overload.

To be useful for admission control purposes, *pathChirp* should be able to estimate an ABw of 128Kbps, which means an utilization level of 98.7% on a STM-1 link. As Figure 5.1 points out, this fact implies a great percentage of overload regardless of the time-scale, making the method unsuitable for this task. Therefore, this chapter focuses only on load control. As a trade-off between the probing time and the number of samples, which is related to the probing load, a time-scale $\tau = 7.5ms$ is used. For such value, the maximum level of utilization to avoid a high overload is 90%, as Figure 5.1 illustrates. The minimum maximum input rates are set to 15Mbps and 60Mbps respectively.

The chapter includes an analysis of profiles other than the exponential structure described in [6] and an optimization of the different *pathChirp* pa-

rameters. It continues describing two iterative approaches, which take advantage of the previous estimation, and a statistical treatment based on least squares fitting. Finally, the possibility of using RTT measurements is studied.

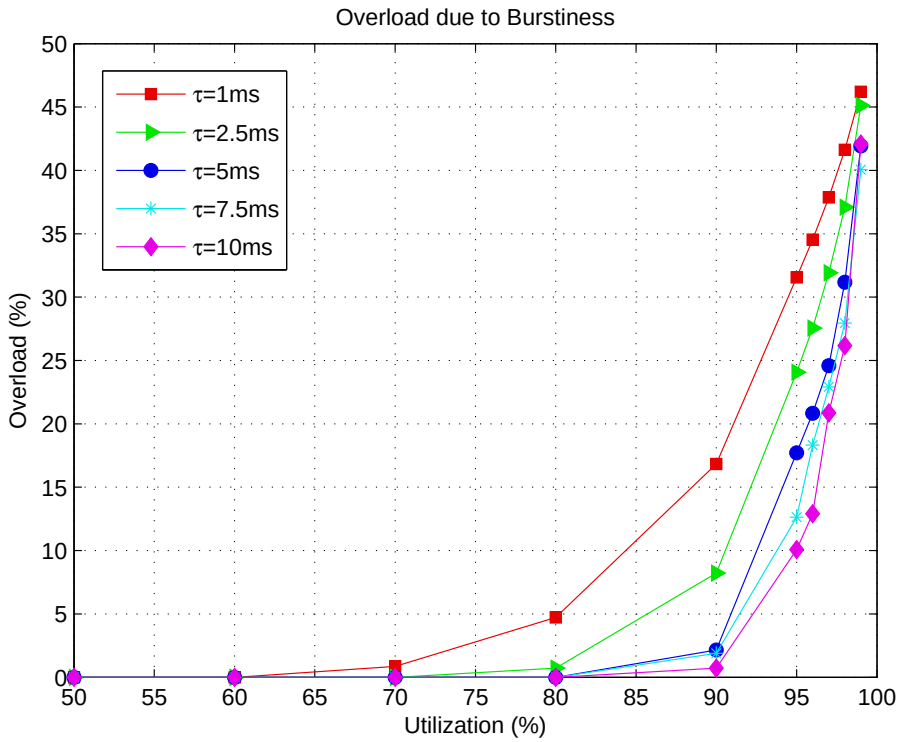

Figure 5.1: As the average utilization increases, the network is more likely to be overloaded due to cross-traffic burstiness, whereas longer time-scales reduces the observed network overload.

5.1 Chirp Profiles

As explained in Section 2.3.2, the chirp profile determines the different rates with which the network is probed and hence, the resolution between two consecutive rates. The exponential structure proposed in [6] has more resolution in the higher stretch of the interval, which could affect the estimations (see Section 3.3.2). This section generalizes the exponential profile and introduces other chirp profiles that show different resolution characteristics. The derivative of each profile is used as a measure of the resolution since the stretch in which the derivative is closer to zero, the higher the resolution.

5.1.1 Description of Profiles

The profiles are defined so as to follow the same guidelines set by the exponential profile described in Section 2.3.2, which are:

- The instantaneous rate R_k decreases as k increases (see Figure 2.11), being expressed as
$$R_k = \frac{P}{T_k} \tag{5.1}$$
- The maximum and minimum input rate are defined for $k = 0$ and $k = K - 2$ respectively as follows
$$R_{max} = R_0 = \frac{P}{\delta} \qquad R_{min} = R_{K-2} \tag{5.2}$$
- The time-scale is calculated using the next equation
$$\tau = \sum_{k=0}^{K-2} T_k \tag{5.3}$$

Exponential Profile

As Figure 5.2(a) illustrates, the *exponential profile* is determined by the following exponential equation
$$f(x) = c(a + b^{-x}) \tag{5.4}$$
and its derivative can be expressed as
$$\frac{d}{dx} f(x) = -\frac{c}{\ln b} b^{-x} \tag{5.5}$$
which shows, as Figure 5.2(c), that it provides more resolution in the higher stretch of the interval. Making the different rates of the chirp satisfy Equation 5.4 to measure a certain ABw range, the instantaneous rate is
$$R_k = \frac{P}{T_k} = \frac{P}{\delta} \frac{r_A + \gamma^{-k}}{r_A + 1} \qquad k = 0, \ldots, K - 2 \tag{5.6}$$
where r_A is calculated as
$$r_A = \frac{R_A}{R_{max} - R_A} \tag{5.7}$$
where $R_A < R_{min}$ is the *asymptotic rate* selected to stress the exponential structure of the profile. The larger the R_A, the more the exponential structure is emphasized. In case of $R_A = 0$, the proposal in [6] is obtained. Substituting Equation 5.2 in 5.6, the next expression for the spread factor can be drawn
$$\gamma = \left(\frac{R_{min}}{R_{max}} (r_A + 1) - r_A \right) \frac{1}{K - 2} \tag{5.8}$$

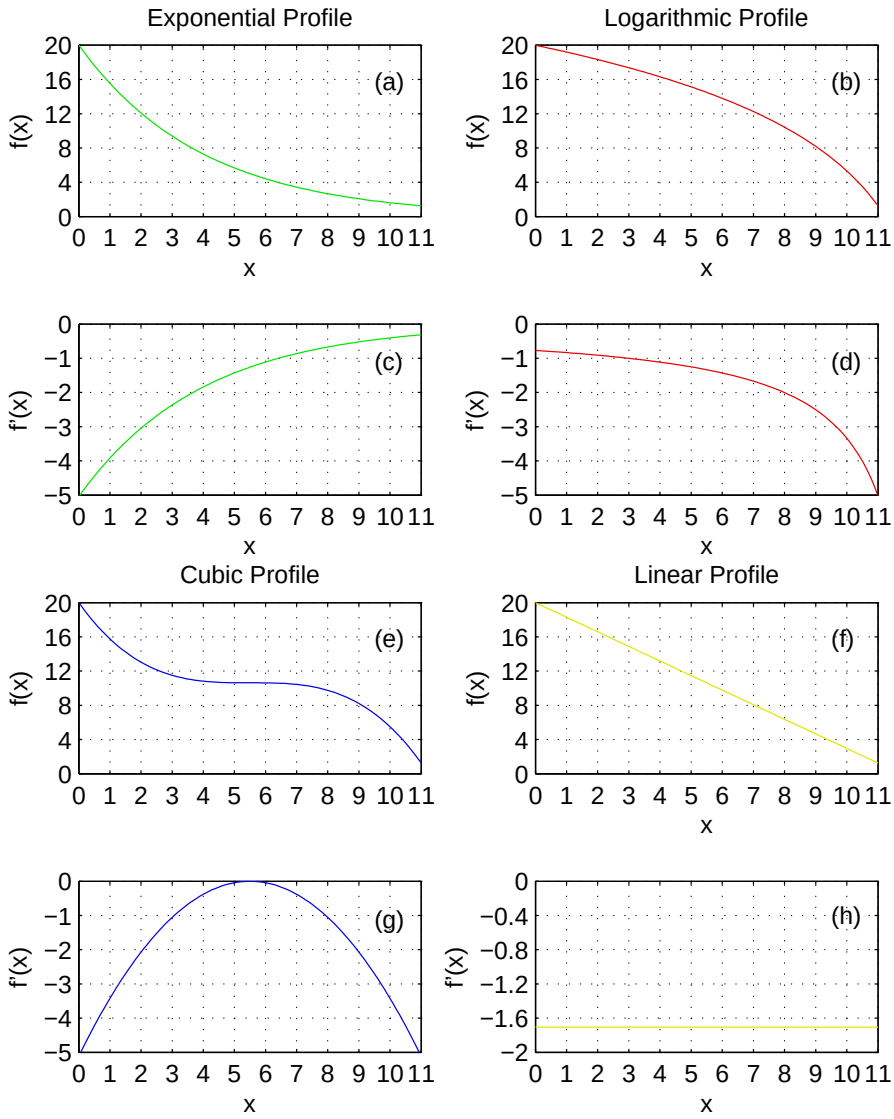

Figure 5.2: Chirp profiles and their derivatives, which show the different resolution characteristics. Note that $f'(x) = \frac{d}{dx} f(x)$.

Logarithmic Profile

The *logarithmic profile* (see Figure 5.2(b)) is fixed by a logarithmic equation as follows

$$f(x) = c \cdot ln[b(a-x)] \qquad \text{if } x \leq a-1 \tag{5.9}$$

and its derivative is

$$\frac{d}{dx} f(x) = \frac{c}{x-a} \tag{5.10}$$

which shows more resolution in the lower stretch of the interval, opposite to the exponential structure, as Figure 5.2(d) illustrates. The instantaneous rate can be calculated as

$$R_k = \frac{P}{T_k} = \frac{P}{\delta} \frac{\ln[\gamma(K-k)]}{\ln(\gamma K)} \qquad k = 0, \ldots, K-2 \tag{5.11}$$

from which, by using Equation 5.2, the spread factor is obtained as

$$\gamma = \frac{2^{r_1}}{K^{r_2}} \tag{5.12}$$

where r_1 and r_2 are defined as

$$r_1 = \frac{R_{max}}{R_{max} - R_{min}} \qquad r_2 = \frac{R_{min}}{R_{max} - R_{min}} \tag{5.13}$$

Cubic Profile

Figure 5.2(e) exemplifies the *cubic profile*, fixed by a cubic equation

$$f(x) = c\left(1 - b(x-a)^3\right) \tag{5.14}$$

and its derivative is

$$\frac{d}{dx} f(x) = -3bc(x-c)^2 \tag{5.15}$$

where the highest resolution is located in the middle of the interval as shown in Figure 5.2(g). An important parameter of the cubic function is the inflection point Θ, in which the slope changes from increasing to decreasing. Such point is obtained as

$$\frac{d^2}{dx^2} f(x) = 0 \Rightarrow \Theta = [a, c] \tag{5.16}$$

For a given ABw range, where the inflection point is located in the middle of the interval, i.e. $a = \frac{K-2}{2}$, the instantaneous rate is

$$R_k = \frac{P}{T_k} = \frac{P}{\delta} \frac{1 - \gamma\left(k - \frac{K-2}{2}\right)^3}{1 + \gamma\left(\frac{K-2}{2}\right)^3} \qquad k = 0, \ldots, K-2 \tag{5.17}$$

from which, the spread factor is obtained, using Equation 5.2, as

$$\gamma = \frac{R_{max} - R_{min}}{R_{max} + R_{min}} \left(\frac{2}{K-2} \right)^3 \tag{5.18}$$

Linear Profile

As Figure 5.2(f) illustrates, the *linear profile* is determined by the following linear equation

$$f(x) = a - bx \tag{5.19}$$

and its derivative can be expressed as

$$\frac{d}{dx} f(x) = -b \tag{5.20}$$

which shows that the resolution is constant for the whole interval (see Figure 5.2(h)). Applying the linear profile to measure a certain ABw range, the instantaneous rate is

$$R_k = \frac{P}{T_k} = \frac{P}{\delta}(1 - \gamma k) \qquad k = 0, \ldots, K-2 \tag{5.21}$$

from which, by using Equation 5.2, the spread factor can be drawn

$$\gamma = \left(1 - \frac{R_{min}}{R_{max}} \right) \frac{1}{K-2} \tag{5.22}$$

5.1.2 Adjustment of Profiles

In order to adjust the previous profiles to measure a similar ABw range during a given time-scale, it is necessary choose the proper spread factor γ and a suitable number of packets per stream K of each profile that meets such requirements. To do so, the next procedure is followed:

1. A time-scale τ, a number of samples N and initial ABw range $[R_{min}, R_{max}]$ are selected.

2. With N and $[R_{min}, R_{max}]$ chosen, and by using the proper equation depending on the profile, the spread factor γ is obtained.

3. Taking into account that $\tau = \delta \sum_{k=0}^{K-2} f(k, \gamma)$, the gap between the two closest packets of the stream δ is calculated as

$$\delta = \frac{\tau}{\sum_{k=0}^{K-2} f(k, \gamma)} \tag{5.23}$$

4. The probing packet size P is obtained as

$$P = round\{\delta R_{max}\} \tag{5.24}$$

where $round\{X\}$ rounds X to the nearest integer, which slightly modifies the initial ABw range.

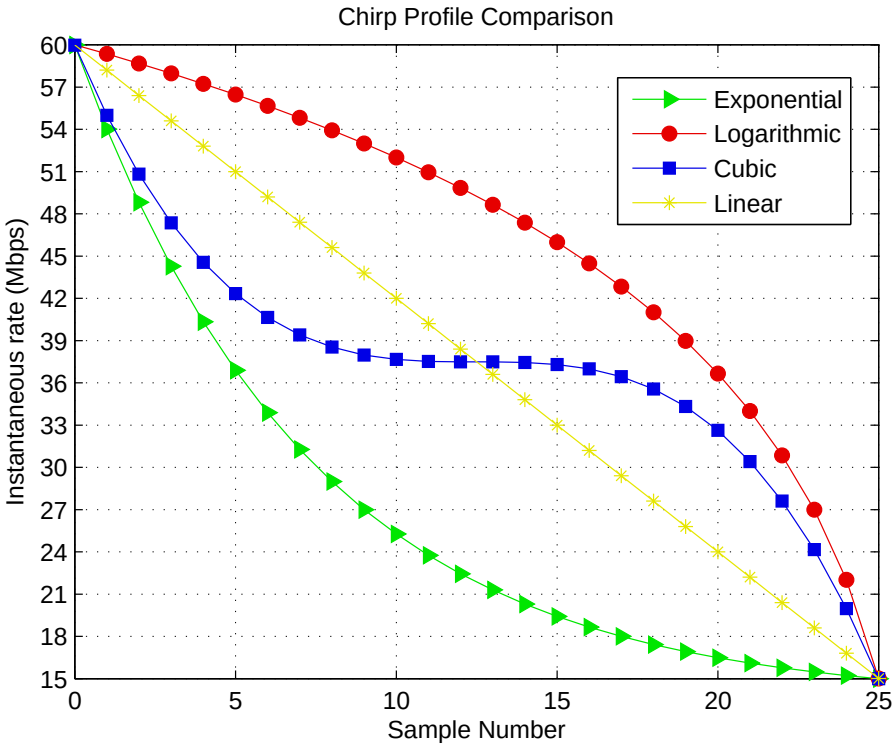

Figure 5.3: Different *pathChirp* profiles to measure the same ABw range with the same number of samples.

Although [6] suggests using $P > 1000B$, this paper also states that choosing $P > 800B$ leads to good results. Taking into account that a range between 15Mbps and 60Mbps is measured using a time-scale of 7.5ms, all the possible number of samples and their respective packet sizes for each profile are summarized in Table 5.1. Since it is desired to maximize N to emphasize the effect of the different profile structures, but keeping the same N for all methods not to change the statistical conditions, 26 is chosen as the number of samples. Note that asymptotic rate R_A is equal to 13.5Mbps. Figure 5.3 shows a graphical representation of such profiles.

Profile	N	P (Bytes)	P_{26} (Bytes)	$L_{p,26}$ (KB)
Exponential	[15-26]	[1421-818]	818	216
Logarithmic	[26-51]	[1481-808]	1481	390
Cubic	[22-40]	[1463-814]	1243	328
Lineal	[20-37]	[1493-814]	1154	304

Table 5.1: A larger packet size means a lower number of samples. The last two columns show the packet size (P_{26}) and the probing load ($L_{p,26}$) obtained for $N = 26$.

5.1.3 Study of Profiles

In order to compare the different profiles, several simulations in a multi-hop path ($H = 5$) are carried out. Estimations of the ABw are taken in steps of 3Mbps from 15Mbps to 60Mbps. Moreover, the capacity of all the hops is set to 155Mbps with 10ms of latency. As in Chapter 4, the utilization of the non-tight-links is fixed to 10%, i.e. 135Mbps of ABw, avoiding secondary tight-links. The joining links have a latency of 10ms and 10Gbps of capacity. Each estimation is repeated 25 times. Note that M is set to 10 and T_{NI} is fixed to 7.5ms.

Figure 5.4 shows a comparison of the different profiles under PSD-CBR traffic, from which it can be seen that the distribution of the samples is highly related to the performance of the method. It is to say that a stretch with a higher density of samples shows a better accuracy in such stretch. Specifically, the exponential profile performs the best in the lower part of the interval, the logarithmic profile excels for higher rates and the cubic profile is better around the middle range, whereas the linear profile shows good accuracy within the whole interval. Such conclusions are reinforced by the results obtained for Poisson traffic model, which can be found in Figures F.6(c) and F.6(d).

5.2 Parameters Optimization

The *pathChirp* proposal in [6] does not give a value for the number of streams to be sent in a fleet (directly related to the probing time of the method), and optimum values for the excursion detection parameters, as it is introduced in Section 3.3.2, are not specified either. This study is divided in the optimization of the parameters related to the probing time and the ones related to the excursion detection. As explained in Section 5.1.3, the linear profile shows

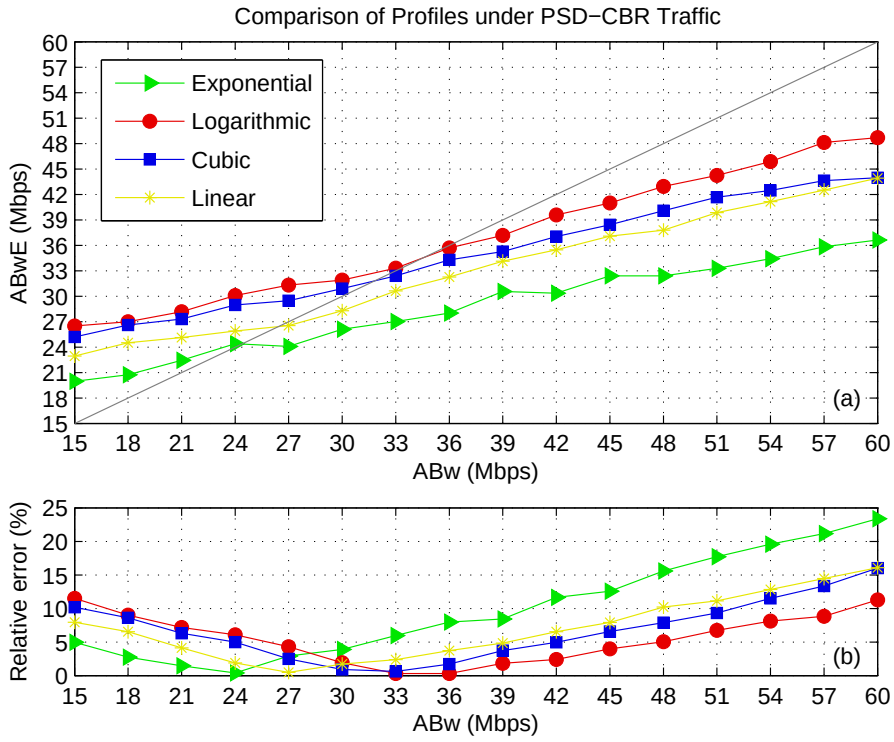

Figure 5.4: *PathChirp* profiles comparison under PSD-CBR traffic.

good accuracy regardless of the location of the ABw in the measurable ABw range, reason why such profile is used for the optimization of the method.

5.2.1 Probing Time Parameters

As Section 3.4.2 describes, the probe duration is completely dependent on the number of streams M that are sent in a fleet and on the non-intrusiveness gap T_{NI} left in between two consecutive streams. In theory, increasing M would make the estimations tend to stabilize, but it would also increase the probe duration. Increasing T_{NI} reduces the chances of a stream interfering with the next one, but it leads to a longer probing time. This section studies the effect of both parameters for the sake of a better performance than obtained in Chapter 4.

Number of Streams per Fleet

The effect of the number of streams to the *pathChirp* performance is studied using a single-hop simulation ($H = 1$) where M varies from 1 to 100 for different ABw rates, i.e. $A = \{15, 30, 45, 60\} Mbps$. The tight-link and the joining links have 155Mbps and 10Gbps of capacity respectively, while all the latencies are fixed to 10ms. Each estimation is repeated 25 times in order to study the variability. Note that T_{NI} is 7.5ms. Figure 5.5 represents the average (a) and the standard deviation (b) of the ABwE, as a function of M, for a simulation under PSD-CBR traffic. The figure shows that the average tends to stabilize and the standard deviation decreases as M increases. Specifically, for $M \geq 50$, the average does not change and the variance is lower than 1.5Mbps.

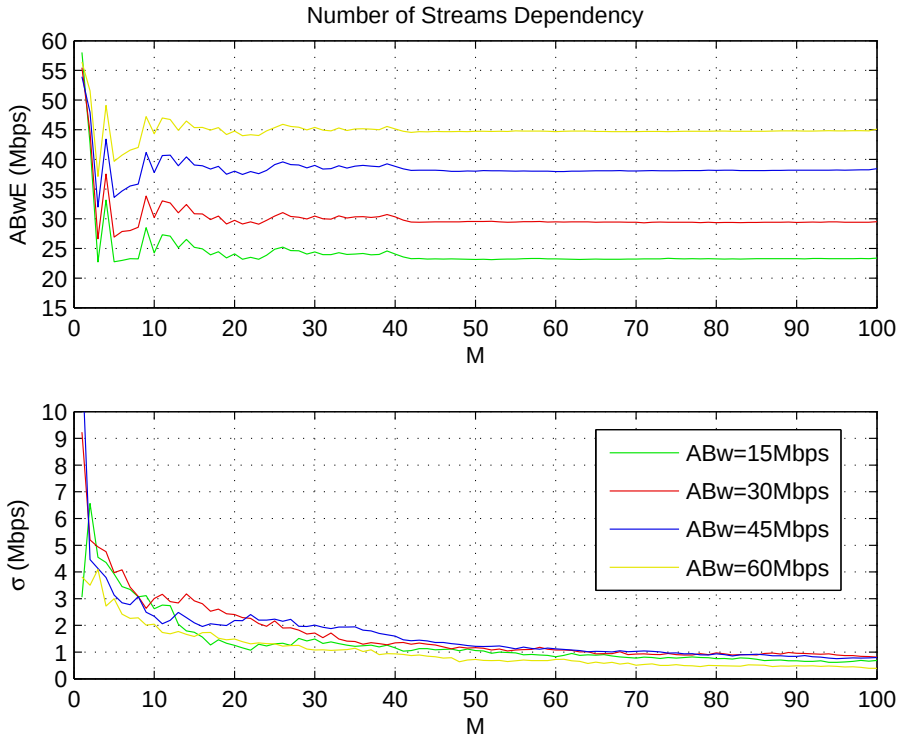

Figure 5.5: As the number of streams per fleet increases, the standard deviation reduces and the ABwE tends to stabilize.

Non-intrusiveness Gap

The effect of the non-intrusiveness gap to the *pathChirp* performance is studied using a single-hop simulation ($H = 1$) where T_{NI} varies in steps of 0.75ms from 0.75ms to 7.5ms and in steps of 7.5ms from 7.5ms to 75ms. Note that the rest of the configuration is set as in the previous section, where M is set to 10. Figure 5.6 shows that T_{NI} does not have a noticeable effect on the performance of the method. Therefore, the initial criteria of $T_{NI} \geq \tau$ can be used in order to reduce the average load whenever there are no time limitations.

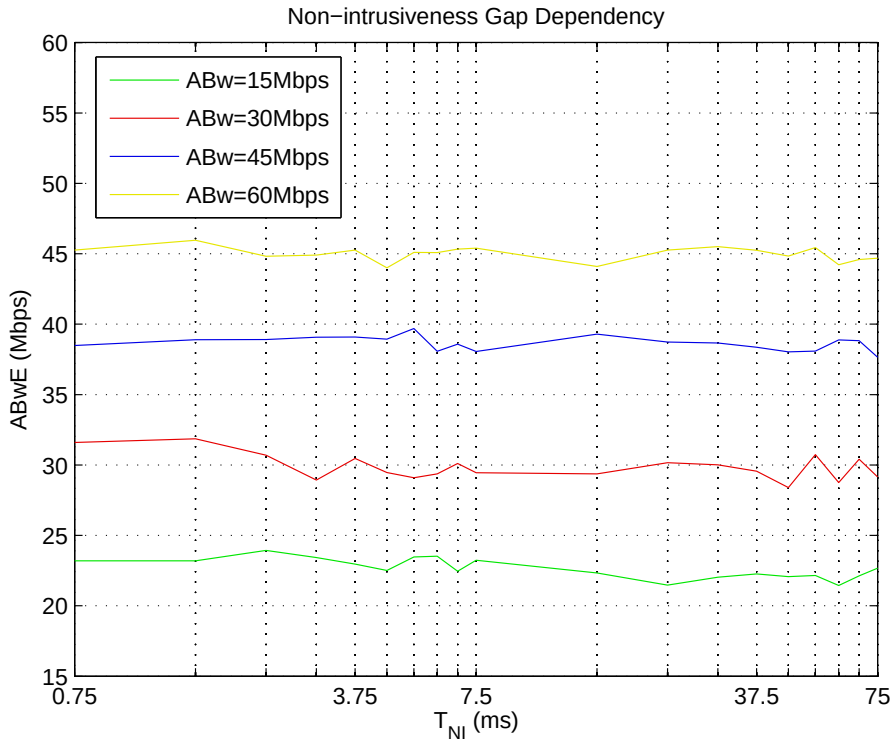

Figure 5.6: The non-intrusiveness gap does not have a noticeable effect to the performance of *pathChirp*.

5.2.2 Excursion Detection Parameters

In Chapter 4, the *pathChirp* excursion detection parameters, i.e. excursion length threshold L and decrease factor F, are set to the proposed Internet configuration values [6]. The reason is that *pathChirp* shows better accuracy with these values than with the default ones as it can be seen in Figure 4.6.

This section studies the performance of the method for a wide range of L and F with the purpose of obtaining the optimum values.

A simulation is performed under PSD-CBR traffic on a multi-hop path with the same configuration as in Section 5.1.3. In the simulation, L varies from 3 to 25 and F changes in steps of 0.5 from 1.5 to 10 to obtain the average relative error of each estimation. *PathChirp* usually underestimates, so the excursion parameters are chosen to compensate such underestimation. As Figure 3.7 shows, rising L quickly leads to overestimation, which can be tuned by changing F, since it produces a slower variation.

Figure 5.7 summarizes the effect of the excursion parameters. The relative error does not decrease for $L \geq 14$, due to the fact that all the excursions have already been discarded, as explained in Section 3.3.2. Specifically, the minimum relative error takes place at $F = 7.5$.

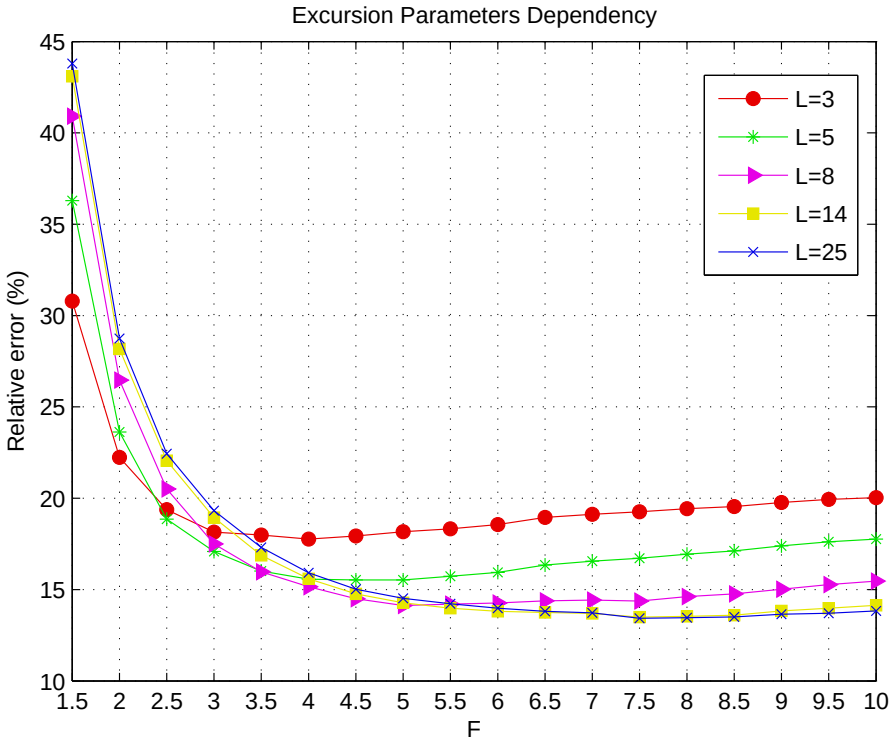

Figure 5.7: The relative error does not decrease for $L \geq 14$. Specifically, the minimum relative error takes place at $F = 7.5$.

5.2.3 Simulation Results for the Optimized Profiles

Figure 5.8 shows a comparison under PSD-CBR traffic between the different profiles before and after using the optimum parameters obtained along this section, i.e. $M = 50$, $F = 7.5$, $L = 14$ and $T_{NI} = \tau$. The multi-hop scenario is the same as in Section 5.1.3. From the figure, it can be drawn that the relative error considerably decreases for the optimum parameters (solid line) compared to the initial proposal (dashed line). Note that the cubic profile is not included because its performance does not substantially improves the linear profile behavior within the whole measurable ABw range and hence, it is not going to be used in the following sections. Table 5.2 summarize the efficiency parameters for optimized profiles, where it is shown that exponential is the most efficient and the less intrusive profile, as it uses the smallest probing packet size. Table 5.3 quantifies the improvements in the average relative error, achieved with the use of the optimized parameters.

Parameter		**Exponential**	**Logarithmic**	**Linear**
Load	$L_p(MB)$	1.05	1.91	1.49
Probing time	$T_p(ms)$	742.5	742.5	742.5
Average rate	$R_p(Mbps)$	11.9	21.6	16.9

Table 5.2: Efficiency parameters for the optimized profiles. The exponential is the most efficient and the less intrusive profile.

Profile	**Initial**	**Optimized**
Exponential	23.6	15.2
Logarithmic	18.9	17.1
Linear	19.2	14.2

Table 5.3: Comparison between the average relative error (%) of the initial and the optimized proposed profiles under PSD-CBR traffic.

5.3 Iterative pathChirp

Section 4.7 concludes that *pathChirp* is a very fast method in giving accurate estimations compared to other methods. This property can be useful when trying to improve the technique since time is not a constraint. This section

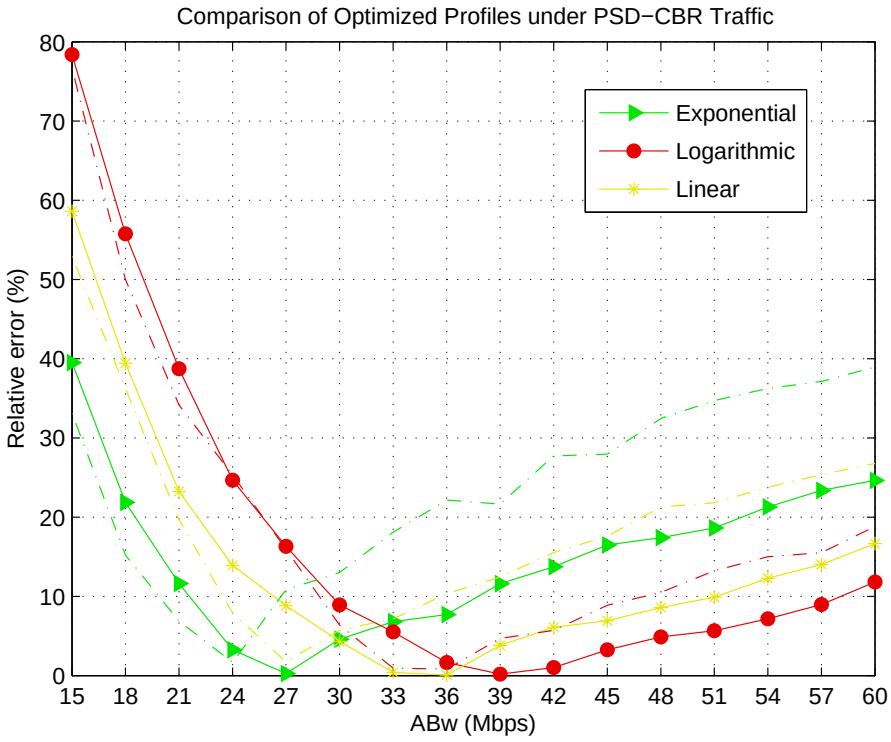

Figure 5.8: The relative error considerably decreases for the optimum parameters (solid line) compared to the initial proposal (dashed line).

proposes two alternatives of improving *pathChirp* making use of iterations. The idea is to take advantage of the result of last ABwE as an input for the method. In the first approach, the objective is to reduce the measurable range so as to increase the resolution, using the prior obtained ABwE. In the second approach, the best profile is used depending on the obtained ABwE in the previous iteration.

5.3.1 Zoom pathChirp

Zoom pathChirp is a variation of the original *pathChirp* proposal that consists of using the previous ABwE to fix the new measurable ABw range. By means of this approach, it is possible to load the network to what it is strictly necessary, avoiding probing the network with high rates when the ABw is low. It also allows the resolution to be increased since the measurable ABw range is narrower. To do so, the next iterative algorithm is followed (see Figure 5.9(a)):

1. A first estimation using the initial range $[R_{min}, R_{max}]$ is carried out so as to locate the ABw within the whole interval.

2. From the obtained ABwE, a narrower measurable range is fixed with the next equation

$$R_{max} = A + A_U \qquad R_{min} = A - A_L \tag{5.25}$$

where A_U and A_L are the ABw upper and lower error bounds respectively.

3. If the obtained ABwE is within a certain interval set by the following expression

$$R_{min} + A_{TH} < A < R_{max} - A_{TH} \tag{5.26}$$

where A_{TH} is the ABw threshold, then the algorithm goes to step 2.

4. Else, if the obtained ABwE is out of such interval, it is considered that a sudden change in the ABw has taken place, so the algorithm goes to step 1 in order to find as fast as possible the new stretch where the ABw is located.

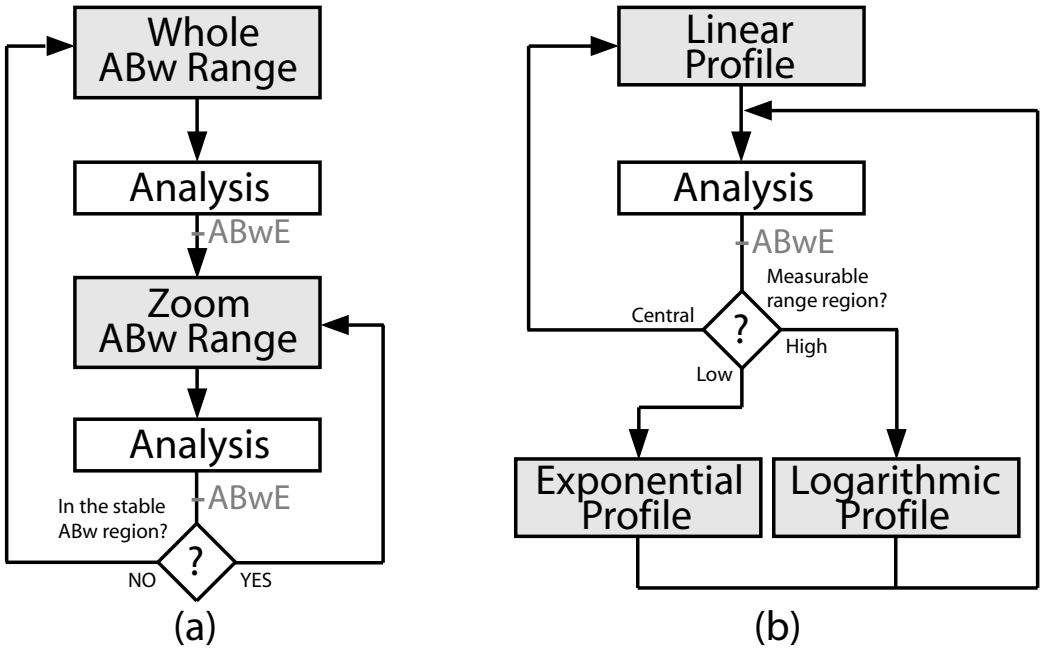

Figure 5.9: The two iterative *pathChirp* schemes are *Zoom pathChirp* (a) and *Adaptive pathChirp* (b).

Note that the number of packets K is adapted each iteration to keep the probing packet size between 800 and 1500 Bytes. The ABw bounds, i.e. A_U and A_L, are obtained from the results of the optimized linear profile evaluated in Section 5.1.3. *PathChirp* tends to underestimate, specifically the average error $\bar{\varepsilon}_A$ is -1.2Mbps with a standard deviation σ_ε of 2.5Mbps. In order to make up for such bias, the following ABw bounds[1] are selected

$$A_L = |\bar{\varepsilon}_A + \sigma_\varepsilon| = 4.53Mbps \quad (5.27)$$

$$A_U = |\bar{\varepsilon}_A - \sigma_\varepsilon| = 6.16Mbps \quad (5.28)$$

Concerning the ABw threshold, if A_{TH} is very small, sudden ABw changes are less likely to be detected, while very large A_{TH} makes the method mostly apply the whole range instead of a narrower interval. As a trade-off between both aspects, A_{TH} is determined as

$$A_{TH} = \frac{A_L + A_U}{F_A} = 2.68Mbps \quad (5.29)$$

where $F_A = 4$ is the ABw decrease factor. The different *Zoom pathChirp* thresholds are illustrated in Figure 5.10.

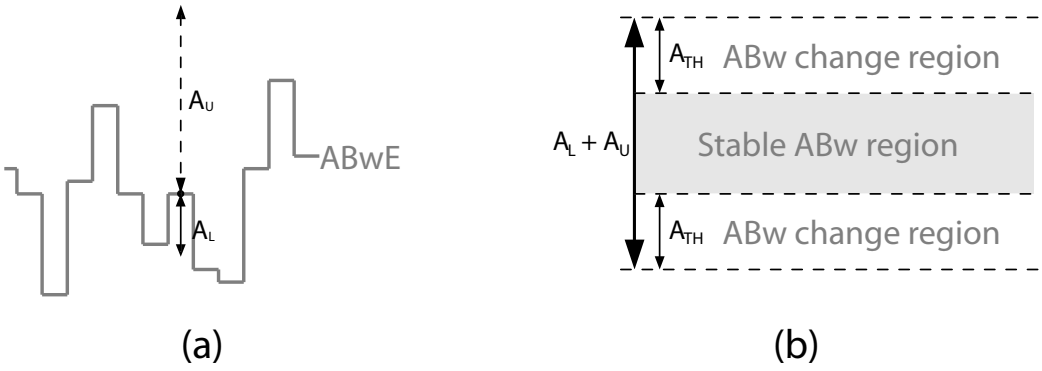

Figure 5.10: The *Zoom pathChirp* thresholds are the ABw upper and lower error bounds (a) and the ABw threshold (b).

5.3.2 Adaptive pathChirp

Adaptive pathChirp employs different chirp profiles taking advantage of the performance of each profile in the range close to the last ABwE. The ABw range is divided into three parts: the low, the central and the high stretches.

[1]Note that if the method tended to overestimate, these ABw bounds would also compensate such overestimation.

Figure 5.9(b) represents the scheme of the iterative algorithm whose steps are:

1. The linear profile is used to determine in which region of the measurable range the ABw is located, since it has a constant resolution within the whole range.

2. If the ABwE is in the low stretch, the exponential profile is used in the next iteration.

3. Else, if it is in the high stretch, the logarithmic is the next profile to be employed.

4. Otherwise, the linear profile is utilized again.

The three parts into which the measurable range is split are determined by two thresholds, empirically obtained from the different chirp profiles study of Figure 5.8. This figure denotes that the exponential profile is the best one when the ABw is lower than $A_{TH1} = 30Mbps$, and the logarithmic is the one to be used when the ABw is greater than $A_{TH2} = 36Mbps$. The linear profile is used when the ABw is between A_{TH1} and A_{TH2}.

5.3.3 Simulation Results

Figure 5.11 shows a comparative study of *Adaptive* and *Zoom pathChirp* under Poisson traffic that changes its average rate during time. The thresholds for both approaches taken from Sections 5.3.1 and 5.3.2. In 5.11(a), it can be observed that *Adaptive pathChirp* quickly adapts to the ABw changes, while *Zoom pathChirp* achieves better accuracy as the ABw remains constant. The relative error during time is shown in 5.11(b), from which it can be shown that, for an stable ABw, *Zoom pathChirp* achieves a better performance with time. The probing load for both methods is represented in 5.11(c). As expected, when the ABw is low, which is considered as a critical situation, *Zoom pathChirp* introduces less traffic, while *Adaptive pathChirp* is more efficient for a higher ABw. This results are also corroborated by a PSD-CBR simulation (see Figure F.7).

5.4 Linear Least Squares Fitting

Regardless of the measured ABw range and of the kind of cross-traffic, *pathChirp* shows a linear relationship with ABw, but the slope of such line

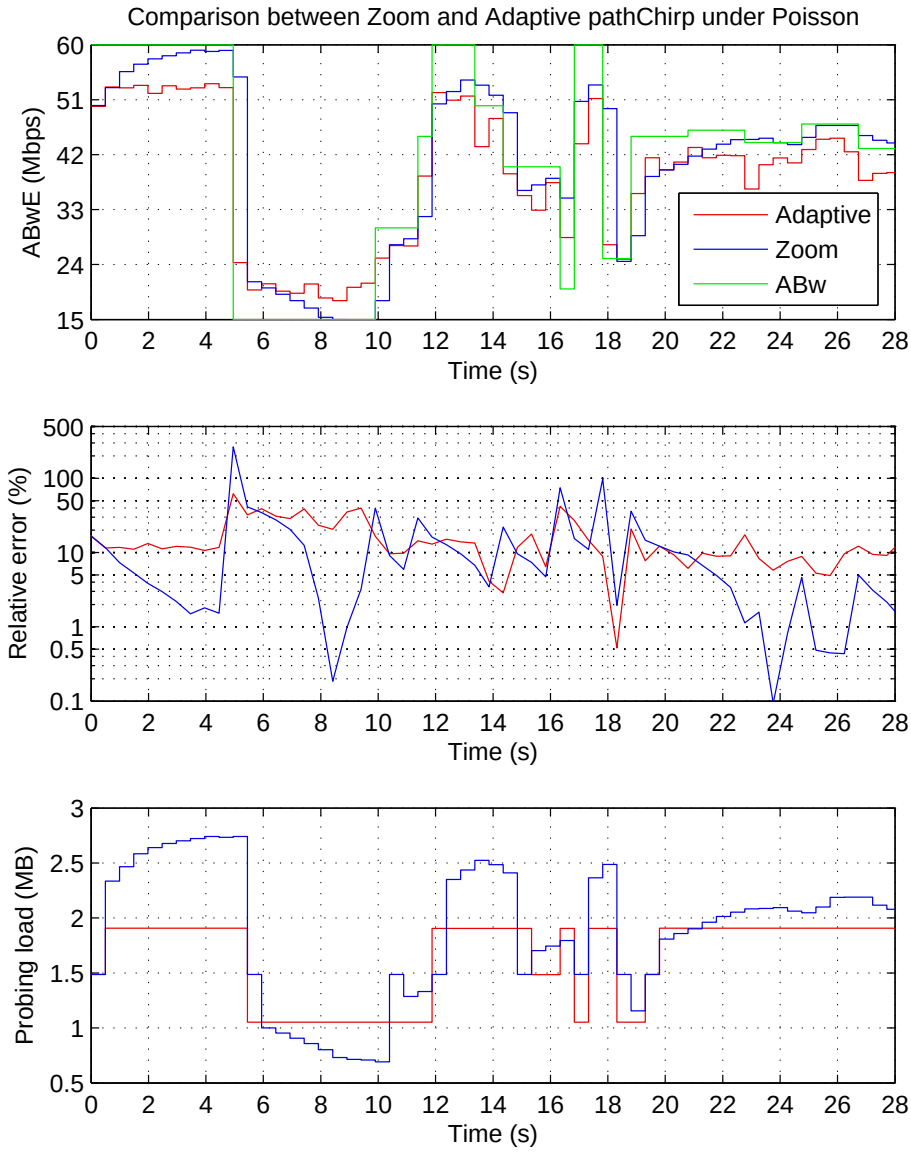

Figure 5.11: *Adaptive* and *Zoom pathChirp* comparison under Poisson with $P_x = 500B$, where (a) represents the ABwE, (b) shows the relative error and (c) plots the probing load during time.

is quite lower compared to the theoretical one, which is equal to 1. For instance, for the linear profile in Figure 5.4, the correlation coefficient is 0.9960 with a slope of $0.4854 < 1$. Taking advantage of this characteristic, a linear least squares fitting can be applied so as to reduce the error. If A and $\hat{A}$ are the real and the estimated ABw respectively, the error is

$$\varepsilon_A = \hat{A} - A \tag{5.30}$$

As both $\hat{A}$ and A are linear equations, the ABwE error can be expressed as $\varepsilon_A = \beta_A A + \alpha_A$. By substituting in Equation 5.30, a less biased estimate of the real ABw can be calculated as

$$A = \frac{\hat{A} - \alpha_A}{1 + \beta_A} \tag{5.31}$$

The problem lies in finding the most suitable regression parameters α_A and β_A regardless of the kind of traffic and of the network topology, which requires a thorough study in order to be used in a real network. As an example of the power of this method, the regression parameters for PSD-CBR traffic in a multi-hop path are obtained for the optimized linear profile from Section 5.2.3 as $\alpha_A = 13.1041 Mbps$ and $\beta_A = 0.6288$. Figure 5.12 compares the results with and without the least squares fitting, from which it can be stated that this statistical procedure considerably improves the accuracy, i.e. the average relative error decreases from 14.2% to 3.2%.

These results are reinforced by a Poisson traffic simulation shown in Figure F.8. Although the regression parameters have been obtained for PSD-CBR traffic, their use reduces the error from 13.4% to 7.8%. If specific regression parameters were calculated for Poisson traffic, the error would be 4.6%. Therefore, the least squares fitting is profitable, even without the optimum parameters.

5.5 Study of RTT Measurements

As stated in Chapter 1, it is beneficial for the proposed tool to be able to work without a destination agent that timestamps the packets arrivals. A possible way to avoid this agent is to send probing packets as if the goal was to take RTT measurements (e.g. using PING packets). The source agent would timestamp the received packets and would analyze these timestamps as if it was the destination agent. The hindrance of using RTT measurements and its possible solutions are described below.

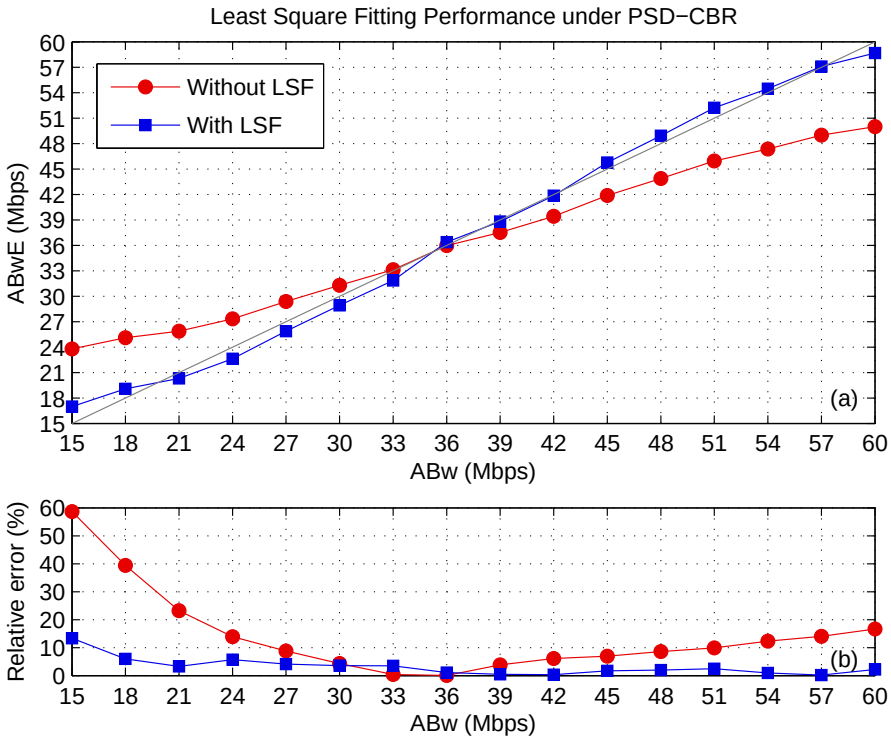

Figure 5.12: Least squares fitting comparison under PSD-CBR (a), where such adjustment considerably reduces the relative error (b).

5.5.1 Path Mirroring Effect

Assuming symmetric routing in a single-hop path, PING packets[2] are replied from the destination to the source and travel through the same link in the round-trip, so they see a 2-hop path. Let A_1 and A_2 be the ABw of the one-way and round-trip hops respectively (see Figure 5.13). The targeted ABw A_1 will only be estimated if $A_1 \leq A_2$, otherwise A_2 will be detected. It is not possible to know which ABw is measured and so, another approach is needed. This effect is corroborated by Figure 5.14(a), which shows a single-hop simulation under PSD-CBR, where A_2 is set to 30Mbps.

5.5.2 Non-intrusive Round-trip

As explained in Section 5.5.1, the round-trip leads to errors when $A_1 > A_2$. One possible solution to avoid relative delay variations in the round-trip is

[2]The packet size is assumed to be equal in both directions.

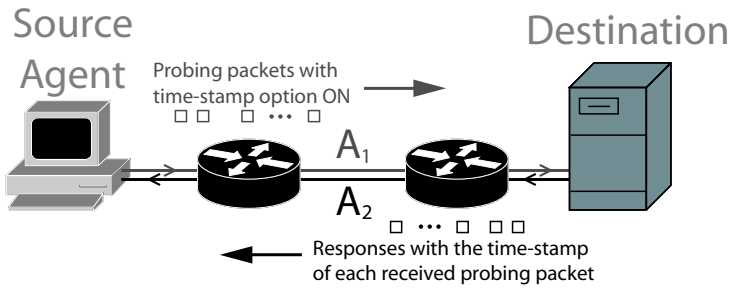

Figure 5.13: TCP timestamp option process.

to decrease round-trip rates below the ABw. To do so, it is necessary to reduce the round-trip packet size enough to make the maximum round-trip rate lower than the minimum measured rate, as the next equation shows

$$P_r \ll P\frac{R_{min}}{R_{max}} \tag{5.32}$$

For instance, $P_r \ll 288B$ for the linear profile to measure an ABw between 15Mbps and 60Mbps. Since it is not possible to modify P_r in PING command, a practical implementation would be the use of a TCP connection with unitary window to take advantage of the ACK (40 Bytes) as response. Nonetheless, the destination host should allow this TCP connection and accept all incoming probing packets. Figure 5.14(b) represents an example of this approach using the same configuration as in Figure 5.14(a). Note that P_r is fixed to 40 Bytes. From the figure, it can be stated that a very low P_r mitigates the effects of the round-trip.

5.5.3 TCP Timestamps Option

There is an option in the TCP protocol called *TCP Timestamps Option* [31] that makes the destination send the arrival timestamps in the replied packets. As Figure 5.13 shows, if this option was used, the source agent would receive the timestamps of all the sent packets and would be able to estimate the ABw of the one-way trip. So the mobile operator willing to estimate the ABw with only a source agent has to make sure that the TCP timestamping option is available at the destination.

5.6 Summary

In this chapter, different modifications to the original *pathChirp* proposal [6] are studied with the aim of improving the performance of the method:

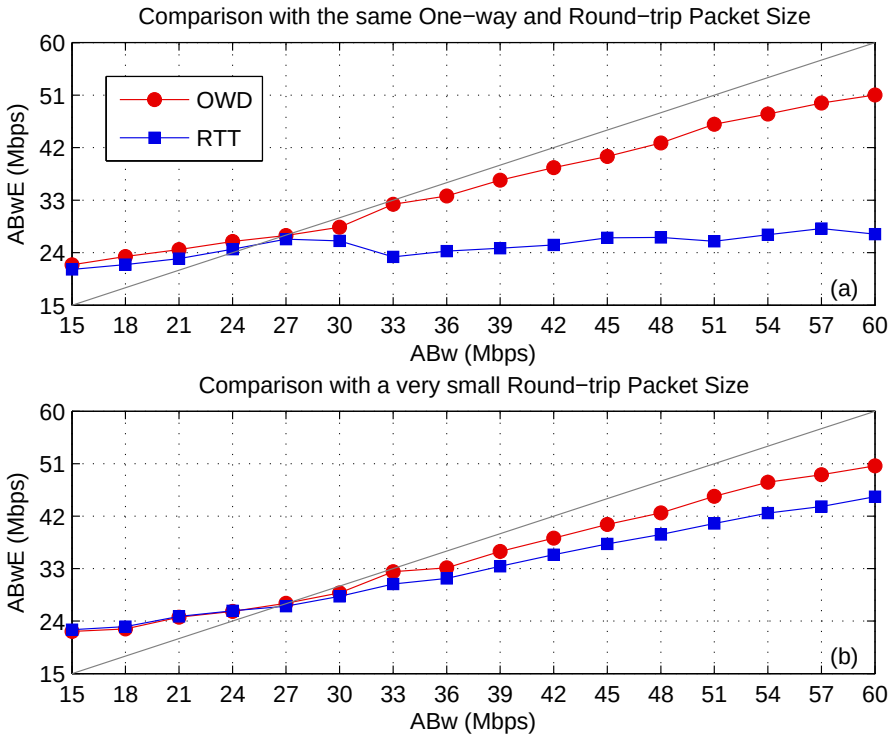

Figure 5.14: RTT measurements effect. The use of equally sized one-way and round-trip packets (a) leads to estimate the lowest ABw of both directions, in this case 30Mbps. This constraint can be mitigated by using a very low round-trip packet size (b).

- From the study of other chirp profiles and the optimization of the different *pathChirp* parameters, the optimized linear profile substantially improves the accuracy and reduces the variability of *pathChirp* regardless of the stretch of the measurable range the ABw is located in.

- In order to take advantage of the previous ABwE, two different iterative *pathChirp* approaches are proposed. One the one hand, *Adaptive pathChirp* quickly adapts to the ABw changes while *Zoom pathChirp* achieves better accuracy as the ABw remains constant. On the other hand, when the ABw is low, *Zoom pathChirp* introduces less traffic, whereas *Adaptive pathChirp* is more efficient for a higher ABw.

- A linear least squares fitting considerably improves the accuracy of *pathChirp*. However, the regression parameters depends on the kind of cross-traffic, which requires a thorough study of the real cross-traffic

characteristics to obtain optimum results.

Besides, this chapter includes a study of the use of RTT measurements so as to avoid the deployment of a destination agent. From this study, it is drawn that RTT measurements lead to errors when the tight-link is located in the round-trip path. One possible solution is to the decrease round-trip rates below the ABw by reducing the round-trip packet size. An alternative is to enable the TCP Timestamps Option when available, which makes the destination include the timestamp of each arrived probing packet in the corresponding replied packet.

Chapter 6

PathChirp under Differentiated Services

As explained in Section 2.4, all the studied AP methods assume FIFO queues. However, the targeted field of application in this report is a packet-switched mobile transport network, which may perform some kind of traffic prioritization. Such characteristic can affect the ABwE agent, so this chapter studies the effects of a *Differentiated Services (DiffServ)* [3] environment to the optimized linear profile proposed in Section 5.2.3. DiffServ allocates bandwidth to different flows achieving congestion avoidance, while it performs prioritization depending on the traffic application in order to guarantee a certain QoS to each flow. A detail description of the Diffserv architecture, the traffic conditioning, the traffic scheduling and the different per-hop behaviors (PHBs) can be found in Appendix G.

6.1 DiffServ Simulation Scenario

A DiffServ implementation varies from one network to another. The network administrator decides the different traffic classes and dropping precedences that routers will distinguish to guarantee a certain QoS to each flow. The different policies, the topology of the network and the thresholds for the different packet markers and packet droppers are also up to the network administrator. The proposed ABwE agent is evaluated in a simple DiffServ environment with up to four different traffic classes and two drop precedences.

6.1.1 DiffServ Simulation Topology

The topology for the DiffServ simulations follows the same idea of the topology used in Section 4.1.1, but taking into account the difference between edge and core routers. As Figure 6.1 illustrates, the tight-link has 155Mbps of capacity while the rest of the links are considered as joining links with a capacity of 10Gbps. All the latencies are set to 10ms. Each cross-traffic source sends packets of a different class at an average rate $R_{x,c}$ ($c = 1, \ldots, 4$). The cross-traffic rate R_x through the tight-link is computed as the sum of the average rates of the four cross-traffic sources regardless of the kind of traffic used, so the rate for each traffic class can be defined by a percentage $\rho_{x,c}$ over R_x as:

$$R_{x,c} = R_x \rho_{x,c} \tag{6.1}$$

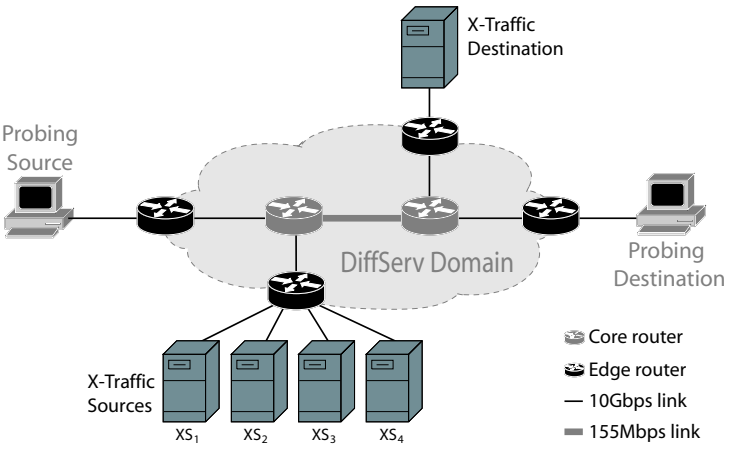

Figure 6.1: DiffServ simulation topology.

6.1.2 DiffServ Simulation Parameters

As it is previously mentioned, up to four different traffic classes are used, according to UMTS QoS specification [32]: *background*, *interactive*, *streaming* and *conversational*. Conversational services, like video telephony, are the most delay-sensitive applications, so EF PHB is used for this class. Streaming and interactive classes have a lower delay requirements, so AF PHB is used with only two dropping precedences. Background traffic is considered as BE. Poisson traffic is used to synthesize conversational and interactive traffic, while PSD-CBR traffic is used for background and streaming. Table 6.1 includes the DSCPs associated to each traffic class and drop precedence, together with the kind of cross-traffic generator.

Traffic Class	DSCP	Generator
Background	BE	PSD-CBR
Interactive	AF11-AF12	Poisson
Streaming	AF211-AF22	PSD-CBR
Conversational	EF	Poisson

Table 6.1: Marking and scheduling configuration for each traffic class.

Marking Configuration

The DiffServ simulation network should guarantee a certain QoS to each traffic class. When there is no network congestion, no packets are dropped, but when congestion is present, the network will drop packets to bring the flows into compliance with their traffic profiles. As only two drop precedences are used, there are two possible packet markers: *Token Bucket Marker (TBM)* and *Time-Sliding Window Two-Color Marker (TSW2CM)*. The latter is more simple, requiring only a *Committed Information Rate (CIR)*, so it is used to simplify the simulation model.

Scheduling Configuration

Priority (PRI) and *Weighted Round Robin (WRR)* scheduling are used to study the effect of the different prioritization modes to the performance of the ABwE technique. When WRR mode is used, the weights are selected proportional to the traffic priority. If all traffic classes are present in a queue, the share for conversational traffic Ψ_{EF} is proportional to its priority weight W_{EF} as

$$\Psi_{EF} = \frac{W_{EF}}{W_{BE} + W_{AF1x} + W_{AF2x} + W_{EF}} \tag{6.2}$$

EF packets require a determined priority to guarantee a certain data rate, independent of the load of other traffic classes. To do so, W_{EF} is selected to assure that $\Psi_{EF} \geq \rho_{x,EF}$, where $\rho_{x,EF}$ is the conversational rate percentage. Following this criteria, W_{EF} is set to

$$W_{EF} \geq ceil\left(\frac{W_{BE} + W_{AF1x} + W_{AF2x}}{1 - \rho_{x,EF}}\right) \tag{6.3}$$

where $ceil(X)$ rounds X to the nearest integer towards infinity. When PRI mode is selected, EF traffic has the highest priority, followed by AF2x, AF1x and BE traffics.

Dropping Configuration

The dropping configuration is the same for the four traffic classes, but using different thresholds. However, probing packets are never marked as out-of-profile since a dropped probing packet would make the method fail and the current estimation be discarded.

RIO[1] *Coupled (RIO-C)* droppers are chosen to assure that in-profile packets are forwarded while possible, by dropping more out-of-profile packets as the queue grows due to in and out-of-profile packets. Such discrimination against out packets is created by carefully selecting the dropping probabilities and congestion thresholds. According to [33], p_{max} should never be greater than 0.1 for the *Random Early Detection (RED)* algorithm to adapt smoothly as the average queue size changes. In their work, they use $p_{max} = 0.02$ for in-profile packets, but the optimal values for Q_{min} and Q_{max} are not defined since they depend on the maximum average delay that can be allowed by the gateway. As a general rule, it is suggested to use $Q_{max} \geq 2Q_{min}$.

Background and interactive traffics are very sensitive to packet loss, so they have a lower associated dropping probability p_{max} compared to streaming and conversational traffic, which allow some packet losses. The parameters, summarized in Table 6.2, are empirically obtained so as the routers trigger congestion avoidance mechanisms when the utilization is over 75%.

DSCP	Q_{min}	Q_{max}	p_{max}
BE	$Q/2$	Q	0.005
AF11	$Q/2$	Q	0.005
AF12	$Q/4$	Q	0.02
AF21	$Q/2$	Q	0.005
AF22	$Q/5$	$Q/2$	0.10
EF	$Q/2$	Q	0.10

Table 6.2: Dropping configuration for each DSCP, where Q is the router queue length.

[1] RED routers with In/Out bit

6.2 DiffServ Simulation Results

A DiffServ environment is scalable depending on the requirements of the network administrator. The study focuses on the configurations that are more likely to affect the ABwE method, i.e. the packet scheduling modes at the network routers queues and the priority of the probe packets with regard to the priority of the different cross-traffic flows.

6.2.1 Priority Scheduling Effects

To understand the effects of PRI scheduling on the performance of the method, two different simulations are carried out using only background and conversational traffic.

In the first simulation, represented by Figure 6.2(a), the ABw is fixed to 15Mbps, so the total cross-traffic rate is $R_x = 140Mbps$. The rate percentage $\rho_{x,EF}$ varies from 0.68 to 0.98 with $\rho_{x,BE} = (1 - \rho_{x,EF})$ leading to $R_{x,EF} \in [95, 137]Mbps$ and $R_{x,BE} \in [45, 3]Mbps$. It can be drawn that background traffic (BE) is invisible to conversational traffic (EF). When the probing traffic is pre-marked as EF, the method takes into account only such kind of traffic, and hence, it estimates the ABw as if conversational traffic was the only cross-traffic in the network. If BE probing traffic is used, it experiences delay due to both conversational and background traffic, which allows the ABw to be estimated.

In the second simulation, represented by Figure 6.2(b), $\rho_{x,BE} = \rho_{x,EF} = 0.5$ and the ABw varies from 15Mbps to 60Mbps. The figure corroborates the results shown in Figure 6.2(a). On the one hand, EF probing ought to estimate the ABw as if there was no other traffic, but it keeps almost constant because the observed ABw $(C - R_{x,EF})$ is always greater than $R_{max} = 60Mbps$. On the other hand, when BE probing is used, the underestimation increases since background traffic always has to wait for the conversational traffic queue to empty. This fact produces a undesired delay leading to errors.

6.2.2 WRR Scheduling Effects

Figure 6.3 shows the effects of the scheduling mode on the accuracy of the tool. It compares the performance of the technique under PRI and WRR with weights $W_{BE} = 1$ and $W_{EF} = 2$, following the same configuration as in Figure 6.2(b). From the figure, it can be seen that the results obtained with BE probing are not dependent on the scheduling mode because it still observes all the cross-traffic due to its low priority. Besides, the PRI scheduling effects

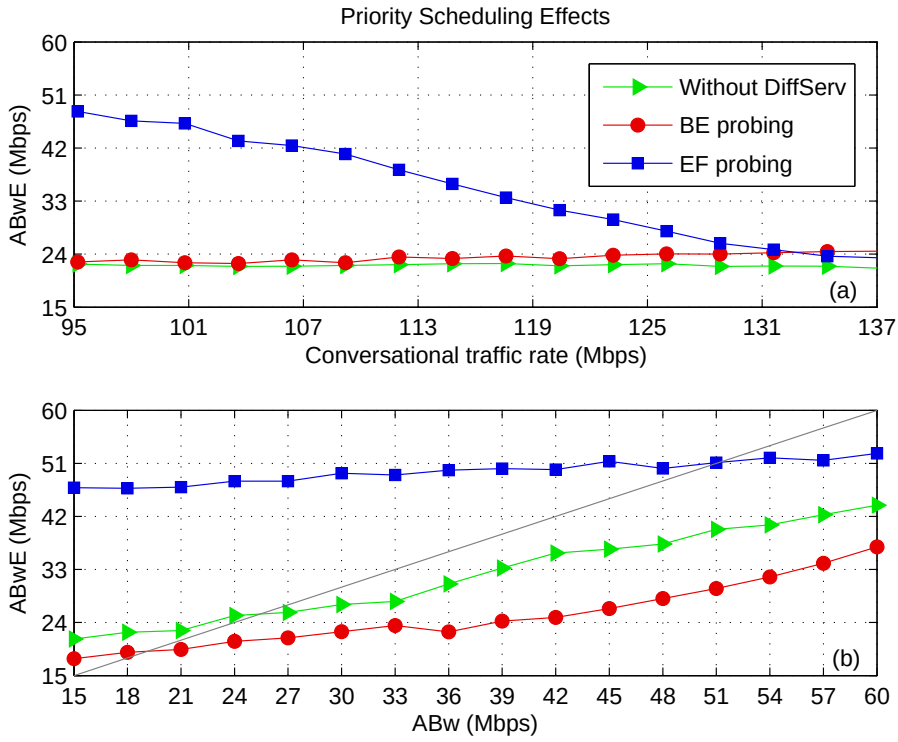

Figure 6.2: PRI scheduling effects with BE and EF traffic. In (a), $R_{x,EF}$ varies from 95Mbps to 137Mbps with $A = 15Mbps$ and $R_{x,BE} = (140Mbps - R_{x,EF})$. In (b), $\rho_{x,BE} = \rho_{x,EF} = 0.5$ and A varies from 15Mbps to 60Mbps.

on EF probing tend to mitigate as the EF traffic priority decreases, since the cross-traffic observed by the EF probe packets is not only the EF traffic but also part of the BE traffic.

6.2.3 Dependency on the Traffic Class Rate

In order to study how the rates of the different traffic classes affect the performance of the proposed ABwE agent, two simulations are carried out making use of WRR scheduling and the four traffic classes described in Section 6.1.2. The simulation setup is the same in both cases but the rate percentages for each class. The CIRs for each traffic class are selected to keep normal network operation when the utilization is below 75% and to trigger congestion avoidance mechanisms when the utilization is higher. The priority weights are chosen according to each traffic class priority, so that BE traffic has the lowest weight and AF1x and AF2x have equal weights. The weight for EF traffic is obtained using Equation 6.3. Table 6.3 includes the common

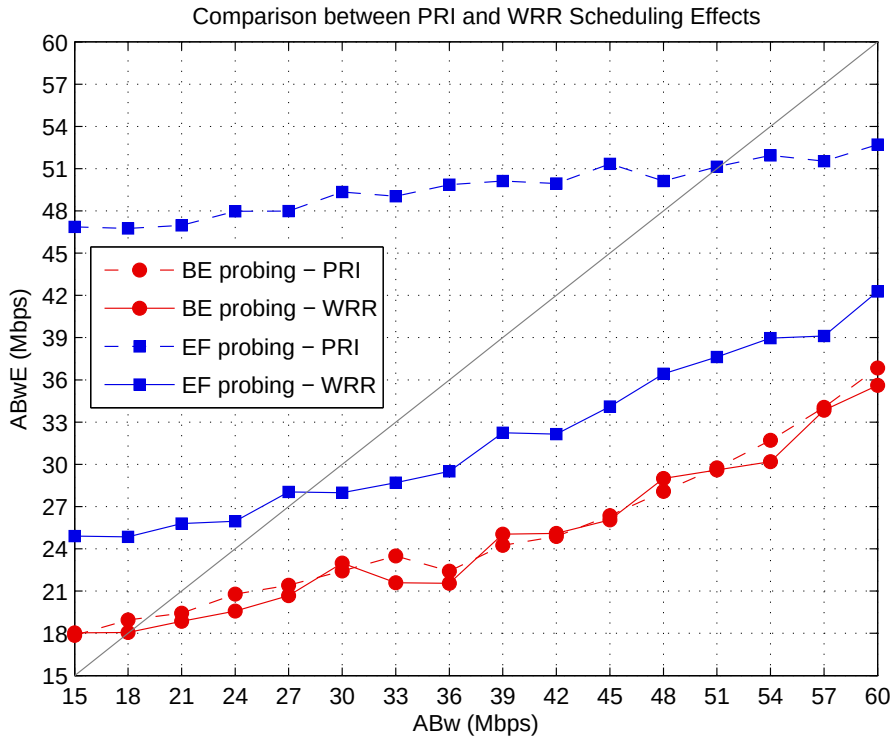

Figure 6.3: PRI and WRR scheduling effects comparison with $\rho_{x,BE} = \rho_{x,EF} = 0.5$. The results obtained with BE probing are independent of the scheduling mode. EF probing underestimates as the EF traffic priority decreases.

simulation setup.

In Figure 6.4(a), the rate percentages for all the traffic classes are the same so as to study the effect of adding more traffic classes on the performance of the ABwE method. In Figure 6.4(b), the rate percentages are $\rho_{x,BE} = 0.1$, $\rho_{x,AF1x} = \rho_{x,AF2x} = 0.2$ and $\rho_{x,EF} = 0.5$. These values are chosen as a forecast of the future mobile services usage, in which interactive and streaming services become more and more important.

When all the traffic classes have the same rate, the method behaves as expected, corroborating the results obtained in Sections 6.2.1 and 6.2.2. It is to say that the traffic with the lowest priority tends to estimate the whole traffic, whereas the other classes detect less traffic as their priority increases. Although classes AF1x and AF2x have the same priority, their performance is not exactly the same probably caused by the use of different kinds of cross-traffic, i.e. Poisson and PSD-CBR traffic respectively.

Although Figure 6.4(b) shows a similar trend, the difference between the

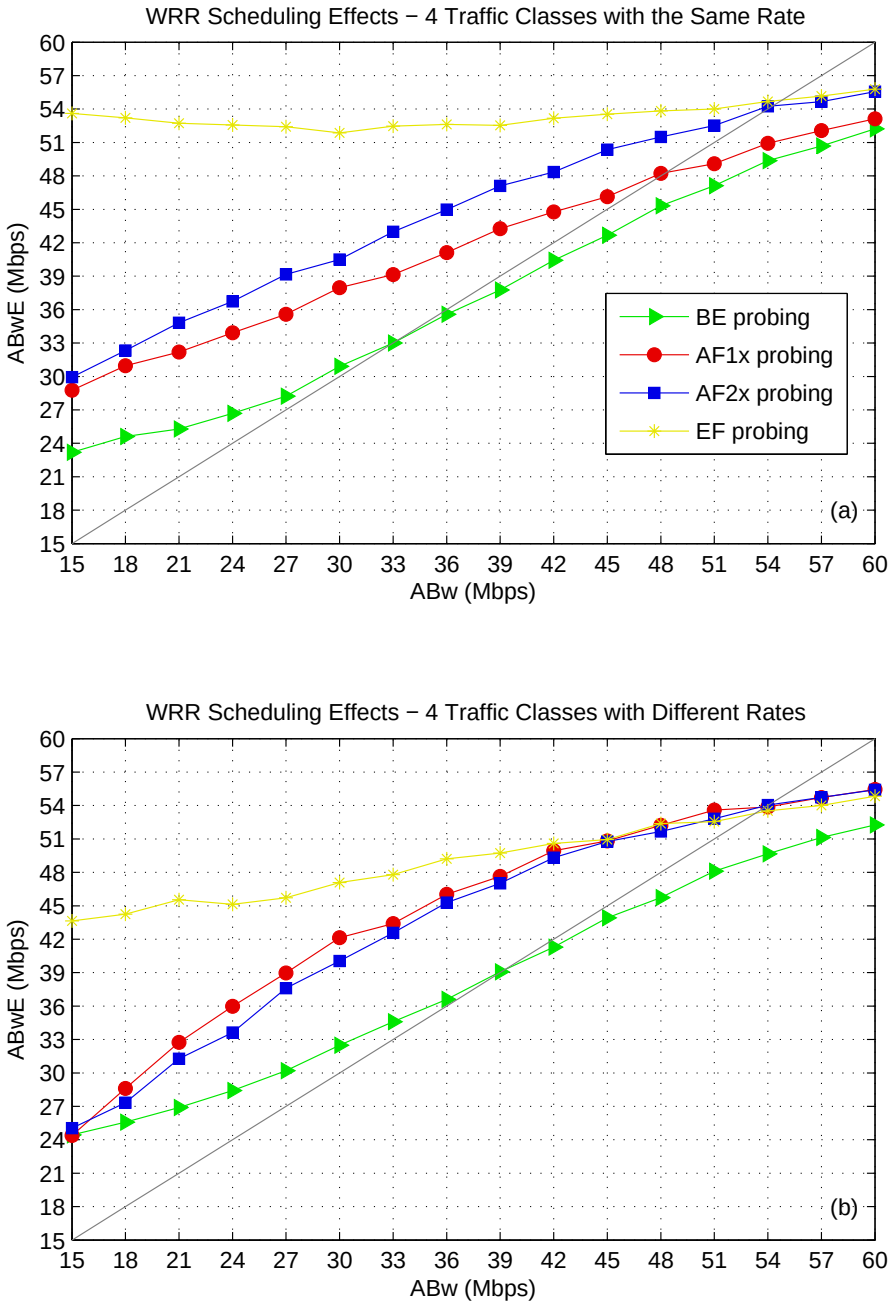

Figure 6.4: WRR scheduling effects with four traffic classes. In (a), $\rho_{x,BE} = \rho_{x,AF1x} = \rho_{x,AF2x} = \rho_{x,EF} = 0.25$. In (b), $\rho_{x,BE} = 0.1$, $\rho_{x,AF1x} = \rho_{x,AF2x} = 0.2$ and $\rho_{x,EF} = 0.5$.

DSCP	$CIR(Mbps)$	W_i
BE	17	1
AF1x	20	2
AF2x	23	2
EF	55	10

Table 6.3: Committed information rates and queue weights for the DiffServ simulation.

AF1x and AF2x probing estimations is lower than in Figure 6.4(a). Besides, since the percentage of EF traffic increases (from 25% o 50%), so does the load observed by EF probing packets and hence, the ABwE decreases compared to the case in which all the rates are equal.

6.3 Summary

In this chapter, the performance of the optimized linear profile proposed in Section 5.2.3 is evaluated in a DiffServ environment. From the simulations, it can be stated that the ABwE method is not only sensitive to the different scheduling modes, but also to the ratio between the rates of different traffic classes. It is shown that, when the probing traffic is inserted with highest priority (EF), the technique tends to estimate the ABw as if such traffic was the only cross-traffic in the network. Nonetheless, the use of the lowest priority (BE) for the probing traffic allows the method to observe all the cross-traffic. All things considered, DiffServ makes the results worse compared to a non-DiffServ domain. In case of employing *pathChirp* in a DiffServ environment, the best choice is the usage of BE priority class for the probing packets.

Chapter 7

Conclusions and Future Work

Knowledge of the Available Bandwidth (ABw) in a packet-switched mobile transport network can be very beneficial for the performance of the whole mobile system and the user's experience, as it could be used for RRM procedures like admission, load and handover control. Simulation studies have been conducted in this project to propose an ABw Estimation (ABwE) technique to be applied in current and emerging mobile communication networks. The main contributions of this report are a comparative study of three ABwE techniques (*TOPP*, *SLoPS* and *pathChirp*) taking into account the statistical conditions of the ABw, several improvements of *pathChirp* in terms of accuracy and efficiency, and a performance evaluation of *pathChirp* under a DiffServ environment.

7.1 Conclusions

Even though the performance of none of the investigated methods (*TOPP*, *SLoPS* and *pathChirp*) is found to be outstanding, *pathChirp* excels as the best tool in terms of both accuracy and efficiency. It shows no packet size dependency and an acceptable behavior in multi-hop environments under different cross-traffic models. The performance of *SLoPS* is found up to two times worse compared with *pathChirp* and it is around six times slower than *pathChirp* to give an estimation. *TOPP* is found to be very sensitive to the cross-traffic packet size (e.g. the average relative error[1] varies from 2% to 80%) and it is the slowest method.

Different modifications of the original proposal in [6] have been studied in order to optimize *pathChirp* in a multi-hop environment for load control

[1]Obtained under CBR traffic in a single-hop path.

purposes. The main findings from this study are:

- The optimization of the different profiles substantially improves the accuracy and reduces the variability of the initial exponential *pathChirp* profile. For instance, the average relative error[2] decreases from 23.6% to 14.2% if the optimized linear profile is used.

- The linear relationship with the ABw allows a linear least squares fitting to be applied, which reduces the average relative error of the optimized linear profile from 14.2% to 3.2%. However, the regression parameters depends on the kind of cross-traffic.

- Two different methods have been proposed for network load monitoring. *Adaptive pathChirp* quickly adapts to the ABw changes, while *Zoom pathChirp* achieves better accuracy as the ABw remains constant. For instance, the relative error[3] in *Zoom pathChirp* decreases from 10% to 1%, keeping around 10% in *Adaptive pathChirp*. In addition, *Zoom pathChirp* is less intrusive when the ABw is low, which avoids network congestion.

The use of RTT measurements to avoid the need for a destination agent leads to errors when the ABw in the round-trip path is lower than the ABw in the one-way path. One possible solution is to the decrease round-trip rates below the ABw by reducing the round-trip packet size. An alternative is to enable the TCP Timestamps Option when available, which makes the destination include the timestamp of each arrived probing packet in the corresponding replied packet.

The performance of *pathChirp* deteriorates in a DiffServ environment. Specifically, *pathChirp* is found not only sensitive to the different scheduling modes, but also to the ratio between the rates of different traffic classes. In case of employing Active Probing (AP) in a DiffServ environment, the best choice is to mark probing packets with Best Effort priority class at the source.

Taking into account that AP techniques are based on interfering the network to estimate the ABw assuming fluid cross-traffic and FIFO queuing, they were unlikely to show a very accurate performance under bursty cross-traffic and traffic prioritization, as it has been drawn from the different simulations. Therefore, the use AP has to be mainly focused on ABw trend detection for load control rather than for real-time adaptive QoS management.

[2]Obtained under PSD-CBR traffic in a 5-hop path.

[3]Obtained under Poisson traffic in a 5-hop path for an ABw around 40Mbps.

7.2 Future Work

From the studies carried out in this project, *SLoPS* and *TOPP* have shown low efficiency and accuracy as compared with *pathChirp* and thus all the optimizations were performed only on *pathChirp*. However, a detailed study of *SLoPS* parameters, and the use of Packet Trains instead of Packet Pairs in *TOPP* should be performed before concluding that *SLoPS* and *TOPP* are unsuitable for mobile networks.

Considering that the characterization of the ABw is affected by the cross-traffic properties and the queuing management at the routers, the practical issue regarding performing measurements in a live network could be used as reference for an investigation of a mobile system featuring handover, load, and admission control based on ABwE. To do so, it is necessary to modify the original *pathChirp* source code to implement the proposed improvements.

More thorough investigation should be done regarding the effects of multi-hop paths in *pathChirp* performance. Besides, it is uncertain how multi-paths affects the method, since packets of the same chirp may travel following different paths from the source to the destination. The employment of the different proposed algorithms, i.e. *Adaptive pathChirp*, *Zoom pathChirp* and *Least Squares Fitting*, could diminish the effects of DiffServ.

It would be interesting to use a method that periodically changes the measurable ABw range to give estimations for admission and load control purposes, since the former requires ABwE more often than the latter. Nonetheless, the accuracy for admission control should be improved first.

The different AP techniques are based on simple traffic and queuing models. It would be useful to describe a more complete mathematical delay model capable of capture the real network behavior so as to develop a more powerful ABwE technique.

The use of direct measurements instead of AP should not be fully discarded inside the mobile network infrastructure, where all the routers are assumed to be under control. A study of the frequency with which the different routers should report the bandwidth measurements to a centralized agent would be essential to make this approach light-weighted and practical.

Finally, the potential application of the proposed ABwE agent is an important issue for future investigation. On the one hand, algorithms for admission, load and handover control should be modified in order to take into account the ABwE. On the other hand, the practical implementation of the ABwE agent could require software updating in the involved network elements.

Bibliography

[1] Heikki Kaaranen, Ari Ahtiainen, Lauri Laitinen, Siamk Naghian, and Valtteri Niemi. *UMTS Networks*. John Wiley and Sons, 2005.

[2] Antti Toskala and Preben E. Mogensen. Utran long term evolution in 3gpp. *Proceedings WPMC '05 Wireless Personal Multimedia Communications*, September 2005.

[3] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An architecture for differentiated services. *RFC 2475*, December 1998.

[4] Diane Kiwior, James Kingston, and Aaron Spratt. Pathmon, a methodology for determining available bandwidth over an unknown network. *IEEE Sarnoff Symposium*, April 2004.

[5] Alok Shriram, Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, and Kimberly C. Claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In Constantinos Dovrolis, editor, *PAM*, volume 3431 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2005.

[6] Vinay J. 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*, April 2003.

[7] Andreas Johnsson, Bob Melander, and Mats Björkman. Bandwidth measurement in wireless networks. In *Mediterranean Ad Hoc Networking Workshop*, Porquerolles, France, June 2005.

[8] Manish Jain and Constantinos Dovrolis. Ten fallacies and pitfalls on end-to-end available bandwidth estimation. In Alfio Lombardo and James F. Kurose, editors, *Internet Measurement Conference*, pages 272–277. ACM, 2004.

[9] Eitan Altman and Tania Jiménez. Ns simulatior for beginners. December 2003.

[10] Ravi Prasad, Constantinos Dovrolis, Margareth Murray, and Kimberly C. Claffy. Bandwidth estimation: Metrics, measurements techniques, and tools. *IEEE Network*, November 2003.

[11] Behrouz A. Forouzan. *TCP/IP Protocol Suite*. McGraw-Hill Professional, 2002.

[12] Kihong Park and Walter Willinger (editors). *Self-Similar Network Traffic and Performance Evaluation*. John Willey, 2000.

[13] Henk C. Tijms. *Understanding Probability*. Cambridge University Press, 2004.

[14] Vinay J. Ribeiro, Mark Coates, Rudolf H. Riedi, Shriram Sarvotham, Brent Hendricks, and Richard Baraniuk. Multifractal cross-traffic estimation. In *Proc. of ITC Specialist Seminar on IP Traffic Measurement*, September 2000.

[15] Rudolf H. Riedi, Matthew S. Crouse, Vinay J. Ribeiro, and Richard G. Baraniuk. A multifractal wavelet model with application to network traffic. *IEEE Transactions on Information Theory*, 45(4):992–1018, 1999.

[16] Bob Melander, Mats Björkman, and Per Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. *IEEE Global Internet Symposium*, November 2000.

[17] Andreas Johnsson, Bob Melander, and Mats Björkman. DietTopp: A first implementation and evaluation of a simplified bandwidth measurement method. In *Second Swedish National Computer Networking Workshop*, page 5, Karlstad, November 2004.

[18] Bob Melander, Mats Björkman, and Per Gunningberg. Regression-based available bandwidth measurements. *International Symposium on Performance Evaluation of Computer and Telecommunications Systems*, July 2002.

[19] William J. Long. Real-time trend detection using segmental linear regression. April 2004.

[20] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Trans. Netw.*, 11(4):537–549, 2003.

[21] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth, 2002.

[22] Ningning Hu and Peter Steenkiste. Evaluation and characterization of available bandwidth probing techniques, 2003.

[23] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure? In *INFOCOM*, pages 905–914, 2001.

[24] Aviva Garrett, Susan B Stillwell, Gary Drenan, and Cris Morris. *Juniper Networks Field Guide and Reference*. Addison-Wesley Professional, 2002.

[25] Sean McCreary and Kimberly C. Claffy. Trends in wide area IP traffic patterns - a view from Ames Internet exchange. *ITC Specialist Seminar*, 2000.

[26] Erol A. Peköz and Nitindra Joglekar. Poisson traffic flow in a general feedback queue. *Journal of Applied Probability*, September 2002.

[27] Attila Pasztor and Darryl Veitch. The packet size dependence of packet pair like methods. *IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, 2002.

[28] CISCO White Papers. Performance measurements of advanced queuing techniques in the CISCO IOS, 2000.

[29] Miikka Lundan and Igor D. D. Curcio. Mobile streaming services in wcdma networks. *iscc*, 00:231–236, 2005.

[30] Alcatel Telecomunications White Papers. Mobile backhaul: from backstage to spotlight, 2004.

[31] Van Jacobson, Robert Braden, and David Borman. Tcp extensions for high performance. *RFC 1323*, May 1992.

[32] Technical Specification 23.107. Version 3.2.0: Qos concept and architecture, March 2000.

[33] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[34] Timo Halonen, Javier Romero, and Juan Melero. *GSM, GPRS and EDGE Performance*. John Wiley and Sons, 2003.

[35] James Curtis and Tony McGregor. Review of bandwidth estimation techniques, 2001.

[36] Allen B. Downey. Using pathchar to estimate internet link characteristics. In *Measurement and Modeling of Computer Systems*, pages 222–223, 1999.

[37] Attila Pásztor and Darryl Veitch. Active probing using packet quartets, 2002.

[38] Ravi Prasad, Constantinos Dovrolis, and Bruce A. Mah. The effect of layer-2 store-and-forward devices on per-hop capacity estimation, November 2002.

[39] Kevin Lai and Mary Baker. Measuring link bandwidths using a deterministic model of packet delay. In *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 283–294, New York, NY, USA, 2000. ACM Press.

[40] Donald A. Berry and Bernard W. Lindgren. *Statistics: Theory and Methods*. Duxbury Press, 1996.

[41] Morris H. DeGroot. *Optimal Statistical Decisions*. Wiley-IEEE, 2004.

[42] Thomas Karagiannis, Mart Molle, and Michalis Faloutsos. Long-range dependence: Ten years of internet traffic modeling. *IEEE Internet Computing*, 8(5):57–64, 2004.

[43] Murad S. Taqqu, Walter Willinger, and Robert Sherman. Proof of a fundamental result in self-similar traffic modeling. *ACMCCR: Computer Communication Review*, 27, 1997.

[44] Jurgen Gross. *Linear Regression*. Springer, 2003.

[45] Philip Rabinowitz Anthony Ralston. *A First Course in Numerical Analysis*. Courier Dover Publications, 2001.

[46] Mike Flannagan, Richard Froom, and Kevin Turek. *Cisco Catalyst Qos*. Cisco Press, 2003.

[47] David D. Clark and Wenjia Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, 1998.

[48] Wenjia Fang, Nabil Seddigh, and Biswajit Nandy. A time sliding window three colour marker. *RFC 2859*, June 2000.

[49] Peter Pieda, Jeremy Ethridge, Mandeep Baines, and Farhan Shallwani. A network simulator differentiated services implementation. *Open IP, Nortel Networks*, July 2000.

[50] Juha Heinanen and Roch Guerin. A two rate three color marker. *RFC 2698*, September 1999.

[51] Juha Heinanen and Roch Guerin. A single rate three color marker. *RFC 2697*, September 1999.

[52] Juha Heinanen, Fred Baker, Walter Weiss, and John Wroclawski. Assured forwarding phb group. *RFC 2597*, June 1999.

[53] Bruce Davie, Anna Charny, Jon Bennett, Kent Benson, Jean-Yves Le Boudec, Bill Courtney, Shahram Davari, Victor Firoiu, and Dimitrios Stiliadis. An expedited forwarding phb. *RFC 3246*, March 2002.

Appendix A

Networking Basics

A.1 Network Architectures

A network is an interlinked system of devices, usually called *nodes*, which share resources and information. A network is formed by several end nodes or *hosts* joined by intermediate nodes, which distribute the information to the different destinations. The *Open System Interconnection (OSI) Reference Model* [11] is a standard which establishes the rules or *protocols* between the different network elements and their functions. This model divides a network into layers, and each layer has a certain function and a fixed interface to communicate with the adjacent layers.

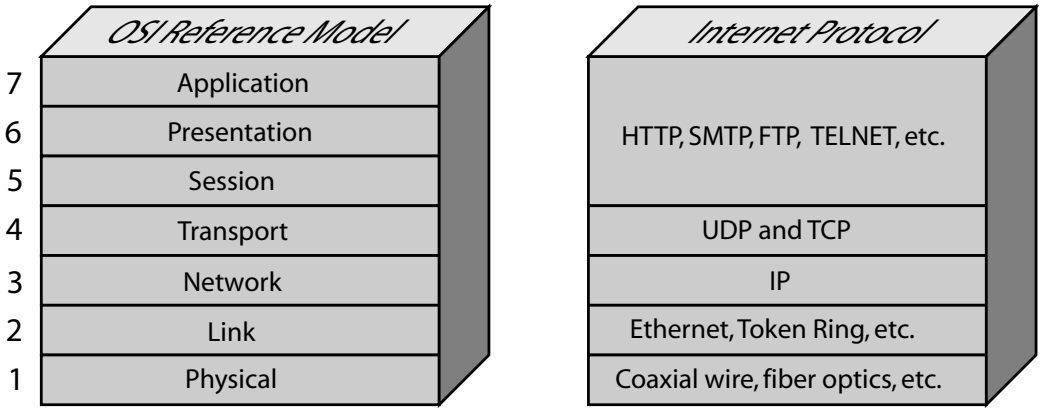

Figure A.1: Comparison between the *OSI Reference Model* and the *Internet Protocol*.

The OSI Reference Model has a practical implementation known as *The Internet Protocol* (see Figure A.1). The *physical layer* determines the char-

acteristics of the transmission media (wired or wireless), voltages and connectors. The *link layer* or *layer-2* controls the communication within a single network, where the information is distributed by devices such as *switches*. The *network layer* or *layer-3* is responsible for transferring information from a source to a destination, which are joined by devices called *routers*, through several heterogeneous networks (different link-layer protocols and topologies). The *transport layer* establishes a reliable end-to-end connection without taking into account the intermediate path. The *upper layers* (session, presentation and application) carry out functions such as synchronism, information encoding and user-services supplying (e-mail, web browsing, etc.). Let define *segment* as a link that joins two layer-2 devices, *hop* as a link that joins two layer-2 devices and *path* as a sequence of consecutive hops joining a source with a destination (see Figure A.2).

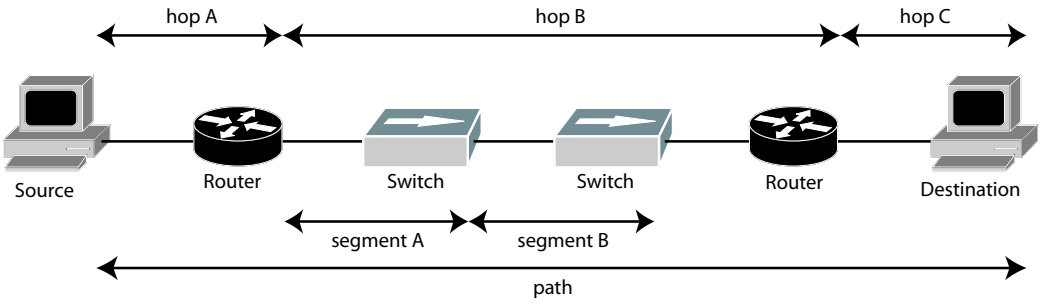

Figure A.2: Network topology elements.

A.2 Mobile Network Architectures

Mobile phone networks have rapidly spread throughout the world since the introduction of the cellular systems around the 1980s, making users substitute their land-line phones for mobile terminals in some cases. The network architecture of any mobile system is divided into *User Equipment (UE)*, *Radio Access Network (RAN)* and *Core Network (CN)*. The UE is the user terminal, for instance, a mobile phone, whereas the RAN is the part of the network that handles the air interface between the UE and the CN. This section presents an overview of the network elements that compose the network architecture of the three most representative mobile systems, illustrated in Figure A.3.

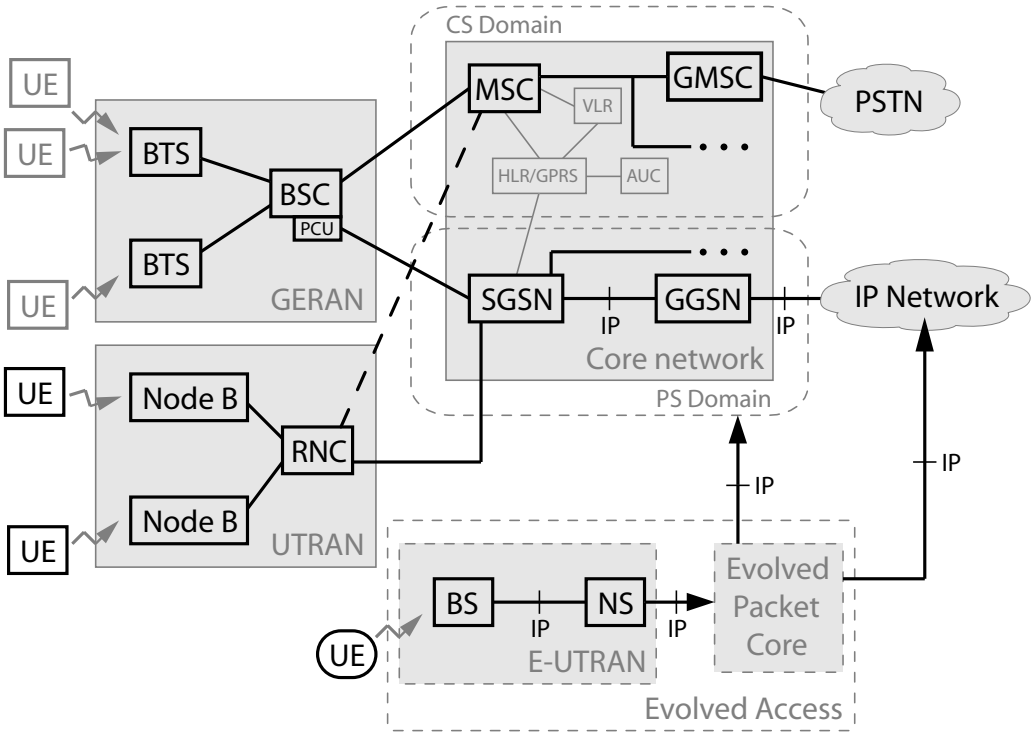

Figure A.3: Simplified mobile systems inter-working network topology.

A.2.1 GSM and GPRS/EDGE

In 1991, the *Global System for Mobile Communications (GSM)* [34] was commercially introduced, turning into the most popular mobile system standard in the world. It is based on *Circuit Switching (CS)* and makes use of a hybrid multiplexing scheme known as FDMA/TDMA[1] with GMSK[2] modulation. It is considered a second generation (2G) system.

An evolution of GSM called *General Packet Radio Service (GPRS)* [34] was released in 2000. It takes advantage of the CS of GSM to route the voice calls, whereas it introduces *Packet Switching (PS)* to handle the data calls. It provides higher data transfer than GSM by allocating more than one time-slot of the TDMA structure to a single user. It is usually classified as a 2.5G system.

Enhanced Data rates for GSM Evolution (EDGE) [34] was firstly available in 2003. It is implemented over the GPRS architecture, so they share most of the network elements. In addition to GMSK, EDGE uses 8PSK[3] modulation,

[1]Frequency and Time Division Multiple Access.
[2]Gaussian Minimum Shift Keying.
[3]8 Phase Shift Keying.

which triples the data rates offered by previous systems. For its deployment, it is necessary to install EDGE-compatible transceivers at the BTS and a software upgrade to the *GPRS/EDGE Radio Access Network (GERAN)*, composed of BTS and BSC. The following lines give a short description of the most relevant GPRS/EDGE mobile network elements for the purpose of this report:

- The *Base station Transceiver Subsystem (BTS)* contains the equipment to transmit and receive radio signals (transceivers), antennas, and equipment to encrypt and decrypt the communication with the BSC.
- The *Base Station Controller (BSC)* manages several BTSs. Among other functions, it handles the allocation of radio channels, receives measurements from UEs and controls handovers between BTSs. *The Packet Control Unit (PCU)* extents the functionality of a BSC for packet data.
- The *Mobile services Switching Center (MSC)* controls several BSCs. It provides CS calling and routing, billing and mobility management. A *Gateway MSC (GMSC)* bridges the mobile telephone network with the *Public Switched Telephone Network (PSTN)*.
- The *Serving GPRS Support Node (SGSN)* performs similar functions as the MSC but in the PS domain. Since it is connected to the GGSN through an IP-based path, it does the tunneling of the user data.
- The *Gateway GPRS Support Node (GGSN)* is the interface between the GPRS/EDGE network and the external PS network. It assigns IP addresses to the UEs.

The *Medium Access Control (MAC)*, is a GERAN link layer protocol to manage the user radio resources. It can make several users share the same channel or assign several channels to a single user to increase its data rate.

A.2.2 UMTS

Universal Mobile Telecommunication System (UMTS) [1] is one of the 3G mobile phone technologies. It is based on WCDMA[4], allowing higher data rates than GPRS/EDGE. The core network architecture is based on GPRS. Its main difference is the RAN, in this case called the *UMTS Terrestrial Radio Access Network (UTRAN)*, which consists of:

[4]Wideband Code Division Multiple Access.

- The *Node-B*, equivalent to the BTS, processes all physical layer data related to the cells under its control.
- The *Radio Network Controller (RNC)*, equivalent to the BSC, controls several Node-Bs and their associated cells. It performs radio resource control and management of the radio carriers per user. It is also responsible for the user mobility management.

The *Radio Resource Management (RRM)* is a network layer functionality present in UTRAN. It manages the network and user resources at the air interface. Among other functions, it performs admission control, congestion control and packet scheduling.

A.2.3 E-UTRAN

Evolved UTRAN (E-UTRAN) is the RAN technology for the next mobile phones generation, due to be available in 2010 [2]. Although its architecture is still open, it is intended to work with previously described systems, but only in the PS domain, as it will be completely IP-based. It will not have connection to the MSC, so the voice services will be handled as Voice over IP (VoIP).

Appendix B

Capacity Estimation Techniques

The study of *Capacity Estimation techniques* is useful to obtain a background knowledge in AP. Some of the ABwE techniques are based on them, so they share common problems. Moreover, some ABwE need an estimation of the tight-link capacity in order to estimate the ABw. In the next sections, the two main groups of Capacity Estimation techniques, i.e. *One Packet* and *Packet Pair techniques*, are reviewed, focusing on the second one, as its foundations are used in some ABwE techniques. In addition, some other new methods, commonly called *Mixed techniques*, are mentioned.

B.1 One Packet Techniques

One Packet (OP) techniques [10, 35] estimate the capacity of individual hops. These techniques take advantage of the linear relationship between the OWD and the probing packet size (see Equation 2.9) to estimate the capacity. In order to avoid deploying special software in each intermediate router of the path to record the arrival timestamps, RTT measurements are taken. The Time-To-Live (TTL) field of an IP header [11] is decreased in each router and an ICMP[1] response is sent back to the source when the TTL expires. So, the hop of study can be targeted by fixing such TTL. Considering that the ICMP response packet size remains constant and the latency of each link does not change during the probing time, the RTT delay from Equation 2.10 can be rewritten as follows,

$$RTT_k^h = \sum_{s=1}^{h} \frac{P_k}{C_s} + \sum_{s=1}^{h} \left(\frac{P_r}{C_s} + 2d_s \right) + \sum_{s=1}^{h} (q_s + q_s^r) \tag{B.1}$$

[1]Internet Control Message Protocol [11]

where the *first term* is linear with the probing packet size allowing the estimation of the capacity, and the *second one* does not depend on the probing packet size, adding just an offset. The *third term* is random, being a source of error. An alternative way of calculate the RTT consists of using PING[2] command [11]. Instead of fixing the number of hops by the TTL, the IP address of the targeted node is needed.

Depending on the way these RTT measurements are processed, different methods have been developed. One of the most well-known techniques is called *Variable Packet Size (VPS)* [10, 35, 36]. In order to minimize the effect of the random term of Equation B.1, this method sends several packets per a given size, assuming that at least one of them and its ICMP response will not be queued, leading to the *Shortest Observed RTT (SORTT)*. So, Equation B.1 can be simplified,

$$RTT_k^h = \alpha_h + \beta_h P_k \tag{B.2}$$

where α_h is the constant term of Equation B.1 and $\beta_h = \sum_{s=1}^{h} \frac{1}{C_s}$ is the slope of the minimum RTT against the probing packet size.

Several probing packets of different sizes are sent in order to calculate β_h by performing a linear regression (see Figure B.1). To estimate the capacity of a certain link, it is necessary to calculate the β parameter of that link and the previous one by using the next equation,

$$C_h = \frac{1}{\beta_h - \beta_{h-1}} \tag{B.3}$$

In order to determine the narrow-link, it is required to estimate the capacity of all the links of path and to apply Equation 2.1. However, VPS does not make an *efficient use* of the probing packets, since it only takes information from the minimum delayed packets. Moreover, the use of the SORTT filtering does not lead to accurate results under *heavy-load conditions*, because all of the probing packets could be delayed. In order to solve these problems, there is another technique called *Accumulation Signature (AccSig)* [37] that alternatively sends packets of only two sizes and measures the RTT variation between two consecutive packets. With this information, a histogram is performed with the distribution of the obtained values. The capacity is estimated by analyzing the two modes[3] of the histogram caused by the difference in size of the packets.

There are many factors that can disturb the RTT measurements. *Asymmetric routing* happens when the probing packet and its response do not follow the same path, leading to incorrect estimations. RTT measurements

[2]Packet INternet Groper

[3]mode: the most frequently occurring value in a set of discrete data.

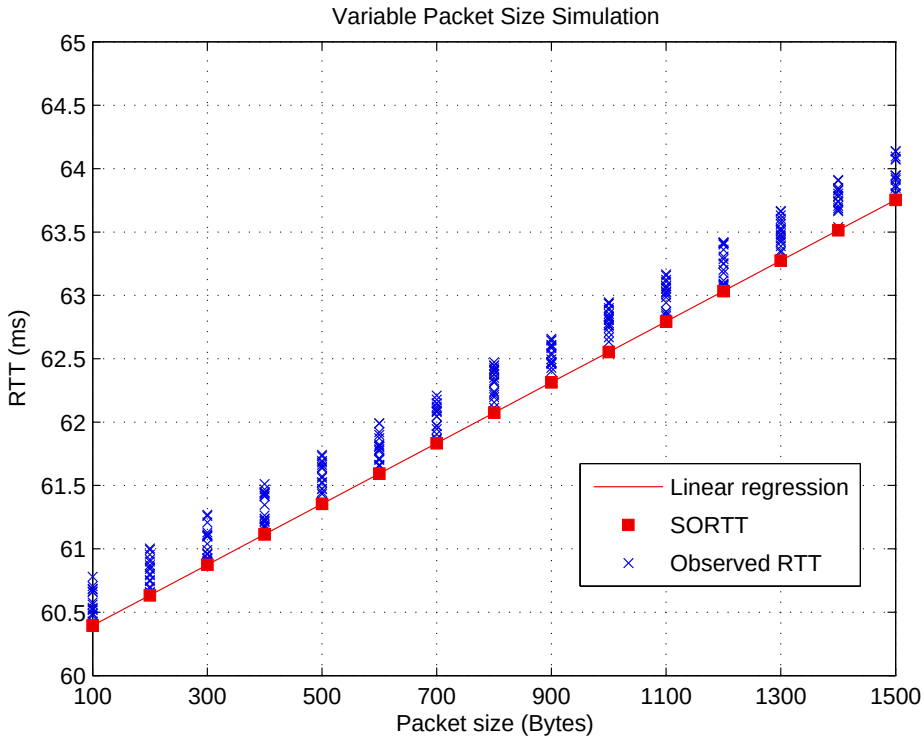

Figure B.1: Variable Packet Size simulation example. Several packets per size are sent and a linear fit is performed using the minimum delayed ones.

are also *ICMP dependant* because some routers consider ICMP responses as rubbish or potentially dangerous and they filter them out. Not only that, RTT measurements are also *layer-2 dependant* [38] because some devices, such as switches, usually use store-and-forward techniques, but they are invisible to routers and cannot decrease the TTL field.

B.2 Packet Pair Techniques

A *Packet Pair (PP)* [10, 35] consists of two packets, usually with the same size, that are sent back-to-back through the path. Unlike the techniques mentioned in the previous section, *PP* probing directly gives a value for the capacity of the *narrow-link*, with no additional information of the capacities of other links in the path. PP techniques are based on the transmission delays that packets suffer in their way from the source to the destination.

During the probe, the source sends multiple *PP*s to the destination. The time distance between the last bit of the first and second packets is the

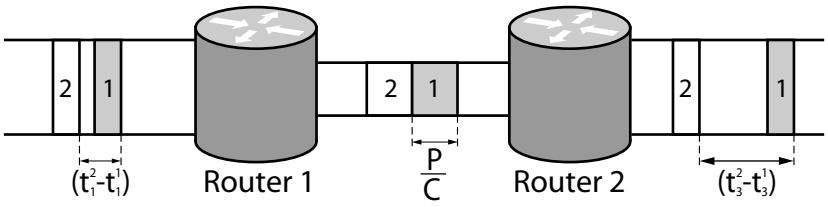

Figure B.2: Packet Pair dispersion.

dispersion of the PP. Figure B.2 represents three links joined by routers with no additional traffic but the PP. As the second link has half the capacity of the first one, the second packet of the pair has time enough to be fully received and queued by the router before the first packet is completely sent. As a result, the second packet will be sent back to back with the first one as the example shows. Due to the high capacity of the third link, the first packet will be completely sent before the second one is totally received by the second router, so the time gap between both packets will increase. Let the second link be the narrow-link of a H-hop path. Given Equation 2.9, and assuming absence of cross-traffic, the dispersion measured by the destination will be

$$\Delta = \max_{h=1,\ldots,H} \left(\frac{P}{C_h} \right) = \frac{P}{\min_{h=1,\ldots,H} (C_h)} = \frac{P}{C} \tag{B.4}$$

provided that the packets are sent back-to-back.

The main advantage of this method is that it performs the measurement of inter-arrival times between packets only at the end host. This fact avoids the problem of asymmetric routing, ICMP dependency and layer-2 effects of RTT-based Capacity Estimation methods. However, this technique is really sensitive, not only to the probing packet size and user time resolution, but also to the cross-traffic.

First, the *cross-traffic* can either increase or decrease the dispersion Δ. If other traffic queues between the *PP* at a certain link, the dispersion increases leading to capacity underestimation, while packets in front of the first packet can make it queue and wait for the second packet, resulting in capacity overestimation. To mitigate these effects, many authors [27, 23] have made use of statistical filtering of the dispersion distribution. Although the detection of the capacity is based on the mode, more recent work has revealed the underlying distribution to have multiple modes due to the queuing of the cross-traffic. For instance, in [27][23], it is proved the dependency of this technique with the cross-traffic packet size and with the probing packet size.

Figure B.3 shows a simulation example of *PP* technique. The simulation scenario consists of an 8-hop path with capacities $C_h \in \{100,\ 75,\ 55,\ 50,$

100, 60, 40, 80}, through which 400 *PP*s are sent. In theory, the capacity of the narrow-link should be distinguished from all the obtained values of the histogram. However, the presence of heavy cross traffic in (a), (b) and (c) makes impossible to decide wether the capacity of the path is the main mode of the histogram or another local mode.

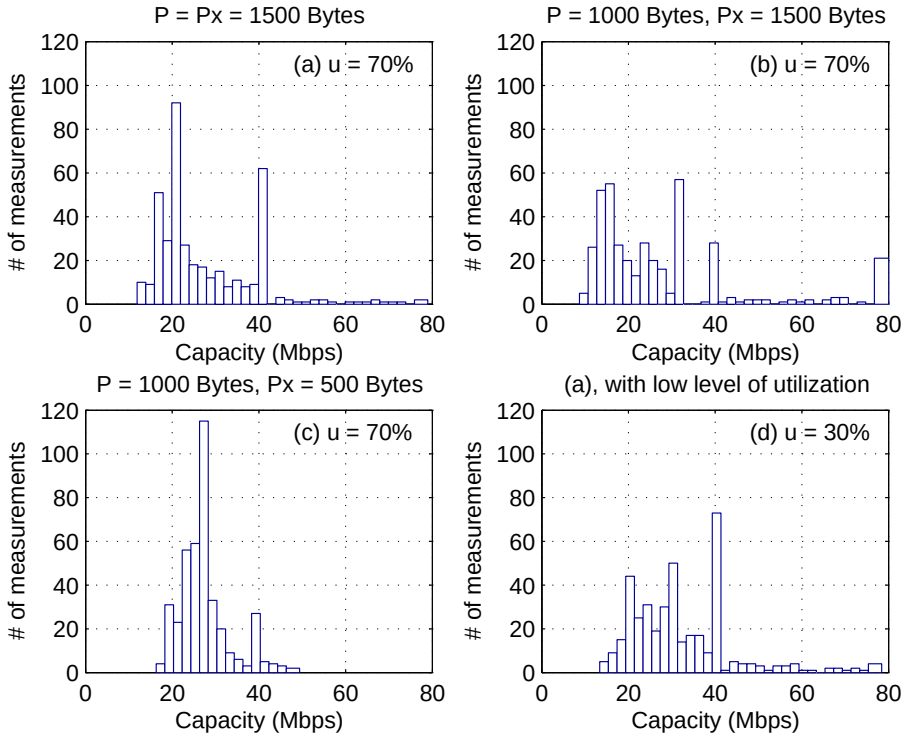

Figure B.3: Packet Pair simulation example of an 8-hop path with $C_h \in \{100, 75, 55, 50, 100, 60, 40, 80\}$. In each histogram, the probing packet size P and the cross-traffic packet size P_x vary. In (a), (b) and (c), the average utilization u of each hop is 70%, whereas in (d), it is only 30%.

Local modes on the left side of the capacity mode (centered on 40Mbps in this case) appear when other traffic packets arrive at a certain hop between the packets of a *PP*. It is easier to explain those modes when the size of all the packets is equal. For instance, in Figure B.3(a), the local mode at 20Mbps can be caused by a packet interfering with the PP at the 40Mbps hop (see Figure B.4(a)) as the time gap between both packets after the narrow-link is

$$\Delta = \frac{P}{40Mbps} + \frac{P_x}{40Mbps} = \frac{P}{20Mbps} \qquad \text{if } P = P_x \tag{B.5}$$

where P_x is the cross-traffic packet size. It can also be produced by two packets at the 60Mbps link (see Figure B.4(b)).

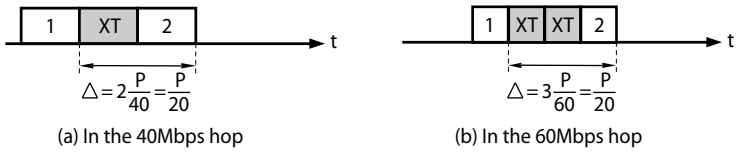

Figure B.4: Related to Figure B.3(a), the generation of local modes at 20Mbps is due to either one cross-traffic packet within a PP at the 40Mbps hop (a) or two cross-traffic packets at the 60Mbps hop (b).

Second, *user time resolution* sets a maximum in the capacity that this method can estimate since the minimum dispersion that the destination can measure is determined by the latency to receive a packet in the OS[4] and to move it from kernel to user space. For example, with $\Delta_{min} = 100\mu s$ and $P = 800B$, the maximum capacity that can be measured is 64Mbps.

Finally, the selection of the appropriate *probing packet size* is very important. Higher P leads to larger dispersion, which is more robust to queuing delay noise and less sensitive to the timestamping resolution at the destination. Nevertheless, the larger the P, the higher the likelihood of cross-traffic arrival. A minimum sized packet, however, is not optimal either, since as P decreases, the dispersion proportionally decreases.

An evolution of PP is *Packet Train (PT)* [10], which employs more than two back-to-back packets. It calculates a dispersion rate by averaging the dispersion between each two consecutive packets of the train. As a result, this method is more robust to random noise caused by cross-traffic. Moreover, it can be used to discover *multi-channeled links* [23].

B.3 Mixed Techniques

Several methods exist that employ a combination of OP and PP techniques in order to get more efficient estimations, although they sometimes have a more complicated implementation. They are used to estimate the capacity of individual hops, so they cannot be compared with PP techniques. The main methods are *Packet Quartets* and *Packet Tailgating*.

[4]Operating System

B.3.1 Packet Quartets

Packet Quartets (PQ) [37] can be considered as an evolution of AccSig, described in Section B.1. The method consists of sending two well-separated PPs to avoid the same queue, typically more than 100ms. Each PP is formed by a probe packet (small size, below 100 Bytes) and a pacesetter packet (big size, over 800 Bytes) with a limited TTL [37]. The probe and its pacesetter packet have to be close enough to share the same queue, so as to remain queued one behind the other, thanks to its difference in size, until the pacesetter is dropped out. Figure B.5 shows an schematic model of PQ, where the pacesetter of the first and second PP are dropped at node A and B respectively. The proper choice of the nodes A and B allows any hop of the path to be targeted. The capacity of a certain hop can be estimated by measuring the delay between the two consecutive probe packets, as in AccSig.

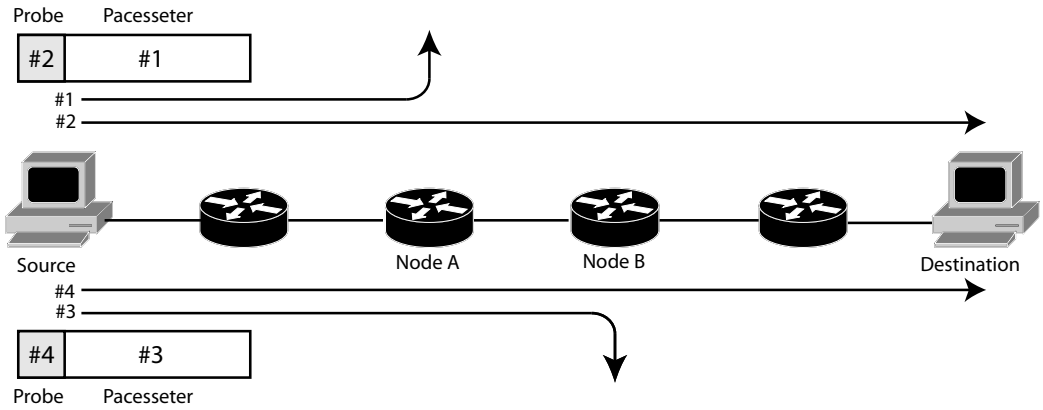

Figure B.5: Packet Quartets model. The pacesetter #1 and #3 of the first and second PP are dropped at node A and B respectively. At the destination, the delay variation between probe packets #2 and #4 is measured.

There are different implementations of PQ depending on the sizes of the probing packet and its pacesetter, and on the relation between the dropping nodes. The main advantage is that the analysis, unlike in VPS and AccSig, is done at the destination. Therefore, it does not need any kind of response reducing the noise due to round-trip and also erasing asymmetric routing problems. It can also estimate layer-2 segments by combining two of those implementations.

B.3.2 Packet Tailgating

Packet Tailgating [39] defines a new deterministic multi-packet model that is more efficient than the models described before since it attempts to unify OP

and PP models using an order of magnitude fewer packets to achieve similar results.

It is divided into two phases. In the first phase, the VPS technique is performed for the whole path (using TTL=H) to obtain a value for β_H and the latency of the path. In the second phase, the aim is to obtain the values of the SORTT for each node. Instead of performing VPS technique directly, it sends a PP composed of a 1500 Bytes packet followed by a 40 Bytes packet. The first packet is set to drop at the targeted hop by setting the TTL field to expire at that hop, so the other packet can continue without queuing to the destination due to its lower transmission delay. All the H hops of the path are targeted with the TTL to obtain all the β_h. With these values, the capacity per hop C_h is calculated from Equation B.3.

The main advantage of this method is that is very fast and efficient compared to techniques described previously since it performs the linear regression only once and uses less probing traffic. Moreover, it uses *TCP FIN* and *TCP RST* messages [11] instead of relying on ICMP responses. Despite all these advantages, this technique is not as accurate as the methods described before [39].

Appendix C

Cross-traffic Models

The goal of this appendix is to describe the different traffic models used along the simulations carried out in this report. It provides an overview of the statistical foundations of each model focusing on how to control the average cross-traffic rate. At the end of the appendix, Figures C.7 and C.8 show a simulation of the random cross-traffic models under the same network conditions and different time-scales for a low and high ABw ranges respectively.

C.1 Constant Bit Rate Traffic Model

The Constant Bit Rate (CBR) traffic model is deterministic. It is based on sending equally sized packets at a fixed rate R_x, so the inter-arrival time T_x remains constant (see Figure C.1). The cross-traffic rate used to control the ABw is

$$R_x = \frac{P_x}{T_x} \tag{C.1}$$

where P_x is the cross-traffic packet size.

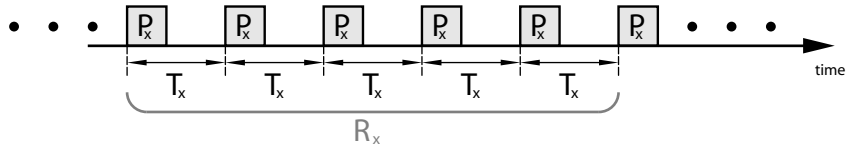

Figure C.1: CBR traffic generation.

Due to its deterministic characteristics and simplicity, CBR traffic is used as the best example to the fluid cross-traffic model initially assumed in most of the ABwE techniques (see Section 2.4).

C.2 PSD-CBR Traffic Model

The Packet Size Distribution CBR (PSD-CBR) traffic model is based on CBR traffic, but making use of a random packet size distribution obtained from a study of Internet traffic characteristics [25]. As Figure C.2 shows, the Cumulative Density Function (CDF) follows the next equation

$$\hat{P}_x = \begin{cases} 40 & p \leq 0.50 \\ 5120p - 2520 & 0.50 < p \leq 0.60 \\ 552 & 0.60 < p \leq 0.70 \\ 576 & 0.70 < p \leq 0.80 \\ 18480p - 14208 & 0.80 < p \leq 0.85 \\ 1500 & p > 0.85 \end{cases} \tag{C.2}$$

where $\hat{P}_x$ is in Bytes and $p \in [0, 1]$ is the probability of the random distribution. The chosen packet size is $P_x = round\{\hat{P}_x\}$, where $round\{x\}$ rounds x to the nearest integer.

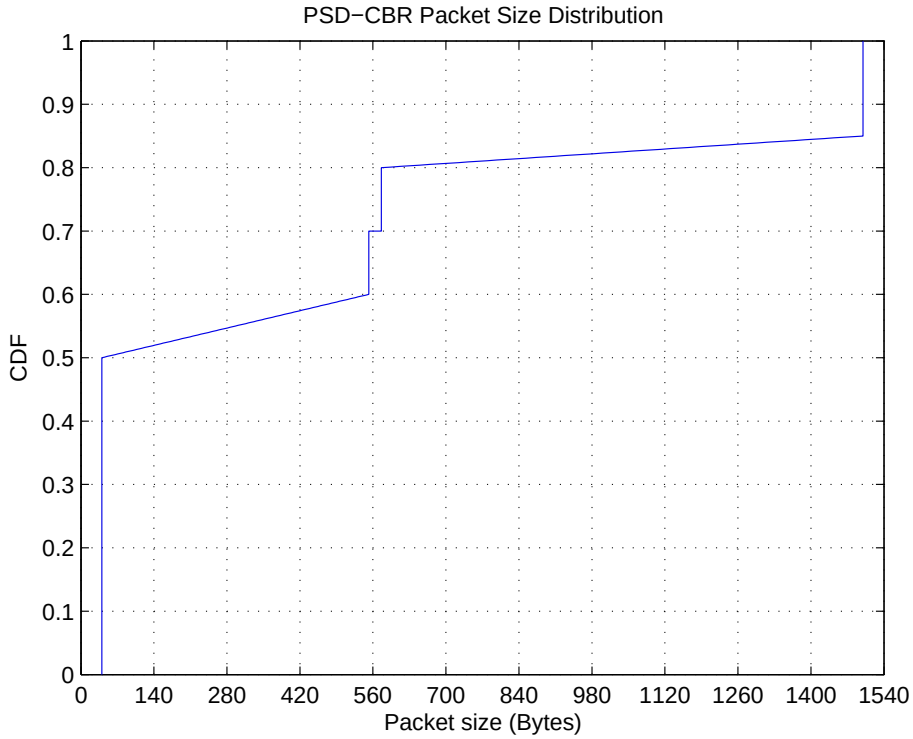

Figure C.2: PSD-CBR packet size distribution.

Due to the fact that the gap between packets T_x remains constant, the

instantaneous rate changes (see Figure C.3). The cross-traffic rate used to control the ABw is calculated as an average,

$$R_x = \frac{E[\hat{P}_x]}{T_x} \tag{C.3}$$

where the expected packet size $E[\hat{P}_x]$ is 439 Bytes.

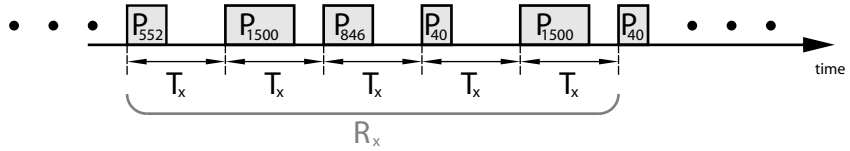

Figure C.3: PSD-CBR traffic generation.

C.3 Poisson Traffic Model

The Poisson traffic model consists of sending equally sized packets with an exponential gap between them. This traffic description is present in many real situations and it is essential in queuing theory.

C.3.1 Poisson Distribution

A *Poisson distribution* [40] is a discrete random distribution defined by the following Probability Density Function (PDF)

$$f(n,t) = \frac{e^{-\lambda t}(\lambda t)^n}{!n} \qquad n \in \mathbb{Z} \tag{C.4}$$

where $\lambda > 0$ is the expected number of occurrences per unit time during a given interval t. A random variable associated with a Poisson distribution is the average time gap T_x between two consecutive occurrences. This situation can be viewed as the probability of at least one occurrence during this interval. As Figure C.4(a) shows, the CDF of this random process is

$$F(t_x) = 1 - f(0,t_x) = 1 - e^{-\lambda t_x} \tag{C.5}$$

Equation C.5 represents the CDF of an exponential distribution [40], whose random variable T_x is continuous, and hence the PDF is

$$f(t_x) = \frac{\partial F(t_x)}{\partial t_x} = \lambda e^{-\lambda t_x} \tag{C.6}$$

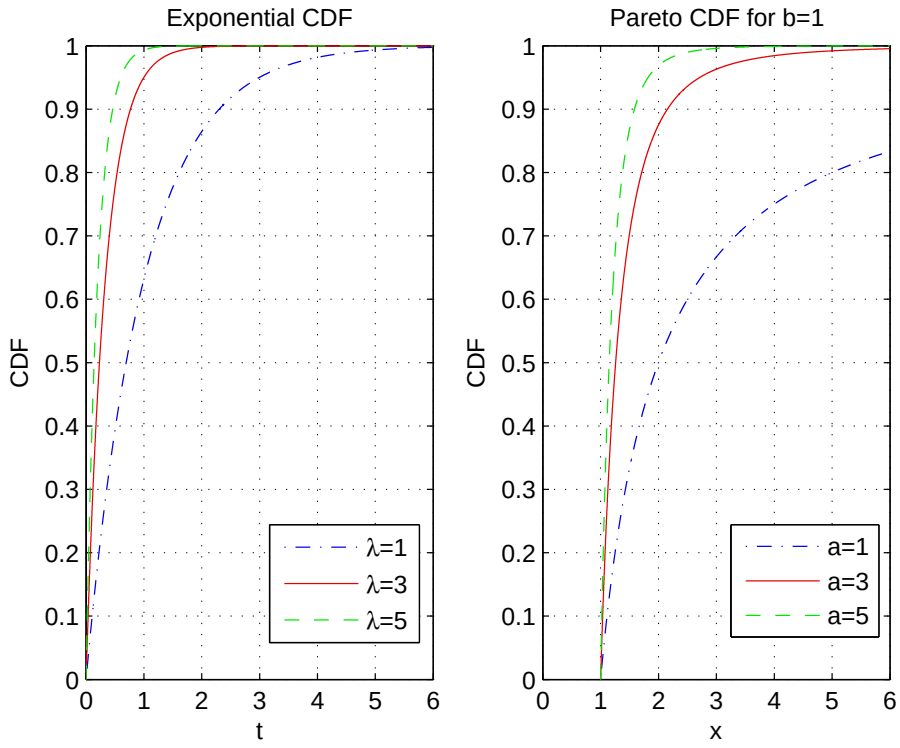

Figure C.4: Exponential (a) and Pareto (b) CDFs.

The expected value is calculated as follows,

$$E[T_x] = \int_0^\infty t_x f(t_x) dt_x = \frac{1}{\lambda} \tag{C.7}$$

C.3.2 Poisson Traffic Generator

The traffic generator is based on a Poisson source with exponential inter-arrivals T_x as Figure C.5 points out. The average cross-traffic rate, which is used to control de ABw, is obtained using the next equation

$$R_x = \frac{P_x}{E[T_x]} = \lambda P_x \tag{C.8}$$

where P_x is the cross-traffic packet size.

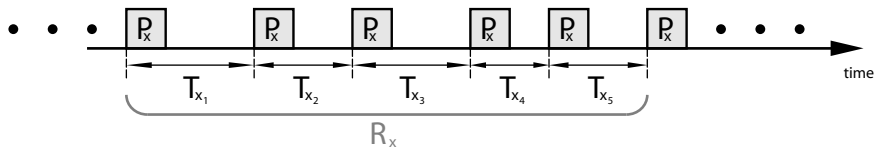

Figure C.5: Poisson traffic generation.

C.4 Pareto ON/OFF Traffic Model

The Pareto ON/OFF traffic model consists of sending packets at a fixed rate only during the ON periods, whereas the OFF periods are idle. This kind of traffic can be used to generate self-similarity, which is very useful to simulate real cross-traffic as it is explained later.

C.4.1 Pareto Distribution

A *Pareto distribution* [41] is a continuous random distribution defined by the following PDF

$$f(x) = \frac{ab^a}{x^{a+1}} \qquad \text{if } x \geq b \tag{C.9}$$

where a is the shape parameter and b is the scale parameter. Figure C.4(b) shows the Pareto CDF,

$$F(x) = \int_b^x f(x')dx' = 1 - \left(\frac{b}{x}\right)^a \qquad \text{if } x \geq b \tag{C.10}$$

If X is a random variable that follows a Pareto distribution, the expected value is

$$E[X] = \int_b^\infty xf(x)dx = \frac{ab}{a-1} \tag{C.11}$$

C.4.2 Pareto ON/OFF Traffic Generator

The traffic generator is based on two Pareto variables. As Figure C.6 illustrates, one determines the expected burst length $E[B_L]$ in number of packets, whereas the other fixes the expected OFF period $E[T_{off}]$. For a given shape parameter a and using Equation C.11, the scale parameter of each distribution is obtained as

$$b_1 = \frac{a-1}{a}E[B_L] \tag{C.12}$$

$$b_2 = \frac{a-1}{a}E[T_{off}] \tag{C.13}$$

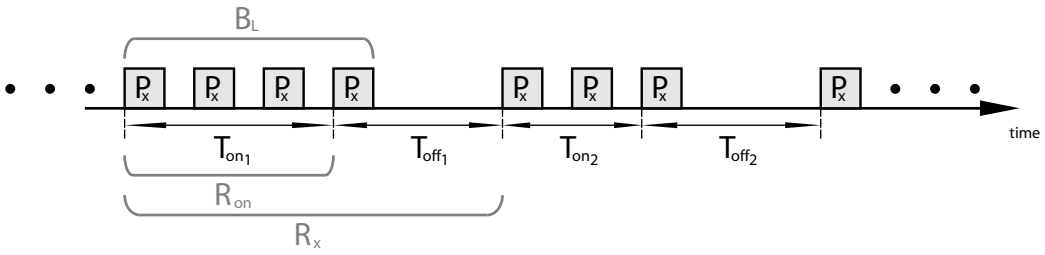

Figure C.6: Pareto traffic generation

The expected burst length is set by the input rate R_{on} during the expected ON period $E[T_{on}]$ and the packet size P_x, using the next equation

$$E[B_L] = round\left\{\frac{R_{on}}{P_x}E[T_{on}]\right\} \tag{C.14}$$

An approximation of the average cross-traffic, which is used to control the ABw, is calculated as

$$R_x = \frac{E[T_{on}]}{E[T_{on}] + E[T_{off}]}R_{on} \tag{C.15}$$

As Figures C.7(c) and C.8(c) point out, the real average cross-traffic rate varies too much from the approximated average obtained from Equation C.15 even under large time-scales. This implies that such approximation cannot be used as a reference of the average cross-traffic rate over the whole simulation time, so it is necessary to calculate the average cross-traffic in real time. As an example, this means that it would not make sense to evaluate a method that attempts to estimate the average ABw of a path comparing its results with the rate obtained by Equation C.15 since it differs from reality.

C.4.3 Self-Similarity

Traditionally, data traffic has been modeled using the background acquired from the study of voice traffic, which has been accurately described by a Poisson process. However, recent studies have revealed that Internet traffic exhibits self-similarity, burstiness and Long-Range Dependency (LRD) [14]. On the one hand, a self-similar process shows a fractal structure, i.e. short-time patterns are equal to long-time patterns. On the other hand, LRD implies statistically significant correlations across large time-scales [42] against memoryless Poisson processes. So, modeling self-similar traffic is not an easy task. In [43], it is suggested to use multiple Pareto aggregation sources with a shape parameter $a < 2$ so as to synthesize self-similar cross-traffic.

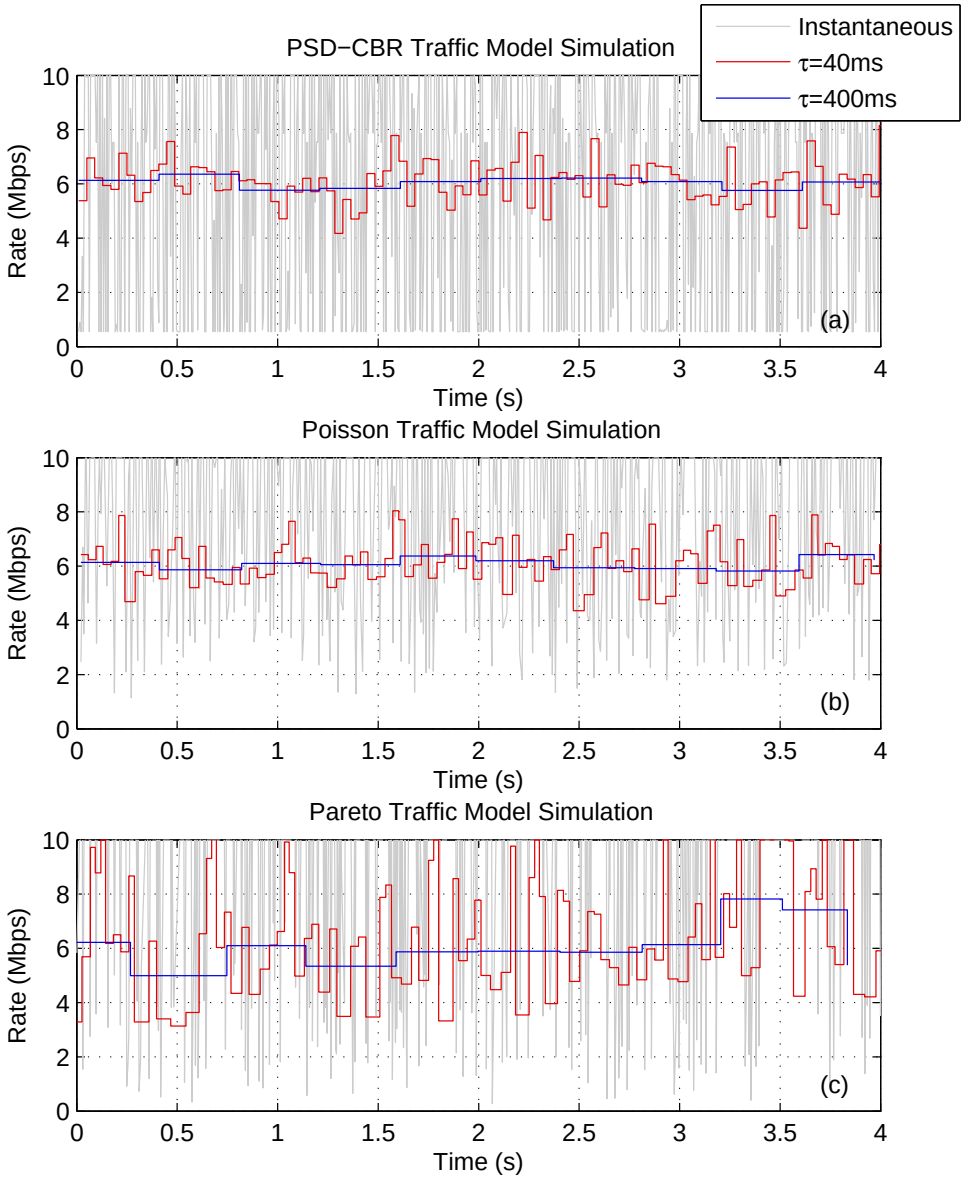

Figure C.7: Simulation traces under PSD-CBR (a), Poisson (b) and Pareto with $a = 1.9$, $E[T_{on}] = 1ms$ and $E[T_{off}] = 3ms$ (c) for different time-scales. The network consists of a single-hop with 10Mbps of capacity, where the average cross-traffic rate is set to 6Mpbs. The cross-traffic packet size in Pareto and Poisson is set to 552 Bytes.

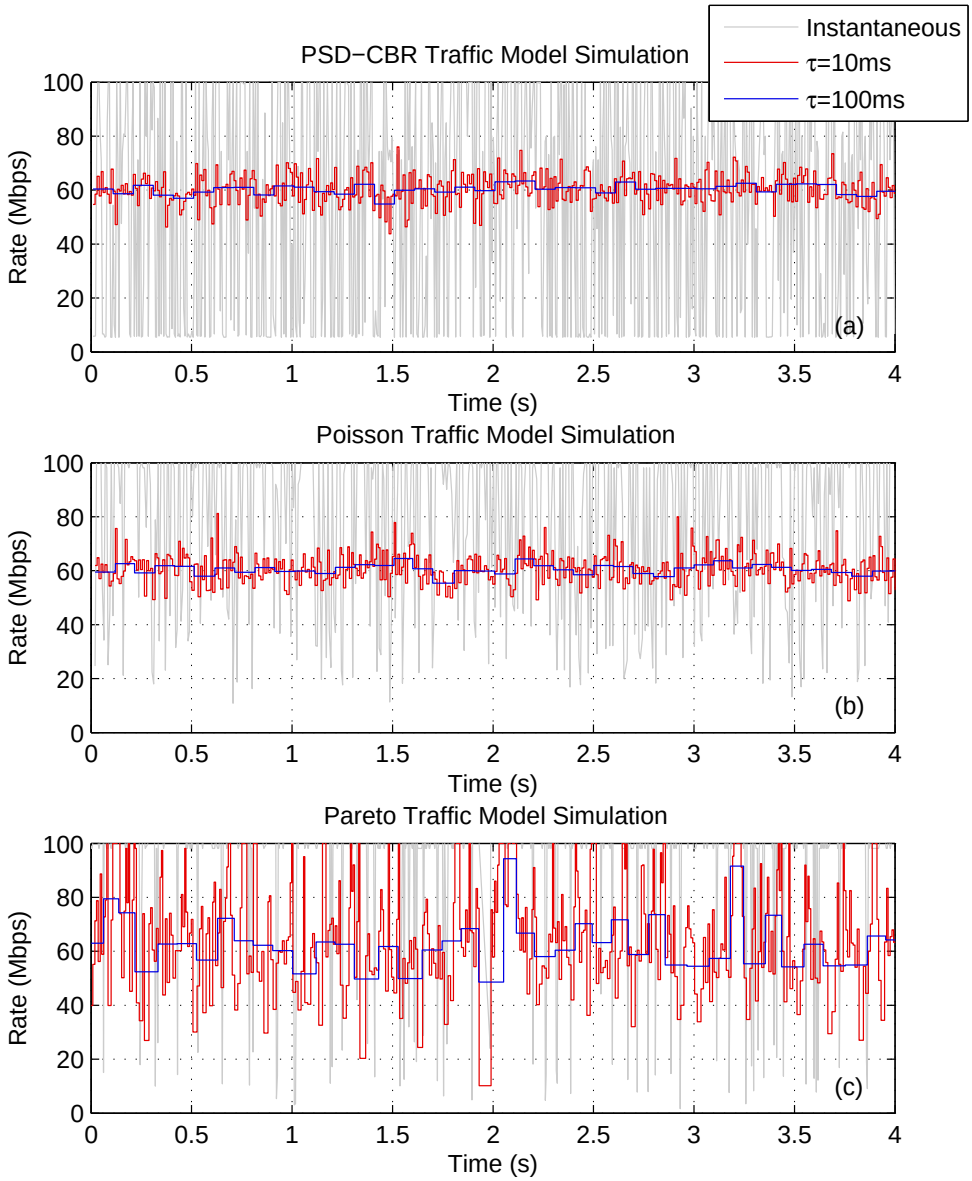

Figure C.8: Simulation traces under PSD-CBR (a), Poisson (b) and Pareto with $a = 1.9$, $E[T_{on}] = 0.25ms$ and $E[T_{off}] = 0.75ms$ (c) with different time-scales. The network consists of a single-hop with 100Mbps of capacity, where the average cross-traffic rate is set to 60Mpbs. The cross-traffic packet size in Pareto and Poisson is set to 552 Bytes.

Appendix D

Turning Point Estimation

As Figure D.1(a) illustrates, suppose a n-long data series (x_i, y_i) that shows a linear relationship as

$$\begin{cases} y = a_1 x + b & x < x_0 \\ y = a_2(x - x_0) + a_1 x_0 + b & x \geq x_0 \end{cases} \tag{D.1}$$

where a_1 and a_2 are the slopes of each line respectively, (x_0, y_0) is the turning point, and b is the offset. If the turning point was known, it would only be necessary to apply a method such as *least squares fitting* [44] in both parts of the data to estimate each segmented line. However, when the turning point is an unknown value, other techniques have to be used to determine it first. Two of them are *Minimum Variance* and *Maximum Second Derivative*, which are explained in the following sections.

D.1 Minimum Variance

This method is based on least squares fitting and it can simultaneously determine, not only the turning point, but also both slopes [19] by measuring the variance of the fitting line. Let first assume an initial point at (0,0). To keep this assumption, it is only necessary to translate the axes properly. Therefore, the new segmented lines are

$$\begin{cases} \bar{y} = a_1 x & x < x_0 \\ \bar{y} = a_2(x - x_0) + a_1 x_0 & x \geq x_0 \end{cases} \tag{D.2}$$

In order to determine the two slopes in terms of the cloud of points and the unknown turning point, it is necessary to apply the least squares fitting

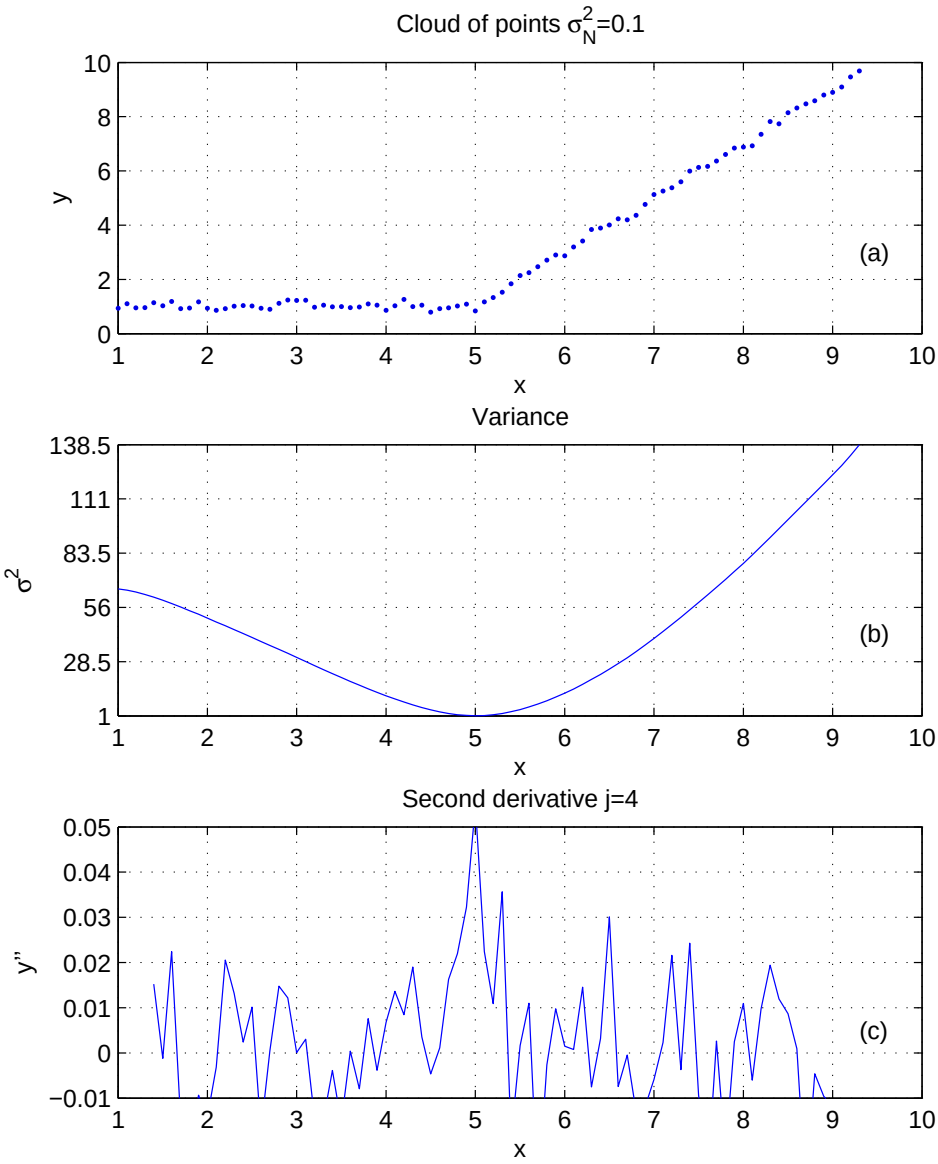

Figure D.1: (a) is a cloud of points with gaussian noise, (b) is a turning point estimation using the minimum variance method and (c) using the maximum second derivative method.

to each slope in Equation D.2,

$$\frac{\partial}{\partial a_1}\sum_{i=1}^{n}(y_i-\hat{y})^2=0 \qquad \frac{\partial}{\partial a_2}\sum_{i=1}^{n}(y_i-\hat{y})^2=0 \tag{D.3}$$

where $\hat{y}$ is the estimated value. If the estimated value for each line is $\hat{y}=a_1x_i$ and $\hat{y}=a_2(x_i-x_0)+a_1x_0$ respectively. Therefore, Equation D.3 can be rewritten as follows

$$\frac{\partial}{\partial a_1}\left(\sum_{i=1}^{k}(y_i-a_1x_i)^2+\sum_{i=k+1}^{n}(y_i-a_2x_i+a_2x_0+a_1x_0)^2\right) = 0 \tag{D.4}$$

$$\frac{\partial}{\partial a_2}\left(\sum_{i=1}^{k}(y_i-a_1x_i)^2+\sum_{i=k+1}^{n}(y_i-a_2x_i+a_2x_0+a_1x_0)^2\right) = 0 \tag{D.5}$$

where k is the last point of the cloud that belongs to the first line. By solving and reordering Equations D.4 and D.5, the following expressions are obtained

$$\sum_{i=1}^{k}\left(y_ix_i-a_1x_i^2\right)+x_0\sum_{i=k+1}^{n}(y_i-a_2x_i)+ka_2x_0^2-ka_1x_0^2 = 0 \tag{D.6}$$

$$\sum_{i=k+1}^{n}\left(y_ix_i-a_2x_i^2+2a_2x_0x_i-a_1x_0x_i-a_2x_0^2+a_1x_0^2\right) = 0 \tag{D.7}$$

In order to simplify Equations D.6 and D.7, the next variables are defined

$$s_{x1}=\sum_{i=1}^{k}x_i \qquad s_{xx1}=\sum_{i=1}^{k}x_i^2 \qquad s_{x2}=\sum_{i=k+1}^{n}x_i \quad s_{xx2}=\sum_{i=k+1}^{n}x_i^2$$

$$s_{y1}=\sum_{i=1}^{k}y_i \qquad s_{yy1}=\sum_{i=1}^{k}y_i^2 \qquad s_{y2}=\sum_{i=k+1}^{n}y_i \quad s_{yy2}=\sum_{i=k+1}^{n}y_i^2$$

$$s_{xy1}=\sum_{i=1}^{k}x_iy_i \quad s_{xy2}=\sum_{i=k+1}^{n}x_iy_i$$

By substituting these variables, Equations D.6 and D.7 are rewritten as

$$\begin{cases} a_1(s_{xx1}+kx_0^2)+a_2x_0(s_{x2}-kx_0)=s_{xy1}+x_0s_{y2} \\ a_1x_0(s_{x2}-kx_0)+a_2(s_{xx2}-x_0(2s_{x2}-kx_0))=s_{xy2}-x_0s_{y2} \end{cases} \tag{D.8}$$

By solving Equation D.8, an estimator for both slopes can be found

$$a_1 = \frac{(s_{xx2}-2x_0s_{x2}+kx_0^2)(s_{xy1}+x_0s_{y2})}{(s_{xx1}+kx_0^2)(s_{xx2}-2x_0s_{x2}+kx_0^2)-x_0^2(s_{x2}-kx_0)^2} - \\ -\frac{x_0(s_{x2}-kx_0)(s_{xy2}-x_0s_{y2})}{(s_{xx1}+kx_0^2)(s_{xx2}-2x_0s_{x2}+kx_0^2)-x_0^2(s_{x2}-kx_0)^2} \tag{D.9}$$

$$a_2 = \frac{(s_{xx1}+kx_0^2)(s_{xy2}-x_0s_{y2})-x_0(s_{x2}-kx_0)(s_{xy1}+x_1s_{y2})}{(s_{xx1}+kx_0^2)(s_{xx2}-2x_0s_{x2}+kx_0^2)-x_0^2(s_{x2}-kx_0)^2} \tag{D.10}$$

In order to determine when the slope changes, the variance σ^2 of the estimation is calculated for each point of the cloud

$$\sigma_k^2 = \sum_{i=1}^{k} (y_i - \bar{y})^2 \qquad k = 2, \ldots, n \tag{D.11}$$

By substituting each estimator, Equation D.11 is expressed as

$$\sigma_k^2 = \sum_{i=1}^{k} (y_i - a_1 x_i)^2 + \sum_{i=k+1}^{n} (y_i - a_2 x_i + a_2 x_0 + a_1 x_0)^2 \tag{D.12}$$

By using Equation D.8, substituting and simplifying, Equation D.12 is reduced to

$$\sigma_k^2 = s_{yy1} + s_{yy2} - a_1 s_{xy1} - a_2 s_{xy2} + (a_2 - a_1) x_0 s_{y2} \tag{D.13}$$

As Figure D.1(b) points out, the variance is calculated iteratively using Equations D.9, D.10 and D.13. The point for which the variance is minimum is considered as the turning point.

D.2 Maximum Second Derivative

The basics of this method is that a local maximum of the second derivative of a function can determine an inflection point, and so, a change in the slope [18]. Let define the second derivative [45] of a function $f(x)$ as

$$f''(x) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{D.14}$$

If the data series follows Equation D.1, then the second derivative given by Equation D.14 can be rewritten as

$$y_i'' \approx \frac{y_{i+j} - 2y_i + y_{i-j}}{(x_{i+j} - x_i)(x_i - x_{i-j})} \qquad i = 1 + j, ..., n - j \tag{D.15}$$

where j is equivalent to the infinitesimal increase h in Equation D.14. The larger the j, the more the fluctuations are reduced, but the worse the resolution. As Figure D.1(c) illustrates, the point for which the second derivative shows a maximum, is reported as the turning point. Then, it is necessary to use a linear regression method, such as least squares fitting, to get the slopes of each line.

D.3 Summary

Although *Maximum Second Derivative* method has an easier implementation, it is more sensitive to noise than *Minimum Variance* method (see Figure D.1) due to the statistical analysis of the latter. This fact can be mitigated by increasing j, but then the resolution is reduced. Therefore, *Minimum Variance* method is more suitable in a highly noisy environment or if the data set is formed by a few points.

Appendix E

Source Code

The purpose of this appendix is to give practical implementations for the different algorithms used by the studied techniques, as a complement to the descriptions and explanations of the different chapters, and as a help for the understanding of all the studied methodologies.

E.1 PathChirp Algorithms

The following section gives a practical implementation of pathChirp algorithms in language C from the proposal in [6].

E.1.1 Determining an Excursion

The next source code determines an excursion, returning the index to set the suitable rate to each packet, where q is the OWD of a single stream, k is the index of studied OWD, K is the number of packets per stream, F is the decrease factor and L is the excursion length threshold.

```
int excursion (float q[],int k,int K, float F, int L)
{
  int j=k+1;
  float max_q=0;

  //Calculus of the excursion length
  while((j<K) && (F*(q[j]-q[k])>max_q))
  {
    if(max_q<(q[j]-q[k])) max_q=q[j]-q[k];
    j=j+1;
  }
```

```
  //Determining whether it is an excursion or not
  if(j>=K-1) return j;
  else if((j-k)>=L) return j;
  else return k;
}
```

E.1.2 Estimation of the Available Bandwidth

The next source code determines the most suitable rate to each sample, according to the study of the excursions, and it calculates the ABwE, where tSS is the timestamps at the source, tSD is the timestamps at the destination, $pSize$ is the packet size, K is the number of packets per stream, M is the number of streams per fleet, F is the decrease factor and L is the excursion length threshold.

```
float estimation (float tSS[][], float tSD[][], int pSize,
int M,int K, float F, int L)
{
  float q[M][K];          //One-Way Delay
  float R[K-1];           //Instantaneous rates
  float E[K-1];           //Estimator per packet
  float D[M];             //Estimator per stream
  float timeGapS=0;       //Stream duration
  float aBW=0;            //Estimated available bandwidth
  int m,k,j,l,s;          //Several counters

  //Calculus of OWDs and initialization of D[m]
  for(m=0;m<M;m++)
  {
    D[m]=0;
    for(k=0;k<K;k++) q[m][k]=tSD[m][k]-tSS[m][k];
  }

  for(k=0;k<K-1;k++)
  {
    timeGap[k]=tSS[0][k+1]-tSS[0][k];
    //Calculus of the instantaneous rates
    R[k]=8*pSize/timeGap[k];
    //Calculus of the stream duration
    timeGapS=timeGapS+timeGap[k];
  }
```

```
//Calculus of estimate per packet in 1st and 2nd cases
for(m=0;m<M;m++)
{
  l=K-2;
  // Initialization of E[k]
  for(k=0;k<K-1;k++) E[k]=0;
  //Current OWD of the stream
  k=0;
  while(k<K-1)
  {
    //If there is an increasing trend
    if(q[m][k]<q[m][k+1])
    {
      //Determining the end of the excursion
      j=excursion(q[m],k,K,F,L);
      //First case
      if(j>k && j<K)
      {
        for(s=k;s<j;s++)
        {
          if(q[m][s]<q[m][s+1]) E[s]=R[s];
        }
      }
      // Second case
      else if(j==K)
      {
        for(s=k;s<K-1;s++) E[s]=R[k];
        l=k;
      }
      if(j==k) j=j+1;
      k=j;
    }
    //Otherwise it selects the next OWD of the stream
    else k=k+1;
  }

  //Calculus of the estimate per stream
  for(k=0;k<K-1;k++)
  {
    //Calculus of the estimate per packet in 3rd case
    if(E[k]==0) D[m]=D[m]+R[l]*timeGap[k];
    else D[m]=D[m]+E[k]*timeGap[k];
  }
```

```
    D[m]=D[m]/timeGapS;
    aBW=aBW+D[m]/M;
  }
  //Available Bandwidth Estimation
  return aBW;
}
```

E.2 SLoPS Algorithms

The following section gives a practical implementation of SLoPS algorithms in language C from the proposal in [21].

E.2.1 Rate Adjustment Algorithm

The next source code gives an implementation of the rate adjustment algorithm taking into account gray-regions, where $RinOLD$ is the current input rate, T is the detected trend, $Rmin$ is the minimum input rate, $Rmax$ is the maximum input rate, w is the desired resolution and X is the gray-region resolution.

```
#define   INCREASING      1
#define   NONINCREASING   0
#define   GRAYREGION     -1

float rateAdjustment (float RinOLD, int T, float Rmin,
float Rmax, float w, float X)
{
  float Rin;                  //New input rate
  float Gmin=0, Gmax=0;       //Gray-regions range
  int i=0;                    //Counter

  while(w<(Rmax-Rmin) && ((X<=abs(Gmin-Rmin)) &&
  (X<=abs(Rmax-Gmax))))
  {
    //If an increasing trend is detected
    if(T=INCREASING)
    {
      Rmax=RinOLD;
      if (Gmax>0) Rin=(Rmax+Gmax)/2;
      else Rin=(Rmax+Rmin)/2;
    }
    //If a non-increasing trend is detected
```

```
    else if(T=NONINCREASING)
    {
      Rmin=RinOLD;
      if (Gmin>0) Rin=(Rmin+Gmin)/2;
      else Rin=(Rmax+Rmin)/2;
    }
    //If a gray-region is detected
    else
    {
      if(!Gmax && !Gmin)
      {
        Gmax=RinOLD;
        Gmin=RinOLD;
      }
      if(Gmax<=RinOLD)
      {
        Gmax=RinOLD;
        Rin=(Rmax+Gmax)/2;
      }
      else if(Gmin>RinOLD)
      {
        Gmin=RinOLD;
        Rin=(Rmin+Gmin)/2;
      }
    }
    i++;
  }
  return Rin;
}
```

E.2.2 Initialization of the Input Rate Range

The next source code shows an exponential initialization of the input rate range in absence of gray-regions, where *Rmin* is the minimum input rate and *Rmax* is the maximum input rate.

```
void initialization (float* Rmin, float* Rmax) {
  float Rin=1;          //Initial input rate (Mbps)
  float RinOLD;         //Previous input rate
  float aBW;            //Available Bandwidth Estimation

  //An estimation using any algorithm, e.g. SLoPS
  aBW=estimation(Rin);
```

```
  //If the ABw is bigger than 1 (Mbps)
  if (Rin<aBW)
  {
    while(Rin<aBW)
    {
      RinOLD=Rin;
      Rin=2Rin;
      aBW=estimation(Rin);
    }
    //Measurable range
    Rmax=&Rin;
    Rmin=&RinOLD;
  }
  //Otherwise
  else
  {
    while(Rin>aBW)
    {
      RinOLD=Rin;
      Rin=Rin/2;
      aBW=estimation(Rin);
    }
    //Measurable range
    Rmax=&RinOLD;
    Rmin=&Rin;
  }
}
```

Appendix F

Simulation Results

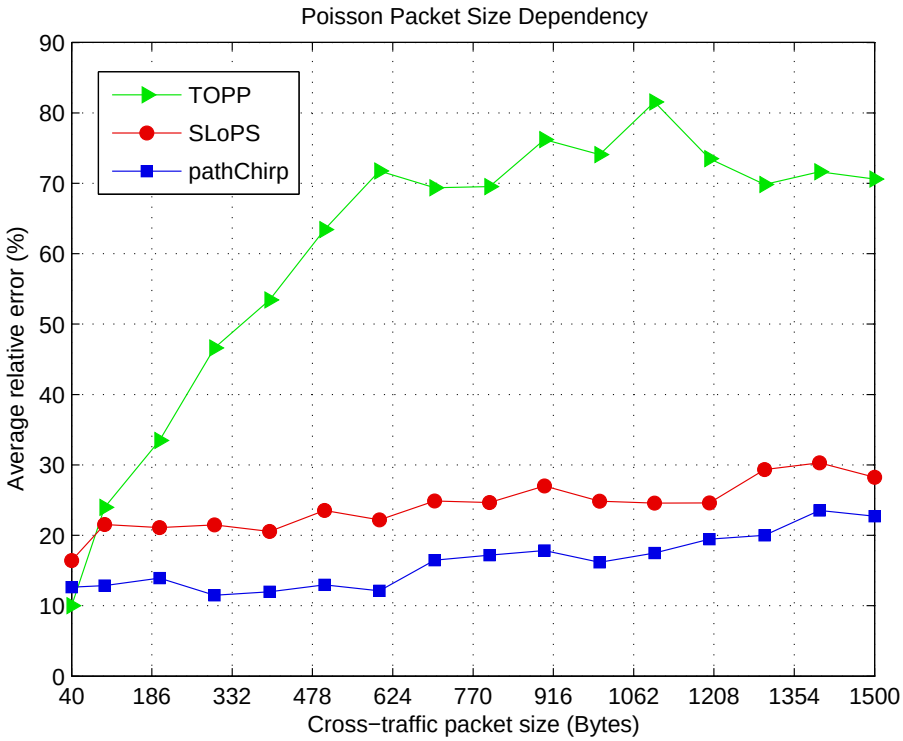

Figure F.1: Poisson packet size comparison for low range in single-hop. *TOPP* shows a great dependency on the cross-traffic packet size. However, *SLoPS* and *pathChirp* are less sensitive.

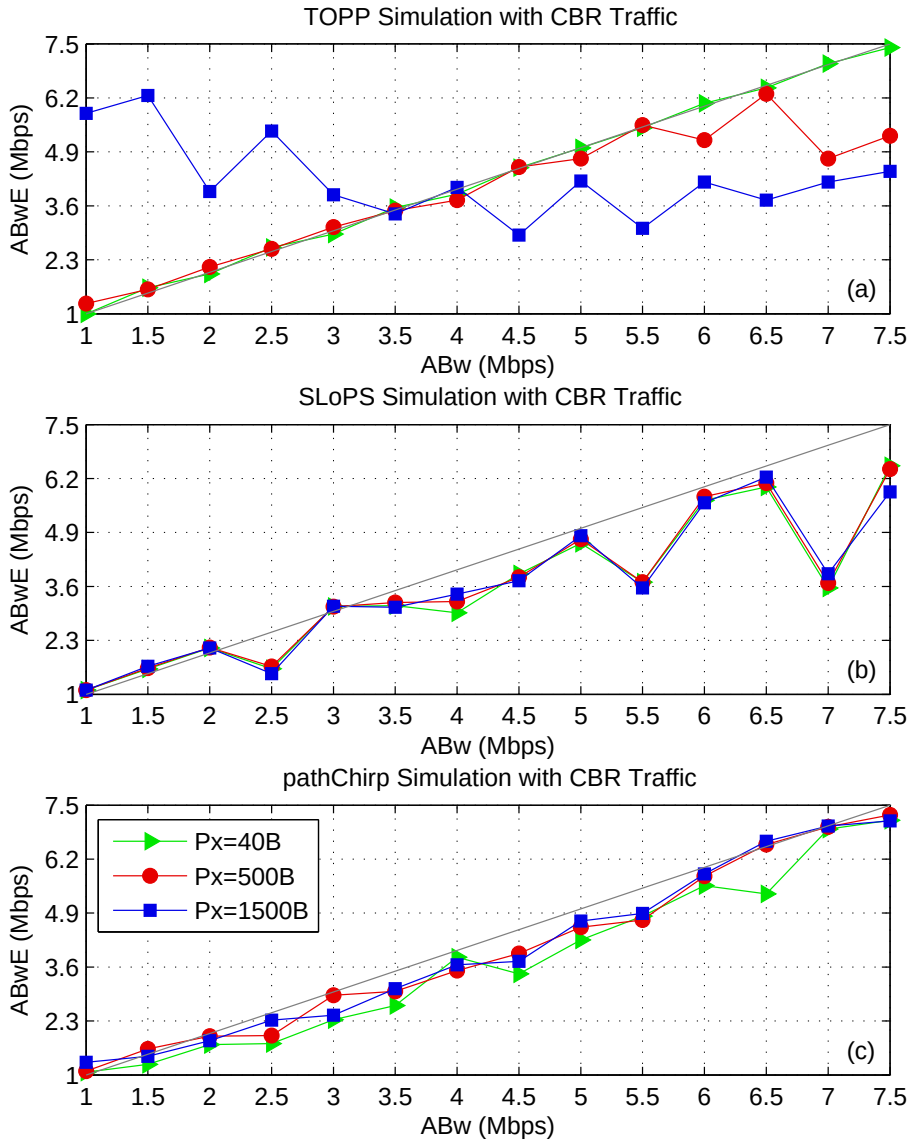

Figure F.2: CBR packet size comparison for low range in single-hop. Opposite to *SLoPS* and *pathChirp*, *TOPP* shows a great dependency on the packet size.

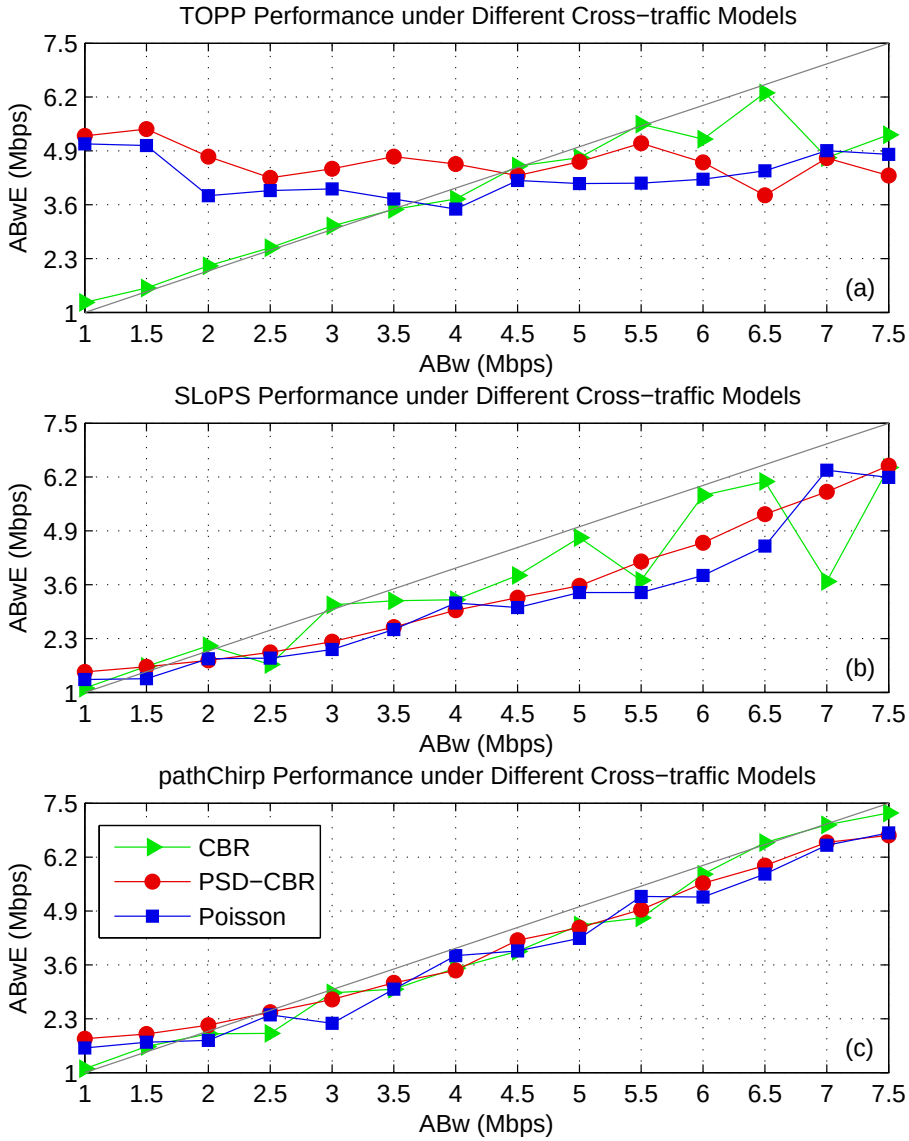

Figure F.3: Cross-traffic comparison for low range in single-hop. *TOPP* shows a great dependency on the kind of cross-traffic, whereas *SLoPS* and *pathChirp* are less sensitive showing the latter a slightly better performance.

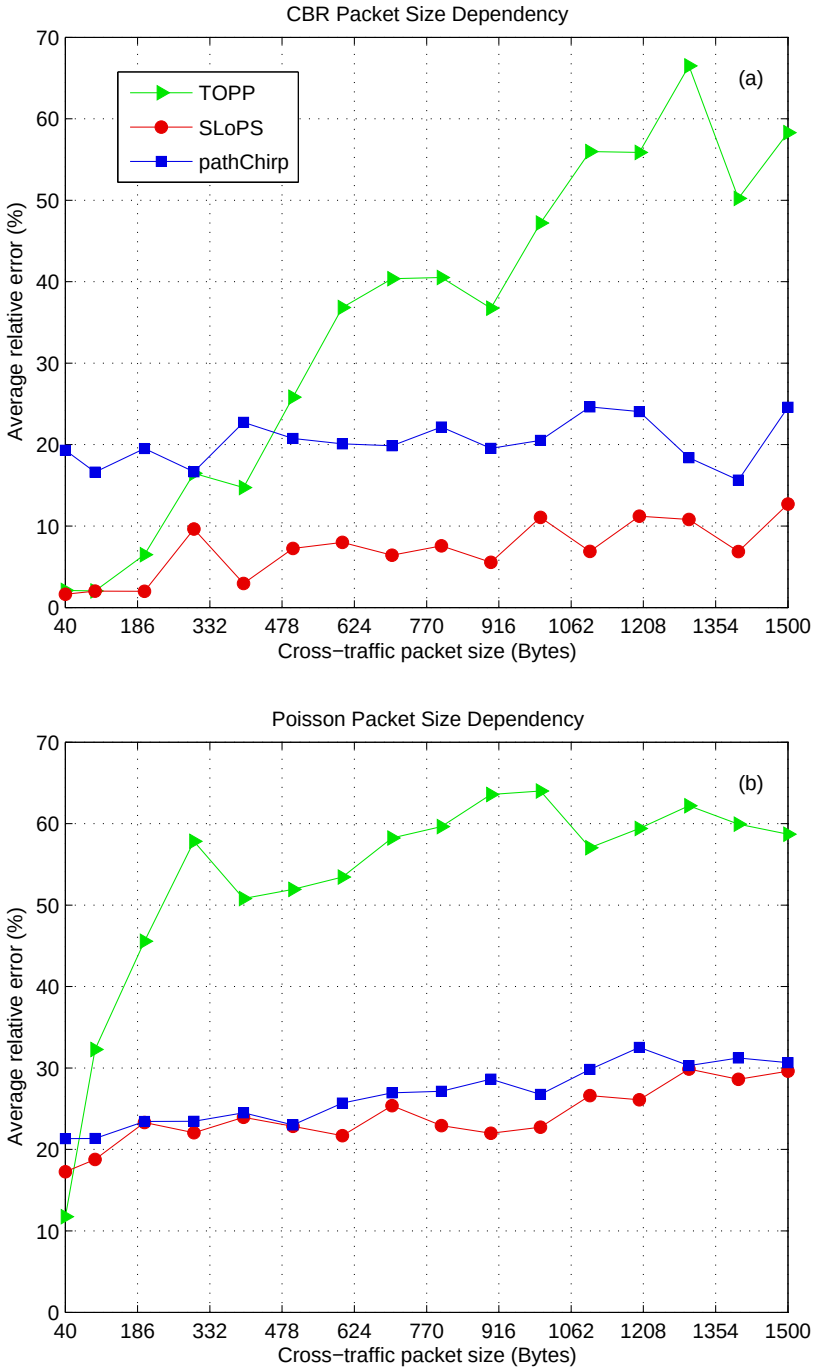

Figure F.4: CBR (a) and Poisson (b) packet size comparison for low range in multi-hop. As in single-hop, *TOPP* is totally dependent on the cross-traffic packet size, whereas *pathChirp* and *SLoPS* perform better.

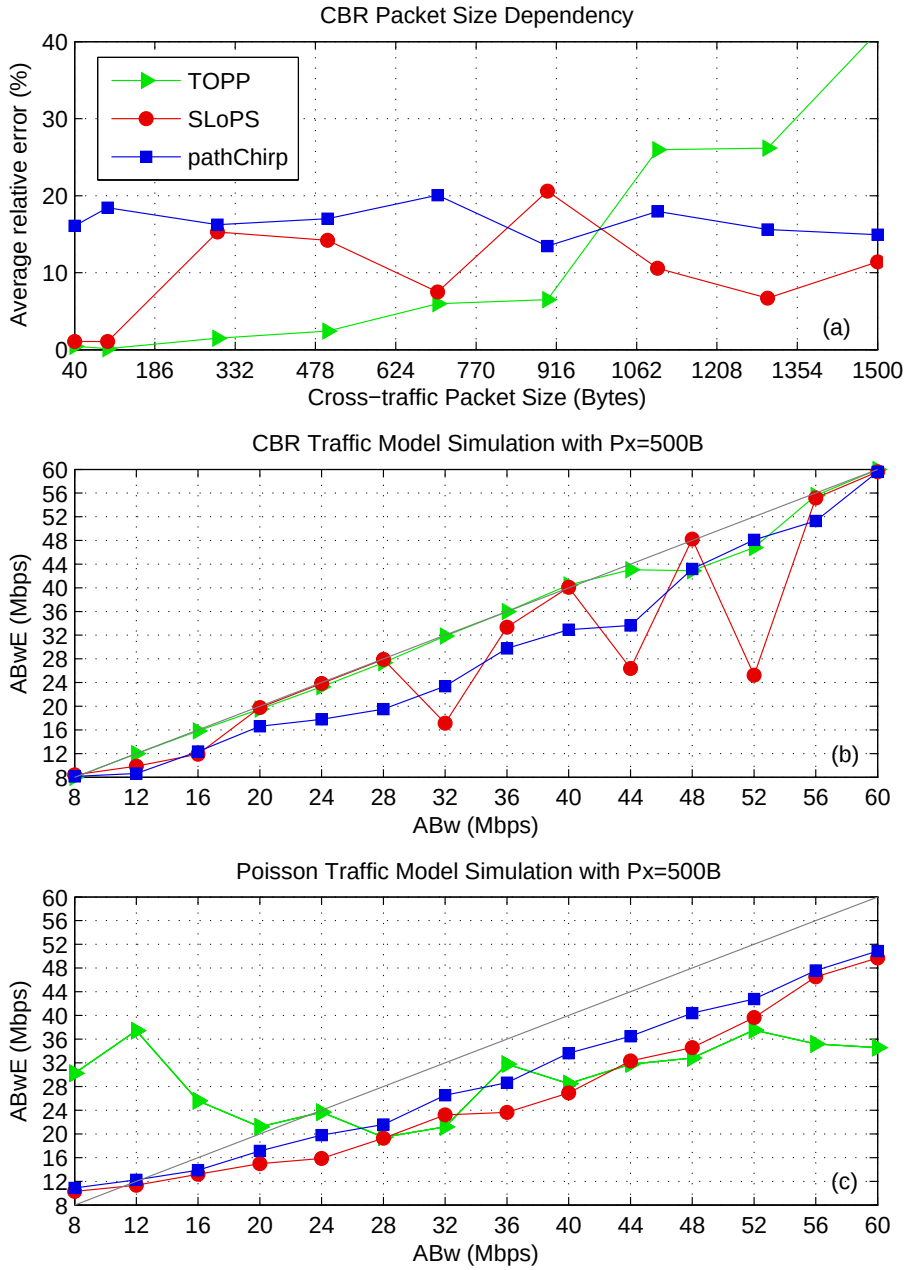

Figure F.5: CBR packet size dependency (a), and CBR (b) and Poisson (c) traffic models comparison for high range in single-hop.

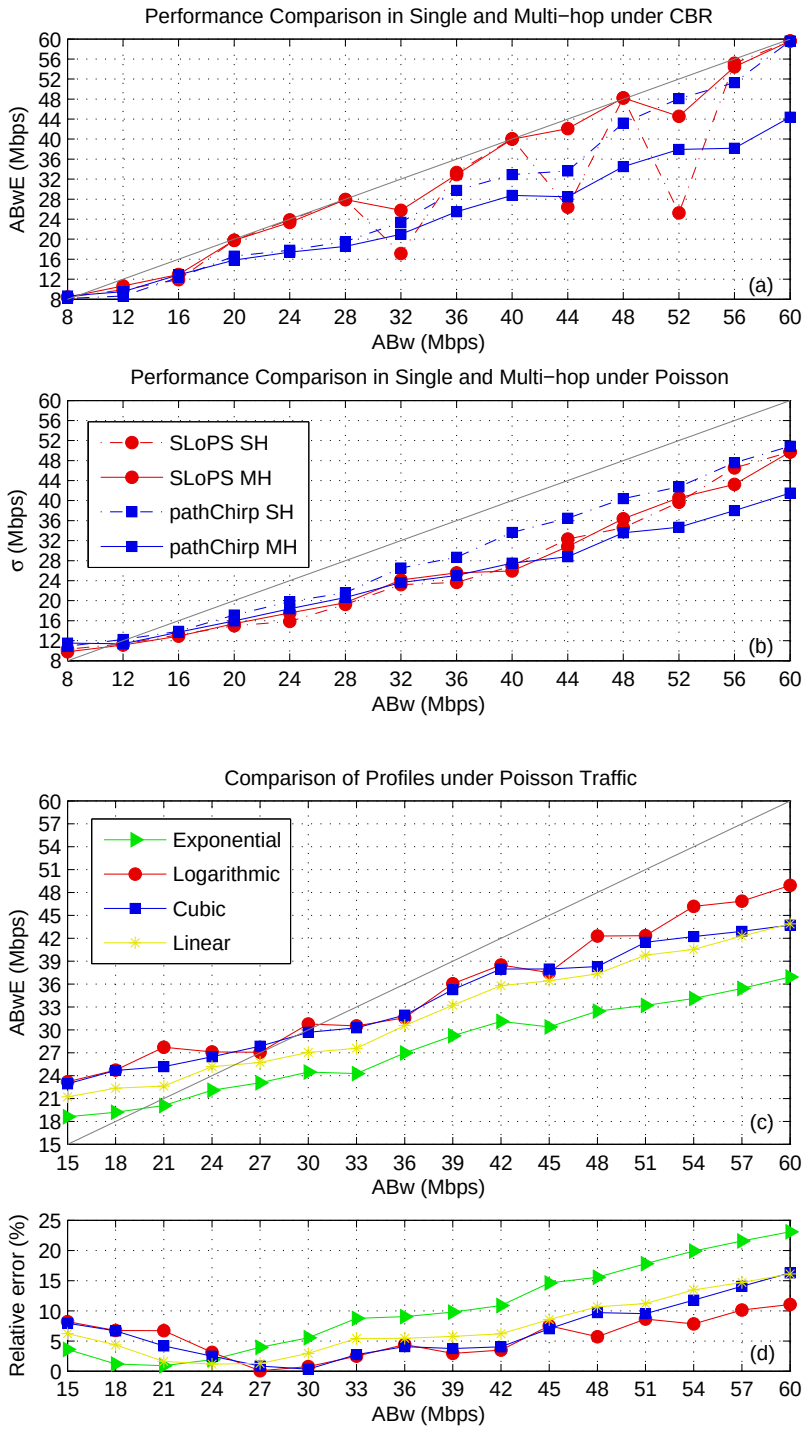

Figure F.6: Single and multi-hop comparison for high range under CBR (a) and Poisson (b) traffic with $P_x = 500B$. *PathChirp* profiles comparison under Poisson traffic, where (c) shows the ABwE and (d) the relative error.

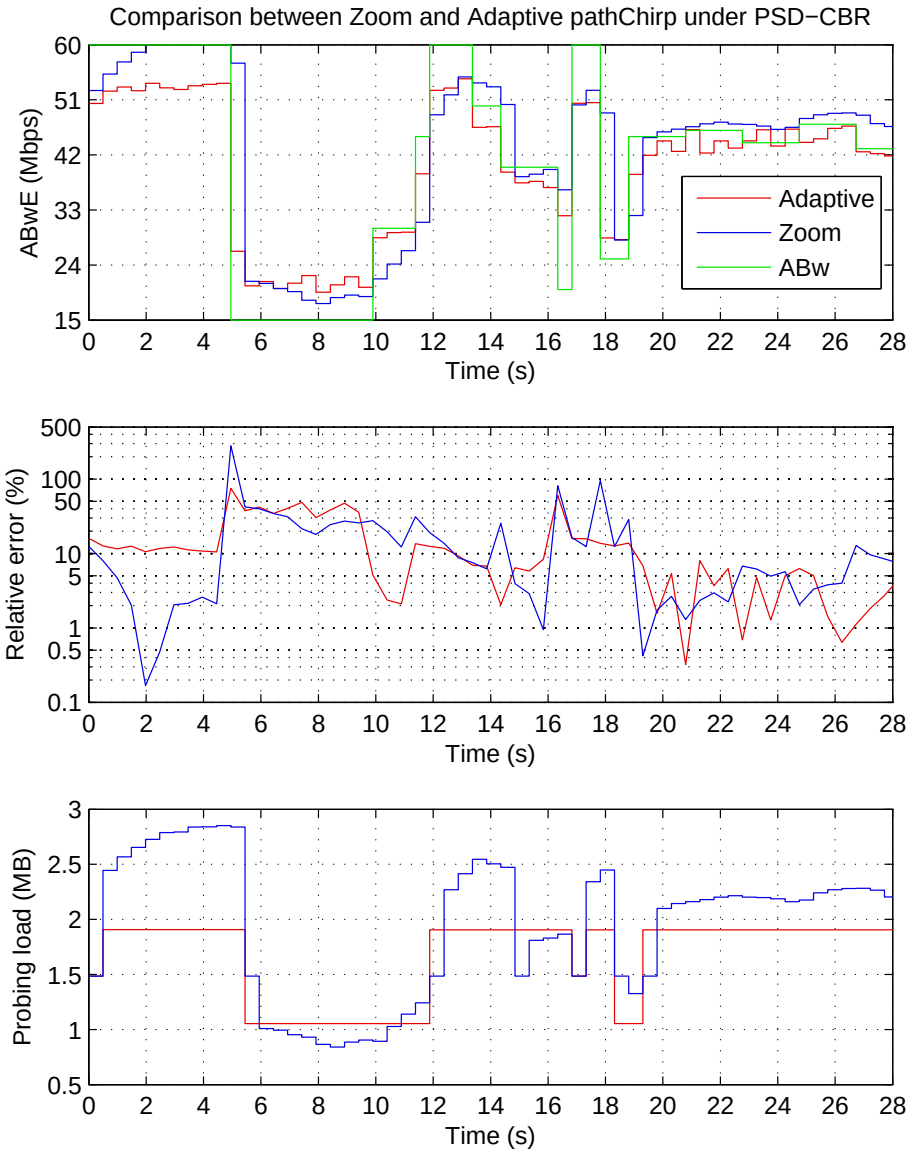

Figure F.7: *Adaptive* and *Zoom pathChirp* comparison under PSD-CBR, where (a) represents the ABwE, (b) shows the relative error and (c) plots the probing load during time.

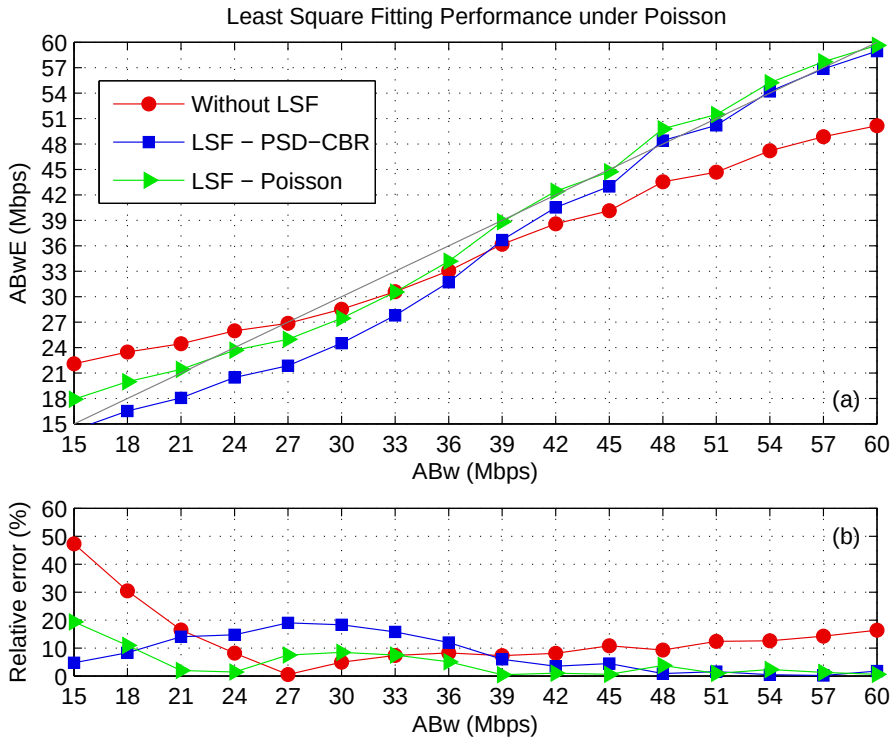

Figure F.8: Least squares fitting comparison under Poisson (a) $P_x = 500B$, where the relative error is represented in (b). Although the least squares fitting depends on the kind of cross-traffic, its use is profitable, even without the optimum parameters

Appendix G

Differentiated Services

The one-way transmission of a packet through a path can be described in terms of average rate, delay, jitter[1], and/or loss. Such characteristics, which define a *service*, can be quantitatively or statistically specified, or determined by access priority to network resources. In the *best-effort* service model, all packets equally compete for network resources. Bandwidth among users is allocated as good as possible without making any commitment as to rate or any other service quality. However, the rise in the usage of IP networks has placed great demand on bandwidth and buffer space at network devices, leading to heavy congestion. Allocate bandwidth to different users in a controlled and predictable way during network congestion is the aim of *Differentiated Services (DiffServ)* [3].

G.1 DiffServ Basics

DiffServ is based on the TCP congestion avoidance procedures in which a single dropped packet produces a sending rate adjustment at the source. The idea is to monitor the traffic of each user as it enters the network and to mark packets as either in or out of their service profiles. *Out-of-profile* packets are those packets in the traffic stream that arrive with a higher rate than recommended to guarantee service allocation for all the traffic flows, so they are preferentially dropped by routers during congestion periods.

A *Service Level Agreement (SLA)* is a service contract between a customer and a service provider that specifies the forwarding service the customer should receive. It may include a *Traffic Conditioning Agreement (TCA)*,

[1]Variation in the time between packets arriving, caused by network congestion, timing drift, or route changes.

which specifies the classifier rules, including the traffic profiles, i.e. the rules for determining whether a particular packet is in-profile or out-of-profile, and the metering, marking, dropping and/or shaping policies.

DiffServ provides service differentiation in one traffic flow direction, thus it is asymmetric. The traffic source pre-marks packets with the suitable *DiffServ CodePoint (DSCP)*[2] according to the desired priority. Once the packet reaches the DiffServ domain, it is checked whether such packet is in compliance with the traffic profile or not, being re-marked and/or dropped if necessary.

G.2 Architecture

The architecture can be divided into edge and core, illustrated in Figure G.1. On the one hand, the edge or boundary nodes interconnect the DiffServ domain to other domains. A traffic conditioner is located in an edge node of DiffServ architecture on the upstream and/or on the downstream domain. The edge node monitors conformance to the TCA, and may drop, shape, or re-mark packets as necessary. On the other hand, within the core of the network, packets are prioritized or dropped depending on the DSCP set by the traffic conditioner.

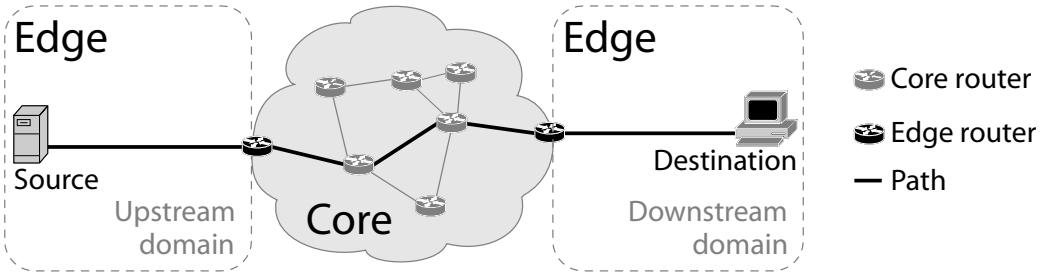

Figure G.1: DiffServ architecture.

G.3 Traffic Conditioner

The *traffic conditioner* performs metering, shaping, dropping and/or re-marking to ensure that the traffic entering the DiffServ domain conforms to the rules specified in the TCA, which vary from one DSCP to another. Figure G.2 represents the conditioner modules. In case of no traffic profile, packets may only pass through a classifier and a marker.

[2]The DSCP is included in the *Type of Service (ToS)* field of the IP header.

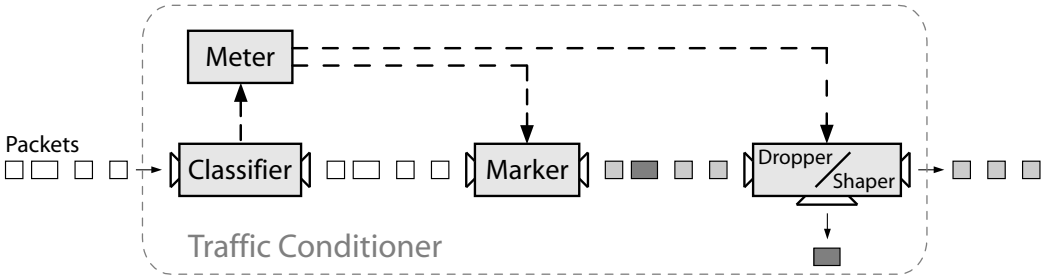

Figure G.2: DiffServ traffic conditioner.

G.3.1 Classifier

The *classifier* gets the conditioning rules for each packet by looking its DSCP in the traffic profile. It specifies the meter that have to be used for the marking and the thresholds to perform the dropping and/or the shaping.

G.3.2 Meters

Traffic meters [3] measure the temporal properties of the stream of packets selected by the classifier against the traffic profile specified in the TCA. A meter passes state information to other conditioning functions to trigger a particular action for each packet which is either in or out-of-profile. Although there are several kinds of meters, DiffServ conditioners are mainly implemented using *Token Bucket* or *Time-Sliding Window*.

Token Bucket

A *Token Bucket (TB)* [46] monitors packets arrivals allowing bursts of up to B_{TB} bytes and a maximum rate R_{TB}. The TB algorithm is as follows:

- A token is added to the bucket every $1/R_{TB}$ seconds.
- The capacity of the bucket is B_{TB} tokens. If a token arrives when the bucket is full, it is discarded.
- When a packet of P bytes arrives, P tokens are removed from the bucket, and the packet is sent to the next DiffServ module.
- If fewer than P tokens are available, no tokens are removed from the bucket, and the packet is considered to be out-of-profile.

Note that B_{TB}, measured in Bytes, is recommended to be equal to or greater than the size of the largest possible IP packet in the stream.

Time-sliding Window

The *Time-Sliding Window (TSW)* meter [47] estimates the sending rate upon each packet arrival taking into account the last estimated rate $\hat{R}_{j-1}$, initialized to the target rate at the beginning of the algorithm. If a packet of size P arrives, the sending rate is calculated using

$$\hat{R}_j = \frac{\hat{R}_{j-1} W_L + P}{W_L + t_L} \tag{G.1}$$

where t_L is the time since last packet arrival and W_L is the window length of past history in units of time. It is recommended to select a W_L larger than the largest RTT [48].

G.3.3 Markers

Packet markers [3] set the DSCP field of a packet according to the state of a meter. When the marker changes the codepoint in a packet it is said to have *re-marked* the packet.

TB Marker

The *TB Marker (TBM)* [49] is the simplest marker of all. It is based on a *Committed Information Rate (CIR)* and its associated *Committed Burst Size (CBS)*. It makes use of a TB meter TB_C configured as stated in Table G.1, leading to two drop precedences.

Two-Rate Three-Color Marker

The *two-rate Three-Color Marker (tr3CM)* [50] marks the packets of an IP stream based on two rates, a *Peak Information Rate (PIR)* and a CIR, and their associated *Peak Burst Size (PBS)* and CBS. A packet is marked red if it exceeds the PIR. Otherwise it is marked either yellow or green depending on whether it exceeds the CIR or not. This is performed using two TB meters, TB_P and TB_C, whose configuration parameters can be found in Table G.1. The token buckets are initially full. When a packet of size P bytes arrives:

- If the packet has been pre-marked as red or if fewer than P tokens are available at TB_P, the packet is marked as red.
- Else, if the packet has been pre-marked as yellow or if fewer than P tokens are available at TB_C, the packet is marked as yellow and P tokens are removed from TB_P.

- Otherwise, the packet is marked as green and P tokens are removed from TB_P and TB_C.

Token Bucket Parameters		TB_P	TB_C	TB_E
Maximum burst size	$B_{TB}(Bytes)$	PBS	CBS	EBS
Maximum rate	$R_{TB}(Mbps)$	PIR	CIR	CIR

Table G.1: Token Bucket configurations for TBM, tr3CM and sr3CM.

Single-Rate Three-Color Marker

The *single-rate Three-Color Marker (sr3CM)* [51] marks packets as either green, yellow or red, according to three traffic parameters: CIR, CBS and *Excess Burst Size (EBS)*[3]. Packets are marked as green if they do not exceed the CBS, as yellow if they exceed the CBS, but not the EBS, and as red otherwise. As in tr3CM, two TB meters are used, TB_C and TB_E (see Table G.1). Both token buckets are initially full. When a packet of size P bytes arrives:

- If the packet has been pre-marked as green and there are more than P tokens available at TB_C, the packet is marked as green and P tokens are removed from TB_C.
- Else, if the packet has been pre-marked as green or yellow and there are more than P tokens available at TB_E, the packet is marked as yellow and P tokens are removed from TB_E.
- Otherwise, the packet is marked as red.

TSW Two-Color Marker

The *TSW Two-Color Marker (TSW2CM)* [49] uses a CIR and two drop precedences. The average rate $\hat{R}_j$ is estimated with the TSW meter. When the estimated rate exceeds the CIR, packets are marked as out-of-profile with a probability p_c obtained from:

$$p_c = \frac{\hat{R}_j - CIR}{\hat{R}_j} \tag{G.2}$$

[3]Maximum number of bits that can exceed the burst size in the first interval of a congestion event.

TSW Three-Color Marker

The *TSW Three-Color Marker (TSW3CM)* [48] marks packets with three drop precedences based on a CIR and a PIR. It makes use of the TSW to estimate the average rate $\hat{R}_j$. The algorithm is as follows:

- If $\hat{R}_j \leq CIR$, packets of the stream are marked as green.
- Else, if $CIR > \hat{R}_j \leq PIR$, packets are marked as yellow with probability p_c and marked as green with probability $(1 - p_c)$.
- Otherwise, packets are marked red with probability p_p, marked yellow with probability p_{cp} and marked green with probability $(1-(p_p+p_{cp}))$.

Note that the probability p_c is obtained from Equation G.2 and p_p and p_{cp} can be obtained from

$$p_p = \frac{\hat{R}_j - PIR}{\hat{R}_j} \tag{G.3}$$

$$p_{cp} = \frac{PIR - CIR}{\hat{R}_j} \tag{G.4}$$

G.3.4 Shapers

Traffic shapers [3] delay some or all of the packets in a traffic stream in order to bring the stream into compliance with the traffic profile. They have a finite-size buffer, so packets may be discarded if there is not sufficient buffer space to hold the delayed packets.

G.3.5 Droppers

Droppers [3] discard some or all of the packets in a traffic stream so as to carry out the traffic profile. This process is know as *policing* the stream. As explained in Section G.1, the dropping causes a rate adjustment. TCP has two ways of dealing with dropped packets:

- The *fast recovery* mode of the sending TCP source halves the window size each time a packet is dropped, and hence, it reduces the sending rate up to half.
- The *slow start* mode occurs when the retransmission timer goes off so that the window size is reduced to one, as a more drastic measure to avoid congestion.

DropTail

A *DropTail* buffer at a router drops every arriving packet when it is full. A buffer overflow could cause the TCP connections associated to that queue reduce their window size at the same time, which is known as global synchronization. Moreover, in this situation, bursty traffic from the same source would be discarded, producing an effect known as bias against bursty cross-traffic.

Random Early Detection

Random Early Detection (RED) routers for congestion avoidance [33] detect incipient congestion by computing the average queue size $\hat{Q}$, calculated using a low-pass filter with an exponential weighted moving average. The aim of RED algorithm is to avoid dropping many consecutive packets and having too long intervals between dropped packets, which can lead to global synchronization [33]. To do so, it follows the next algorithm:

- Normal operation ($\hat{Q} \leq Q_{min}$): no packets are dropped.
- Congestion avoidance ($Q_{min} < \hat{Q} \leq Q_{max}$): packets are dropped with probability p_d, which is a function of the average queue size, designed to slowly increase as a counter n increases. This counter is increased every time a packet arrives and is reset when a packet is dropped. This probability is calculated as

$$p_d = \frac{p_t}{1 - np_t} \tag{G.5}$$

  where p_t is obtained from

$$p_t = \frac{\hat{Q} - Q_{min}}{Q_{max} - Q_{min}} p_{max} \tag{G.6}$$

  where p_{max} is a RED parameter that determines the maximum probability for p_t.

- Congestion control ($\hat{Q} > Q_{max}$): every arriving packet is dropped.

Instead of dropping the packet, RED routers may set to 1 the *Explicit Congestion Notification (ECN)*[4] bit to signal the presence of congestion to the transport layer protocol [47].

[4]The ECN bit is included in the *Type of Service (ToS)* field of the IP header.

RED Routers with In/Out Bit

Diffserv droppers can be implemented with *RED Routers with In/Out Bit (RIO)*. They use twin RED algorithms for dropping packets, one for in-profile packets and one for out-of-profile packets, being the latter more aggressive in dropping packets, so the twin RED queues parameters should be chosen accordingly. One configuration is *RIO De-coupled (RIO-D)*, in which the probability p_t^{in} is obtained from the average queue size of the in-profile packets $\hat{Q}^{in}$, as Figure G.3(a) shows, whereas p_t^{out} is obtained from the average queue size of the out-of-profile packets $\hat{Q}^{out}$ (see Figure G.3(b)). Another configuration is *RIO Coupled (RIO-C)*, in which p_t^{in} is calculated as in RIO-D, and p_t^{out} is calculated from the average queue size of the out-of-profile and in-profile packets together $\hat{Q}^{in+out}$, as Figure G.3(c) illustrates.

Weighted RED

In *Weighted RED (WRED)*, all probabilities are based on a single queue length, sharing Q_{max} (see Figure G.3(d)). Both probabilities p_t^{in} and p_t^{out} are calculated taking into account $\hat{Q}^{in+out}$.

G.4 Traffic Scheduling

The order of the packets to leave a router, taking into account their traffic class, is determined by the scheduling configuration. The main scheduling modes are:

- *Round Robin (RR)* is the simplest scheduling algorithm. A DiffServ router configured for RR scheduling alternatively sends one packet of each traffic class. All classes receive a share of the network resources proportional to their sending rate.

- *Weighted Round Robin (WRR)* schedules packets according to the weights associated with their traffic classes. When all the weights are equal, the scheduling mode is RR. For instance, let W_1, W_2 and W_3 be the queue weights for the three traffic classes of a network. Then, W_2 out of $(W_1 + W_2 + W_3)$ sent packets will be class-2 packets.

- *Priority (PRI)* is based on the idea that no packet is sent while there is another packet in the queue with a higher priority. It is used when high priority traffic is critical, but it may lead the low priority traffic not to be forwarded at all.

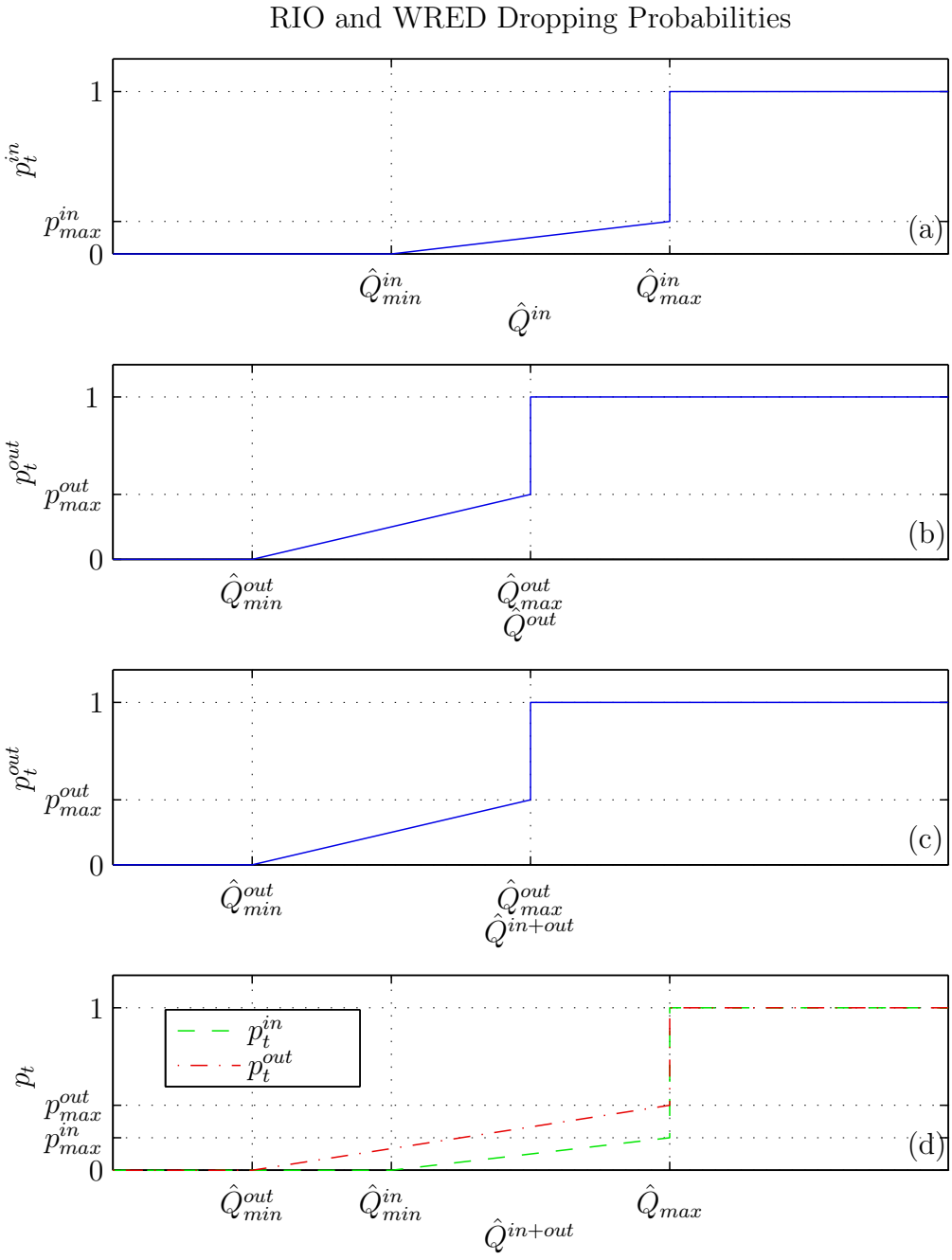

Figure G.3: In-profile packet dropping probability in RIO-D and RIO-C (a), out-of-profile packet dropping probability in RIO-D (b), out-of-profile packet dropping probability in RIO-C (c) and packet dropping probability in WRED (d).

G.5 Per-hop Behaviors

Network nodes that implement the DiffServ enhancements use the DSCP of a packet to select a per-hop behavior (PHB) as the specific forwarding treatment for that packet.

G.5.1 Assured Forwarding

Assured Forwarding (AF) [52] offers different levels of QoS to different flows while it guarantees that they do not exceed the subscribed information rate, i.e. packets are in-profile, during times of congestion. The AF DSCPs are defined as $AFyx$, with y representing the traffic class and x representing the precedence. A packet with a higher y have to have a higher scheduling priority, while a packet with a higher x have to have a higher dropping probability.

G.5.2 Expedited Forwarding

Expedited Forwarding (EF) [53] attempts to provide a low delay, low jitter and low loss service to a determined traffic aggregate by ensuring that it is served at a certain rate, independent of the load of other (non-EF) traffic. A priority queue is the perfect example for an EF implementation, having the EF aggregate the highest priority. Another option for the EF implementation is a WRR scheduler in which the queue weights are chosen to guarantee the EF traffic a certain rate.

G.5.3 Best Effort

Best Effort (BE) does not make any commitment as to QoS, which means that the traffic is allocated according to the available resources. To do so, BE traffic usually has the lowest scheduling priority and a low dropping probability.