End-to-End Available Bandwidth Estimation and Monitoring

Cesar Dario Guerrero Santander

University of South Florida

Follow this and additional works at: https://scholarcommons.usf.edu/etd

Part of the American Studies Commons

Scholar Commons Citation
Guerrero Santander, Cesar Dario, "End-to-End Available Bandwidth Estimation and Monitoring" (2009).
Graduate Theses and Dissertations.
https://scholarcommons.usf.edu/etd/1994

This Dissertation is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
End-to-End Available Bandwidth Estimation and Monitoring

by

Cesar Dario Guerrero Santander

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy
Department of Computer Science & Engineering
College of Engineering
University of South Florida

Major Professor: Miguel A. Labrador, Ph.D.
Kenneth Christensen, Ph.D.
Tapas K. Das, Ph.D.
Adriana Iamnitchi, Ph.D.
Rafael Perez, Ph.D.

Date of Approval:
February 20, 2009

Keywords: Hidden Markov Model, Traceband, Network Measurement, Moving Average, Network Testbed, Dummynet

© 2009, Cesar Dario Guerrero Santander
To my family
Acknowledgements

I would like to gratefully thank my mentor and advisor Dr. Miguel Labrador for his guidance, patience, and most importantly, his friendship during my graduate studies. He gave me the opportunity to go beyond my academic and research duties and to make this a meaningful personal and professional experience. I am also thankful to him for supporting me as Research Assistant.

I would like to thank the University of South Florida, Universidad Autonoma de Bucaramanga and the Colombian Fulbright Commission for supporting this work. I would also like to sincerely thank Dr. Christensen, Dr. Perez, Dr. Iamnitchi, and Dr. Das for being part of my committee and for their valuable comments and suggestions.

I want to express my gratitude to my wife Marcela. Her love, patience, understanding, and encouragement enabled me to complete this dream. Thanks for Sarah. This dissertation is dedicated to them. I also thank my parents and sisters, for their love and support at every step of my life.

Finally, and most importantly, I thank God for guiding my life and my dreams.
# Table of Contents

List of Tables v

List of Figures vii

Abstract xiii

Chapter 1 Introduction 1

1.1 Background 2

1.1.1 End-to-End Path 3

1.1.2 End-to-End Available Bandwidth 3

1.1.3 Available Bandwidth Estimation 5

1.2 Why is the Estimation of the Available Bandwidth Difficult? 6

1.2.1 System Timing 7

1.2.2 End-host Throughput 9

1.2.3 End-to-End Pathologies 9

1.2.4 Queuing Behavior 10

1.3 Problem Statement 10

1.4 Contributions 11

1.5 Organization of the Dissertation 12
Chapter 2 Literature Review

2.1 Available Bandwidth Estimation Techniques

2.1.1 Probe Gap Model (PGM)

2.1.2 Probe Rate Model (PRM)

2.2 Available Bandwidth Estimation Tools

2.2.1 Spruce

2.2.2 Abing

2.2.3 IGI

2.2.4 Pathload

2.2.5 Pathchirp

Chapter 3 Evaluation of Current Available Bandwidth Estimation Tools

3.1 Performance Metrics

3.2 Testbed

3.3 Analytically-Based Available Bandwidth Evaluation

3.3.1 Experiments

3.3.2 Results

3.3.2.1 Estimation Error

3.3.2.2 Overhead

3.3.2.3 Estimation Time

3.4 Experimentally-Based Available Bandwidth Evaluation

3.4.1 Phase One: The \(2^5\) Factorial Design
List of Tables

Table 1.1  End-host NIC achievable throughput. 9
Table 3.1  Parameters used by the estimation tools. 36
Table 3.2  Factors and levels in the $2^5$ factorial design. 48
Table 3.3  Main effect in the performance metrics when varying one factor. 50
Table 3.4  Summary of best/worst performing tools. 77
Table 3.5  Qualitative assessment of application requirements and best tools for the set of representative applications. 77
Table 4.1  Available bandwidth HMM variables. 87
Table 5.1  Performance evaluation for 30% Poisson cross traffic. 95
Table 5.2  Performance evaluation for 30% burst cross traffic. 99
Table 5.3  Performance evaluation for 30% self-similar cross traffic. 101
Table 5.4  Performance evaluation with real cross traffic in a 100 Mbps path. 101
Table 5.5  Estimation error after applying moving average to experiment results with Poisson traffic. 105
Table 5.6  Estimation error after applying the moving average to experiment results with self-similar traffic. 107
Table 5.7  Traceband performance for high and low tight link capacities. 109
| Table A.1 | $2^5$ factorial design matrix. | 124 |
| Table A.2 | Main effect in the performance metrics when varying two factors. | 126 |
| Table A.3 | Main effect in the performance metrics when varying three factors. | 127 |
| Table A.4 | Main effect in the performance metrics when varying four factors. | 128 |
| Table A.5 | Main effect in the performance metrics when varying five factors. | 129 |
| Table B.1 | Observation probability matrix for 5 states and 5 observation symbols. | 130 |
| Table B.2 | Observation probability matrix for 10 states and 10 observation symbols. | 130 |
List of Figures

Figure 1.1  End-to-end communication path.  
Figure 1.2  Available bandwidth in an average timescale period.  
Figure 1.3  Narrow and tight links.  
Figure 1.4  Single link model for bandwidth estimation.  
Figure 2.1  IGI turning point.  
Figure 2.2  Pathload gray region.  
Figure 2.3  Chirp queuing delay signature.  
Figure 3.1  Testbed to evaluate bandwidth estimation tools.  
Figure 3.2  Network of queues for the evaluation testbed.  
Figure 3.3  Pathload estimation with 0% cross traffic.  
Figure 3.4  IGI estimation with 0% cross traffic.  
Figure 3.5  Spruce estimation with 0% cross traffic.  
Figure 3.6  Pathload estimation with 25% cross traffic.  
Figure 3.7  IGI estimation with 25% cross traffic.  
Figure 3.8  Spruce estimation with 25% cross traffic.  
Figure 3.9  Pathload estimation with 50% cross traffic.
Figure 3.10 IGI estimation with 50% cross traffic.

Figure 3.11 Spruce estimation with 50% cross traffic.

Figure 3.12 Pathload estimation with 75% cross traffic.

Figure 3.13 IGI estimation with 75% cross traffic.

Figure 3.14 Spruce estimation with 75% cross traffic.

Figure 3.15 Pathload relative error for 0%, 25%, 50% and 75% cross traffic.

Figure 3.16 IGI relative error for 0%, 25%, 50% and 75% cross traffic.

Figure 3.17 Spruce relative error for 0%, 25%, 50% and 75% cross traffic.

Figure 3.18 Pathload overhead for 0%, 25%, 50% and 75% cross traffic.

Figure 3.19 IGI overhead for 0%, 25%, 50% and 75% cross traffic.

Figure 3.20 Spruce overhead for 0%, 25%, 50% and 75% cross traffic.

Figure 3.21 Pathload time for 0%, 25%, 50% and 75% cross traffic.

Figure 3.22 IGI time for 0%, 25%, 50% and 75% cross traffic.

Figure 3.23 Spruce time for 0%, 25%, 50% and 75% cross traffic.

Figure 3.24 Main effect on the response variables when factors are varied.

Figure 3.25 Estimation error at 20% cross traffic with variable capacity, 10 ms OWD, 1% PLR.

Figure 3.26 Estimation error at 75% cross traffic with variable capacity, 10 ms OWD, 1% PLR.

Figure 3.27 Convergence time at 20% cross traffic with variable capacity, 10 ms OWD, 1% PLR.

Figure 3.28 Convergence time at 75% cross traffic with variable capacity, 10 ms OWD, 1% PLR.
Figure 3.29 Overhead at 20% cross traffic with variable capacity, 10 ms OWD, 1% PLR.

Figure 3.30 Overhead at 75% cross traffic with variable capacity, 10 ms OWD, 1% PLR.

Figure 3.31 Estimation error at 5 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

Figure 3.32 Estimation error at 100 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

Figure 3.33 Estimation time at 5 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

Figure 3.34 Estimation time at 100 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

Figure 3.35 Overhead at 5 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

Figure 3.36 Overhead at 100 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

Figure 3.37 Estimation error at 5 Mbps for variable packet loss rate, 10 ms delay, and 75% cross traffic.

Figure 3.38 Estimation error at 100 Mbps for variable packet loss rate, 10 ms delay, and 75% cross traffic.

Figure 3.39 Estimation time at 5 Mbps for variable packet loss rate, 10 ms delay, and 75% cross traffic.

Figure 3.40 Estimation time at 100 Mbps for variable packet loss rate, 10 ms delay, and 75% cross traffic.

Figure 3.41 Reliability at 5 Mbps for variable packet loss rate, 10 ms delay, and 75% cross traffic.

Figure 3.42 Reliability at 100 Mbps for variable packet loss rate, 10 ms delay, and 75% cross traffic.
Figure 3.43 Estimation error at 5 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR. 68

Figure 3.44 Estimation error at 100 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR. 68

Figure 3.45 Estimation time at 5 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR. 69

Figure 3.46 Estimation time at 100 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR. 69

Figure 3.47 Overhead at 5 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR. 70

Figure 3.48 Overhead at 100 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR. 70

Figure 3.49 Estimation error at 5 Mbps and 20% cross traffic for variable cross traffic packet size, 10 ms OWD, and 1% PLR. 71

Figure 3.50 Estimation error at 5 Mbps and 75% cross traffic for variable cross traffic packet size, 10 ms OWD, and 1% PLR. 71

Figure 3.51 Estimation error at 100 Mbps and 20% cross traffic for variable cross traffic packet size, 10 ms OWD, and 1% PLR. 71

Figure 3.52 Estimation error at 100 Mbps and 75% cross traffic for variable cross traffic packet size, 10 ms OWD, and 1% PLR. 72

Figure 4.1 Available bandwidth Markov model. 81

Figure 4.2 Hidden Markov model for available bandwidth estimations. 84

Figure 5.1 Traceband receiver pseudo code. 93

Figure 5.2 Testbed to evaluate the performance of Traceband. 94

Figure 5.3 Pathload estimation for a 10 Mbps tight link with 30% of Poisson cross traffic. 96
Figure 5.4  Spruce estimation for a 10 Mbps tight link with 30% of Poisson cross traffic.

Figure 5.5  Traceband estimation for a 10 Mbps tight link with 30% of Poisson cross traffic.

Figure 5.6  Pathload estimation for a 10 Mbps tight link with 30% of bursty cross traffic.

Figure 5.7  Spruce estimation for a 10 Mbps tight link with 30% of bursty cross traffic.

Figure 5.8  Traceband estimation for a 10 Mbps tight link with 30% of bursty cross traffic.

Figure 5.9  Pathload estimation for a 10 Mbps capacity and 30% self-similar cross traffic.

Figure 5.10  Spruce estimation for a 10 Mbps capacity and 30% self-similar cross traffic.

Figure 5.11  Traceband estimation for a 10 Mbps capacity and 30% self-similar cross traffic.

Figure 5.12  Pathload estimation for a 100 Mbps tight link with Internet cross traffic.

Figure 5.13  Spruce estimation for a 100 Mbps tight link with Internet cross traffic.

Figure 5.14  Traceband estimation for a 100 Mbps tight link with Internet cross traffic.

Figure 5.15  Moving average post processing to Pathload experiments with Poisson cross traffic.

Figure 5.16  Moving average post processing to Spruce experiments with Poisson cross traffic.

Figure 5.17  Moving average post processing to Traceband experiments with Poisson cross traffic.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.18</td>
<td>Moving average algorithm.</td>
<td>105</td>
</tr>
<tr>
<td>5.19</td>
<td>Moving average post processing to Pathload experiments with self-similar cross traffic.</td>
<td>106</td>
</tr>
<tr>
<td>5.20</td>
<td>Moving average post processing to Spruce experiments with self-similar cross traffic.</td>
<td>106</td>
</tr>
<tr>
<td>5.21</td>
<td>Moving average post processing to Traceband experiments with self-similar cross traffic.</td>
<td>107</td>
</tr>
<tr>
<td>5.22</td>
<td>Effect of the Hurst parameter in the estimation error.</td>
<td>108</td>
</tr>
<tr>
<td>5.23</td>
<td>Traceband estimation error and time for different number of states.</td>
<td>110</td>
</tr>
</tbody>
</table>
End-to-End Available Bandwidth Estimation and Monitoring

Cesar Dario Guerrero Santander

ABSTRACT

Available Bandwidth Estimation Techniques and Tools (ABETTs) have recently been envisioned as a supporting mechanism in areas such as compliance of service level agreements, network management, traffic engineering and real-time resource provisioning, flow and congestion control, construction of overlay networks, fast detection of failures and network attacks, and admission control. However, it is unknown whether current ABETTs can run efficiently in any type of network, under different network conditions, and whether they can provide accurate available bandwidth estimates at the timescales needed by these applications.

This dissertation investigates techniques and tools able to provide accurate, low overhead, reliable, and fast available bandwidth estimations. First, it shows how it is that the network can be sampled to get information about the available bandwidth. All current estimation tools use either the probe gap model or the probe rate model sampling techniques. Since the last technique introduces high additional traffic to the network, the probe gap model is the sampling method used in this work. Then, both an analytical and experimental approach are used to perform an extensive performance evaluation of current available bandwidth estimation tools over a flexible and controlled testbed. The
results of the evaluation highlight accuracy, overhead, convergence time, and reliability performance issues of current tools that limit their use by some of the envisioned applications. Single estimations are affected by the bursty nature of the cross traffic and by errors generated by the network infrastructure.

A hidden Markov model approach to end-to-end available bandwidth estimation and monitoring is investigated to address these issues. This approach builds a model that incorporates the dynamics of the available bandwidth. Every sample that generates an estimation is adjusted by the model. This adjustment makes it possible to obtain acceptable estimation accuracy with a small number of samples and in a short period of time.

Finally, the new approach is implemented in a tool called Traceband. The tool, written in ANSI C, is evaluated and compared with Pathload and Spruce, the best estimation tools belonging to the probe rate model and the probe gap model, respectively. The evaluation is performed using Poisson, bursty, and self-similar synthetic cross traffic and real traffic from a network path at University of South Florida. Results show that Traceband provides more estimations per unit time with comparable accuracy to Pathload and Spruce and introduces minimum probing traffic. Traceband also includes an optional moving average technique that smooths out the estimations and improves its accuracy even further.
Chapter 1: Introduction

The new century has seen a continuous increasing number of Internet users and network applications. Internet users have grown around 300% from 2000 to 2008 [1] [2] and network applications have grown from email to voice over IP, video steaming, peer to peer (P2P) file transfers, overlay networks, among others. For some of these network applications, information about the available bandwidth can be used to improve their performance. For example, network management tools that monitor large networked systems can use available bandwidth data to show the current utilization of the network resources. Internet service providers and users can monitor and verify service level agreements (SLA) to manage their contracts. Traffic engineering mechanisms would be able to perform real-time resource provisioning while balancing the load of the network. Call admission control mechanisms might take advantage of available bandwidth information to either admit or reject a new incoming connection, avoiding network congestion and guaranteeing the quality of service of current and new connections. Overlay networks could determine the most appropriate topology based on available bandwidth information. Transport layer protocols might decide to change the transmission rate according to the amount of bandwidth available in the path, using the network resources efficiently while avoiding congestion. Also, network available bandwidth information could be an important indicator to detect network failures and malicious attacks.
Although the envisioned usefulness of available bandwidth information is not in question, current available bandwidth estimation tools cannot be used by most of the network applications requiring the estimation [3]. Moreover, they cannot be used in every network scenario. For example, while an available bandwidth estimation with a 10% error could be considered within acceptable values for a routing protocol to make routing decisions, it may be completely unacceptable for SLA verification. Similarly, it may be just fine for a tool to take several seconds or even minutes to provide an estimate for a network management system, but it would be useless for a transport layer protocol to make rate changing decisions. Finally, although it may not be a big issue to use a very intrusive available bandwidth estimation tool in an optical network, it may consume very scarce and precious resources in a wireless mobile ad hoc network.

This dissertation studies current available bandwidth estimation techniques and tools and proposes a novel accurate, low-overhead, reliable, and fast estimation approach that has been implemented in an available bandwidth estimation tool called Traceband.

1.1 Background

In computer networks, bandwidth is a rate measure defined as the amount of bits transmitted in a communication channel per unit time. It is generally specified in bits per second (bps). There are two different metrics related to bandwidth. One is the capacity and the other one is the the available bandwidth. The capacity of a link is the maximum amount of bits that can be transmitted to the link per unit time. That is, the maximum bandwidth. The available bandwidth is the spare capacity.
1.1.1 End-to-End Path

An end-to-end communication path is a single route that connects two end hosts through a set of communication links or hops connected via network devices. Although that route can change, it has been shown by Paxson [4] and Zhang [5] that end-to-end paths between Internet hosts are stable on scales ranging from hours to days.

As it is shown in Figure 1.1, sender and receiver end hosts are communicated through four single links (or four hops) connected via routers A,B, and C. Each link in the path has a particular capacity determined by the Network Interface Controller (NIC) attached to the corresponding network device in the path. For example, the capacity of the link between routers B and C is determined by the NIC’s maximum transmission rate in router B connected to router C.

1.1.2 End-to-End Available Bandwidth

The minimum of all non-utilized link capacities throughout the communication path is called the end-to-end available bandwidth. This is a time-varying metric related to the individual utilization of each link throughout the path. Defining $\tau$ as the averaging
timescale of the available bandwidth [6], the average utilization of link \( i \) for a sample of time \( \tau \), is given by

\[
\overline{u}_i = \frac{1}{\tau} \int_t^{t+\tau} u_i(s)\,ds, \quad 0 \leq \overline{u}_i \leq 1.
\] (1.1)

For a link \( i \) with capacity \( C_i \), the available bandwidth of the link in the interval \((t, t + \tau)\) can be defined as the average non-utilized capacity during the time \( \tau \) (see Figure 1.2):

\[
\overline{A}_i = C_i[1 - \overline{u}_i].
\] (1.2)

For an end-to-end path with \( H \) hops, the available bandwidth during \( \tau \) is given by the link with the minimum non-utilized capacity of all hops, as follows:

\[
\overline{A} = \min_{i=1..H}(\overline{A}_i).
\] (1.3)

As it is shown in Figure 1.3, the link with the minimum capacity is known as the narrow link and the link with the minimum available bandwidth is known as the tight link, which
Figure 1.3: Narrow and tight links.

is considered the bottleneck of the path and the link that determines the end-to-end available bandwidth.

1.1.3 Available Bandwidth Estimation

To estimate the available bandwidth in an end-to-end path it is necessary to sample the network by sending probing packets. Although most of the available bandwidth tools generate those packets as additional traffic in the network, Man et al. [7] [8] propose the use of carefully selected and delayed data packets to serve as probing packets without inserting additional traffic to the network.

From the analysis of the delays that probing packets suffer when passing through the tight link, the available bandwidth can be determined. The behavior of a probing packet pair after leaving the tight link is shown in Figure 1.4. This single link model is based on the assumption of a single queue following the first-come-first-served discipline. As it is shown in the Figure, if two consecutive packets are sent to the network path, they arrive to the node with a determined initial time-separation between them ($\Delta_{in}$). After interacting in the tight link queue with the cross traffic coming from different sources, the pair of
probing packets will leave the router with a new time-separation ($\Delta_{\text{out}}$). The difference between them $\Delta_{\text{out}} - \Delta_{\text{in}}$ is the packet pair dispersion.

The packet pair dispersion can be negative, positive, or equal to zero. As it is shown in Figure 1.4, a negative value ($\Delta_{\text{out}} < \Delta_{\text{in}}$) occurs when the first packet finds cross traffic packets in the queue followed by the second packet. A positive value ($\Delta_{\text{out}} > \Delta_{\text{in}}$) occurs when cross traffic packets are inserted between the probing packet pair in the queue. Finally, a value of zero ($\Delta_{\text{out}} = \Delta_{\text{in}}$) occurs when the link has no enough cross traffic to affect the initial packet separation.

Based on the tight link model, there are two different approaches to estimate the available bandwidth in an end-to-end path: the probe gap model (PGM) and the probe rate model (PRM). PGM observes probing packet pair dispersions to estimate the amount of cross traffic. PRM observes variations in the probing packet one way delay to determine the available bandwidth. Both models will be described in Section 2.1.

1.2 Why is the Estimation of the Available Bandwidth Difficult?

The estimation of the available bandwidth is difficult for two main reasons. One is the burst nature of the cross traffic and the other is related to errors generated by end hosts.
and routers along the end-to-end path. Due to the burst nature of the cross traffic, a single pair of probing packets cannot capture the average traffic load in the single link model described before. To deal with this problem, estimation tools based on both the PGM and PRM approaches use a train of probing packets to generate a single averaged measurement. This solution is also used in the tool presented as part of this work.

In addition, end hosts and routers are sources of errors to the estimation tools [9] [10] [11]. Incorrect packet time stamps, poor NIC utilization, out-of-order packet delivery, packet replication, packet corruption, and changing queuing behaviors affect the accuracy of the estimation performed by a single pair of probing packets. Most of these errors can be corrected in a controlled testing environment but not in a real scenario where users can run the estimation application but have no knowledge about how to conveniently set up their machines. The estimation tool presented in this work does not prevent these errors to occur but builds a model of the available bandwidth to statistically adjust the erratic measurements. The following sections provide a more detailed explanation of these errors.

1.2.1 System Timing

When sending probing packets and measuring their gaps or rates at the receiver, accurate timing is required. The time at which packets are time stamped right before a "sendto ()" socket function at the sender is different to the time at which the packet is actually sent. Similarly, the time at which a packet is seen by the receiver’s NIC is different from the time it is reported to the estimation application. Several factors contribute to timing errors:
• **Timer resolutions** in most operating systems are around 1 ms. Sending and time stamping probing packets will usually require a better resolution. For example, the transmission time of a 100-Bytes packet in a 100 Mbps link is 8 µs. If two of those packets are sent back-to-back, their time stamps would be identical if a 1 ms-resolution timer is in place.

• **Context switch** can affect the gap time between packets when the estimation process is abruptly suspended by the operating system to assign the CPU to another process. Specifically, most of the estimation tools transmit a train of packets at desired time by reading the system clock in a polling loop until the whole train is completely sent. If this loop is interrupted by a context switching event, timing inside that loop will be unreliable.

• **Interrupt coalescence (IC)** [12] [13] is a mechanism implemented in most of high-speed network interface cards that affects appropriate probing packets time-stamping at the receiver side of the estimation application. This mechanism delays the generation of a CPU interruption for every packet arriving to the NIC. Instead, IC stores in the NIC several packets before interrupting the CPU to notify their arrival. Therefore, all packets reported in a single interruption will have the same incorrect time-stamp at the application level.

• **System call delays** due to time-queries like "gettimeofday()" and socket operations like "sendto()" or "recvfrom()" add several microseconds to the measurements. For most operating systems and computer architectures, gettimeofday() takes about 1 µs and sendto() and recvfrom() about 40 µs each.
Table 1.1: End-host NIC achievable throughput.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>CPU</th>
<th>NIC achievable throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10 Mbps</td>
</tr>
<tr>
<td>Linux 2.4.1</td>
<td>Intel P4 2.0 GHz</td>
<td>8.8</td>
</tr>
<tr>
<td>Linux 2.4.1</td>
<td>AMD AthlonXP 2500+</td>
<td>8.9</td>
</tr>
<tr>
<td>FreeBSD 4.8</td>
<td>Intel P4 2.0 GHz</td>
<td>8.8</td>
</tr>
<tr>
<td>Mac OS X 10.2</td>
<td>Power G4 1 GHz</td>
<td>8.6</td>
</tr>
<tr>
<td>Windows XP SP2</td>
<td>Intel P4 2.4 GHz</td>
<td>8.8</td>
</tr>
</tbody>
</table>

1.2.2 End-host Throughput

Estimation applications require probing packets to be sent at a specific rate. However, NIC achievable throughput is inferior to its real capacity. Table 1.1 shows the results of a study performed in [9] testing the achievable throughput for six different operating systems installed on different computers. For example, a 10 Mbps NIC cannot achieve more than 8.9 Mbps. In fact, the greater the capacity the lower the NIC utilization. A 1000 Mbps NIC cannot achieve more than 340 Mbps. This poor NIC utilization is in part caused because end hosts are general-purpose personal computers. Network-oriented devices such as routers are hardware and software designed to achieve very high throughput.

1.2.3 End-to-End Pathologies

Network pathologies as named by Paxson in [10] are referred to unusual or unexpected network events like out-of-order delivery, replication, and corruption that affect most of the estimation mechanisms. Out-of-order delivery is an issue since most estimation methods assume that packets will keep the delivery sequence established by the FIFO queuing policy in the routers. However, any time a route changes, if the new route offers a
lower delay than the old one, then packets could be received in a different order [14]. The reordered packets will in particular affect estimation based on gaps between packet pairs since the pairs sequence will be lost.

Packet replication occurs when the network delivers multiple copies of the same packet. Packet corruption occurs when the network delivers an incorrect copy of the original packet. Both pathologies are obviously harmful when they occur in probing packets used to sample the available bandwidth.

1.2.4 Queuing Behavior

The tight link model assumes a single queue following the first-come-first-served discipline. This is not always the case. Many routers have implemented weighted fair queuing (WFQ) mechanisms that can change the delivering order of the received packets. In addition, due to the burst behavior of the cross traffic, it is shown in Figure 1.4, that the first of a probing packet pair might find the queue busy. This event will produce a negative dispersion between packet pairs that is ignored by most of the estimation mechanisms.

1.3 Problem Statement

Estimating the available bandwidth in an end-to-end path is required by several network applications to improve their performance. However, the estimation accuracy is affected by the burst nature of cross traffic and errors associated to the network infrastructure. These issues force the estimation tools to collect several samples from the network to provide an average available bandwidth within certain accepted values of accuracy. This
increase in accuracy by sending several probing packets increases also the overhead and the time spent to report a result. Similar trade-offs have to be made to provide the convergence time, overhead, and reliability required by the applications.

This dissertation claims that accurate, non-intrusive, reliable and fast end-to-end available bandwidth estimation can be achieved by sending probing packet pairs to the network.

1.4 Contributions

This dissertation makes the following contributions completely described in the remaining chapters of this manuscript.

- An extensive evaluation of current available bandwidth estimation tools.

  Previous evaluations of available bandwidth estimation tools [15] [16] [17] were performed over limited number of tools and network scenarios. This work presents an evaluation of the most important estimation tools using analytic and experimental representations of the real available bandwidth and a factorial design technique to determine the most relevant experiments to be performed over network scenarios never tested before. In addition, this study is unique since it concludes about the usefulness of the estimation tools from the network applications point of view. The results of this contribution are presented in [18] and [3].

- An end-to-end available bandwidth estimation model.

  This is the first work that uses a hidden Markov model approach to represent and estimate the available bandwidth. This approach combined with a moving average technique reduces the number of samples required and provides a fast and accurate estimation. This novel estimation approach has been published in [19].
• A novel end-to-end available bandwidth estimation tool.

Based on the estimation model, a new tool called Traceband is built to provide fast, reliable, accurate and low overhead estimations. This tool is the only tool able to accurately monitor the available bandwidth with a granularity never shown before. Traceband description is presented in [20].

• A research and teaching network infrastructure.

A fully controlled testbed is built to perform the evaluations. This testbed allows to emulate different network scenarios never studied in previous available bandwidth evaluations. This infrastructure is currently used for performance evaluation in the Computer Networks graduate class and has been used to perform the evaluations shown in [21], [18], and [3].

1.5 Organization of the Dissertation

The remainder of the dissertation is organized as follows: Chapter 2 presents the two available bandwidth estimation techniques and the current tools developed using these two approaches. Chapter 3 presents an extensive analytical and experimental evaluation of current available bandwidth estimation tools in scenarios never evaluated before. It is shown how a factorial design technique helps to considerably reduce the number of experiments required in the evaluation. Chapter 4 presents the hidden Markov model approach used in this dissertation as the available bandwidth estimation model to be implemented in a new estimation tool. Chapter 5 describes the operation and implementation of a new available bandwidth estimation tool called Traceband. Chapter 6 concludes the dissertation and presents direction for future research.
Chapter 2: Literature Review

The problem of bandwidth estimation has been studied for several years by many authors. The first approach by Keshav on packet-pair flow control [22] relied on fair queuing in all network routers to estimate bandwidth by sending back-to-back probing packets. Jacobson [23] proposed to use ACK packets to estimate bandwidth based on the spacing between them. Carter introduced cprobe [24] which sends a short train of ICMP echo packets between two hosts and uses the spacing between the first and last returning packet to estimate the available bandwidth. Later, Dovrolis [25] pointed out that what cprobe measures is the asymptotic dispersion rate (ADR), which is different to the available bandwidth. A similar approach was proposed by Jin [26] in a tool called pipechar.

These studies have triggered the development of available bandwidth estimation tools for the last seven years. This chapter introduces the available bandwidth estimation approaches currently used and the estimation tools developed upon them.

2.1 Available Bandwidth Estimation Techniques

There are two different approaches to estimate the available bandwidth in an end-to-end path: the probe gap model (PGM) and the probe rate model (PRM). PGM observes probing packet pair dispersions while PRM observes one way delays in the probing packets.
Both approaches utilize a train of probing packets to generate an averaged estimation and cope in that way with the burstiness nature of cross traffic.

Although not necessary for the estimation tools to work, two assumptions are required to hold for the analytical validity of the estimation models:

- Routers along the path exhibit a FIFO queuing discipline.
- The single-link model shown in Figure 1.4 is one where the cross traffic rate is constant during the averaging timescale $\tau$.

2.1.1 Probe Gap Model (PGM)

This model bases the estimation on the gap dispersion between two consecutive probing packets at the receiver, which has a strong correlation with the amount of cross traffic in the tight link. The dispersion increases linearly with the cross traffic rate if the queue of the tight link (Figure 1.4) does not become empty after the first packet of the pair leaves the router and before the second packet arrives at the router [27]. Therefore, the available bandwidth is estimated by determining the amount of cross traffic and subtracting it from the known capacity of the tight link:

$$\overline{A} = C \times (1 - \varepsilon)$$  \hspace{1cm} (2.1)

where $\varepsilon$ is the relative dispersion or strain [28] defined by:

$$\varepsilon = \frac{\Delta_{\text{out}} - \Delta_{\text{in}}}{\Delta_{\text{in}}}.$$  \hspace{1cm} (2.2)
Examples of available estimation tools based on the probe gap model approach are *Spruce* [29], *Abing* [30] and *IGI* [27].

### 2.1.2 Probe Rate Model (PRM)

This is a model based on the idea of *induced congestion*, in which the tools send probe packet trains at increasing rates and the receiver observe variations in the average train one way delay looking for the turning point, or the point at which the delay of the probe packets starts increasing in a consistent basis. If a train is sent at a rate less than the path available bandwidth, the train will experience similar delays. On the other hand, if the train rate is greater than the path available bandwidth, the train will queue in the tight link router and will experience increasing delays (turning point). The available bandwidth is then estimated looking at the probe packet rate utilized when the turning point is found. At this point, the train rate is equal to the available bandwidth in the end-to-end path.

Examples of tools in the probe rate model are *Pathload* [31] and *Pathchirp* [32].

This method was initially known as the train of packet pair (TOPP) mechanism as defined by Melander [33] [34]. He proposed to inject pairs of probe packets into the network and observes at the receiver the reception times of the probe packets to estimate available bandwidth. The sender starts transmitting a set of $n$ separated pairs of equally sized packets $L$ at some rate $R_{\text{min}}$. This rate is then increased and another train is sent. This goes on until the maximum probing rate $R_{\text{max}}$ is reached. From the relation between the input and output rates, the available bandwidth is estimated. TOPP was only simulated using the network simulator *ns-2* [35].
2.2 Available Bandwidth Estimation Tools

This section describes the estimation tools as presented by their authors. The performance of all the tools will be evaluated in Chapter 3. The basic notation used to explain the operation of the tools is based on the single link model presented in Section 1.1.3. The first three tools presented here use the probe gap model as estimation approach and last remaining two the probe rate model.

2.2.1 Spruce

Spruce [29] uses the probe gap model approach to perform the estimation. It sends a Poisson sample of 1500B UDP pairs of packets with an intra-pair gap equal to the narrow link transmission time of a 1500B packet. That guarantees that the second packet arrives to the narrow link queue before the first packet leaves that queue. By setting the inter-pair gap to the output of an exponentially distributed function, Spruce performs a Poisson sampling process that allows the tool to be non-intrusive.

Using the dispersion of the probe packets measured at the receiver, Spruce calculates the average rate of the traffic that arrives to the queue between the two packets as the capacity of the tight link $C_t$ multiplied by the relative dispersion obtained from Equation 2.2.

The available bandwidth is determined by subtracting that cross traffic rate from the capacity in the tight link. After performing $K$ sample measurements, the tool reports the average of all the available bandwidths calculated. The default value for $K$ is 100. Spruce estimation requires a previous calculation of the tight link capacity.
As presented by the authors, Spruce can be distinguished from other available bandwidth tools by the following aspects:

- Spruce uses a Poisson process of packet pairs rather than packet trains (or chirps). This form of sampling allows Spruce to be both non-intrusive and robust.
- By carefully choosing the value of the initial gap, Spruce ensures that the bottleneck queue does not empty between the two probes in a pair, which is a requirement for the correctness of the gap model.
- Spruce separates capacity measurement from available bandwidth measurement. It assumes that capacity can be measured easily with one of the capacity measurement tools and that capacity stays stable when measuring available bandwidth.
- Spruce does not overwhelm the narrow link on a path because its probe rate is no more traffic than the minimum of 240 Kb/s and 5% of the capacity of the narrow link.
- Apart from the number of pairs K over which to average the measurements, Spruce does not have any tunable parameters

2.2.2 Abing

Abing [30] is based on the probe gap model. It sends twenty back-to-back 1500-Byte long packet pairs with a known separation of 50 ms. After passing through the tight link, according to the authors, probing packets can be separated by cross traffic (CT) packets in any place on the end-to-end path. Separation of the first probing packet (P1) from the second probing packet (P2) of a pair can happen even where there is no real bottleneck
or congestion. The time delay \((Td)\) between P1 and P2 will grow discretely because it is caused by CT packets with particular lengths and finally it will contain the delay caused by all the CT packets inserted between the probing packets in any hop along the path.

The final time delay \(Td\) between packet pairs will have the information of the amount of cross traffic throughout different links with different capacities. That value corresponds to the load on the path. As the load on the path grows, \(Td\) also grows.

The authors observed two components in \(Td\). One component \(Td_{init}\) is common to all individual measurements and is caused by the narrow link; the other component \(Td_{var}\) is variable and reflects queuing changes. Therefore, the packet dispersion has a linear and a non-linear growth:

\[
Td = Td_{init} + Td_{var}. \tag{2.3}
\]

The linear growth occurs when the current hop has a higher utilization factor than the previous hop. The non-linear grow can be caused by the "stretching" effect depicted in Figure 1.4 (when \(\Delta_{out} > \Delta_{in}\)). The authors find what they call a conversion function that relates the available bandwidth with the dispersion value \(Td\). In other words, Abing uses the same probe gap model used by other tools but is unique in the way it estimates the amount of cross traffic traversing every link in the end-to-end path (which is related to \(Td\)).

Abing sends 40 probing packets per measurement to calculate a mean value for \(Td\) and then, the amount of cross traffic and the available bandwidth in the path.
2.2.3 IGI

IGI [27] uses the probe gap model. The authors develop two packet pair techniques to characterize the available bandwidth. One is IGI (initial gap increasing) and the other PTR (packet transmission rate). These techniques are used to experimentally determine the initial gap ($\Delta_{in}$) that will yield a high correlation between the competing traffic throughput on the tight link and the output gap ($\Delta_{out}$) at the destination.

IGI finds an initial probing gap value so that a probing packet train interacts with the cross traffic in a non empty narrow link queue, which is called by the authors the Joint Queuing Region (JQR). In that region, there is a proportional relation between the gap when probing packets leave the queue (output gap) and the cross traffic. The authors find two components in the mathematical definition of the output gap under this JQR region:

$$\Delta_{out} = g_B + \frac{B_C \cdot \Delta_{in}}{C_t}. \quad (2.4)$$

The first component is the time taken to process the first packet $P_1$ (see Figure 1.4) denoted by $g_B$. This value is called by the authors the bottleneck gap since it is the gap value of two back-to-back probing packets on the bottleneck link (which is assumed to be the tight link). The second component is the time taken to process the cross traffic that arrives between the two probing packets $P_1$ and $P_2$. $B_C$ is the competing traffic throughput for the time interval of packets $P_1$ and $P_2$. The key to an accurate available bandwidth estimation by IGI is to find and input gap $\Delta_{in}$ so that the probing packet train operates in this JQR region.

The other region called the Disjoint Queuing Region (DQR) occurs when the second probing packet $P_2$ finds the queue empty. This happens if the queue is empty after $P_1$
leaves the router and before P2 arrives. In that case, the output gap $\Delta_{out}$ is the initial gap minus the queuing delay for P1:

$$\Delta_{out} = \Delta_{in} - \frac{Q}{C}$$

(2.5)

where $Q$ is the queue size when the first packet arrives to the router. The problem is that packet pairs operating in the DQR region will provide wrong values for the purpose of relating the cross traffic with the $\Delta_{out}$. To estimate the amount of competing traffic, IGI focuses on increased gaps in a probing packet train operating in the JQR. Specifically, consider a probing train in which $M$ probing gaps are increased, $K$ are unchanged, and $N$ are decreased. By applying Equation 2.4 it is obtained the estimation of the competing traffic load:

$$B_C = \frac{C \sum_{i=1}^{M} (g_i^+ - g_B)}{\sum_{i=1}^{M} g_i^+ + \sum_{i=1}^{K} g_i^- + \sum_{i=1}^{N} g_i^-}.$$  

(2.6)

That is, the amount of cross traffic that arrive to the router during the probing period divided by the total probing time. Increased, unchanged and decreased gap values are denoted by $g_i^+$, $g_i^-$, and $g_i^-$ respectively. Equation 2.6 is called the IGI formula.

Figure 2.1: IGI turning point. Initial gap is equal to the output gap for 0.8 milliseconds.
Using the same IGI formula notation, if $L$ is the probing packet size, the average transmission rate of the packet train can be estimated by the PTR formula:

$$A = \frac{(M + K + N) \cdot L}{\sum_{i=1}^{M} g_i^+ + \sum_{i=1}^{K} g_i^- + \sum_{i=1}^{N} g_i^-}. \quad (2.7)$$

When the initial gap is increased and equal to the output gap, the available bandwidth on the tight link is equal to the average rate of the packet train. After that point, called by the authors the turning point (see gap of 0.8 milliseconds in Figure 2.1), the narrow link will be overflowed by the probing packets.

Both IGI and PTR algorithms send to the destination a sequence of packet trains with increasing initial gap. They monitor the difference between the average $\Delta_{in}$ and $\Delta_{out}$ gaps until that difference becomes zero. At that point, the packet train is operating at the turning point and the IGI and PTR formulas are applied to compute the final measurement. The available bandwidth is obtained by subtracting the estimated competing traffic throughput from the value of $C_t$ measured by any capacity estimation tool.

Although a key element in IGI is the selection of the initial gap, there are two more factors that affect the accuracy of the tool. The first factor is the selection of the probing packet size. Measurements using small probing packets are very sensitive to interference.

The other factor is the number of probing packets sent. Sending too many packets can cause queue overflow and packet losses, increase the load on the network, and lengthen the time it takes to get an estimate.

By experimentation, the authors show that the quality of the estimates is not very sensitive to the probing packet size and the number of packets, and that there is a fairly large range of good values for these two parameters. For example, a 700-Byte packet size and 60
packets per train work well on the Internet. It is shown by the authors that in the case of multiple hops and significant cross traffic following the tight link, the accuracy of IGI suffers. A similar situation is found when the tight link is not the narrow link. Other authors have found that IGI was unresponsive to variations in cross traffic at Gbps speeds [17].

2.2.4 Pathload

Pathload [31] uses the Self-Loading Periodic Stream (SLoPS) [36] technique which follows the same principle of the probe rate model. In general terms, SLoPS is based on the fact that the one way delay of a periodic packet stream increases when the rate of the probing traffic is higher than the available bandwidth in the path. Otherwise, there is no increase in the delay measured. A fleet of streams (of a fixed number of packets each) are sent at varying rates and the one way delay trend of each stream is then characterized at the receiver as either increasing or decreasing. When that delay is in a gray region where there is not clearly increasing nor decreasing trend (see Figure 2.2), the methodology presents a variation range of the available bandwidth.

A more detailed description of SLoPS is the following. Suppose a sender transmitting a single stream of packets to the receiver. Every packet $i$ is timestamped by the sender before it is transmitted and its arrival time is calculated at the receiver. The difference of both times is the relative one way delay of the packet denoted by $D_i$. After receiving the entire stream of packets, the OWDs values are inspected to check whether the transmission rate of the stream $R$ is larger than the available bandwidth $\bar{A}$. When $R > \bar{A}$, the relative OWDs of the $K$ packets in the stream $\{D_1, D_2, \cdots, D_K\}$ are expected to have an "increasing" trend. This is because the stream creates a short-term overload in
Figure 2.2: Pathload gray region. There is not clearly increasing or decreasing trend in the one way delay between packets 35 and 40.

the tight link. During that period the tight link queue builds up and the queuing delay of packet $i$ in the stream is expected to be larger that the queuing delay of packet $j$ with $i > j$. This effect is what the authors call as self-loading of the periodic stream. When $R < \overline{A}$, the relative OWDS of the $K$ packets are expected to have a "non-increasing" trend. The available bandwidth is given by the rate at which an "increasing" trend in the stream starts to be observed.

To detect the "increasing" trend in the OWDS of a stream, the algorithm implemented in Pathload does the following. The $K$ OWDS measurements are divided in $\Gamma = \sqrt{K}$ groups. For each group, it is calculated the median OWD $\hat{D}_i$ of the group.

Two statistics are used to determine if the stream shows an "increasing" trend. One is called the pairwise comparison test (PCT) which for every stream is calculated by:

$$S_{PCT} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}_k > \hat{D}_{k-1})}{\Gamma - 1} \quad (2.8)$$
where \( I \) is one if \( \hat{D}_k > \hat{D}_{k-1} \), and zero otherwise. A strong "increasing" trend in the OWDs will be detected when \( S_{PCT} \) is close to one. In Pathload an "increasing" trend is reported if \( S_{PCT} > 0.55 \), a "non-increasing" trend if \( S_{PCT} < 0.45 \), and an "ambiguous" trend otherwise.

The other metric is called the pairwise difference test (PDT) which for every stream is calculated by:

\[
S_{PDT} = \frac{\hat{D}_\Gamma - \hat{D}_1}{\sum_{k=2}^{\Gamma} |\hat{D}_k > \hat{D}_{k-1}|}
\]  

(2.9)

A strong "increasing" trend in the OWDs will be detected when \( S_{PDT} \) is close to one. In Pathload an "increasing" trend is reported if \( S_{PCT} > 0.66 \), a "non-increasing" trend if \( S_{PCT} < 0.54 \), and an "ambiguous" trend otherwise.

To determine whether a stream is characterized by an "increasing" trend or not, Pathload does the following. If one of the \( S_{PCT} \) and \( S_{PDT} \) values reports "increasing" trend, while the other is either "increasing" or "ambiguous", the stream is characterized as type-I ("increasing"). If one metric reports "non-increasing" trend while the other is either "non-increasing" or "ambiguous", the stream is characterized as type-N ("non-increasing"). Finally, if both metrics report "ambiguous", or when one is "increasing" and the other is "non-increasing", the stream is discarded. As explained before, when the stream is in a gray region where there is not clearly "increasing" nor "decreasing" trend in the OWDs, Pathload reports a variation range of the available bandwidth.

Pathload sends periodic packet streams (fleets) of UDP traffic and uses a TCP connection to send trend results back to the sender. Given a desired stream rate \( R \), Pathload sets the packet inter-departure time \( T \) at 100 \( \mu \)s and calculates the necessary packet size \( L \) to satisfy \( R = L/T \). If the \( L \) is less than 96 bytes, Pathload uses this minimum value and calculates \( T \) instead.
Figure 2.3: Chirp queuing delay signature. Excursions end when the queuing delay returns to zero (between 3 and 7.5 ms) or when there is an increasing queuing delay (after 10 ms)

2.2.5 Pathchirp

Pathchirp [32] also uses the probe rate model. Instead of sending a packet train (or stream) at a specific rate as Pathload does, Pathchirp increases the probing rate within each train in an exponential manner. By doing that, Pathchirp captures delay correlation information using a smaller number of probing packets. Similar to Pathload, Pathchirp uses information of the relative OWDs of probe packets.

The tool sends several packet chirps to the receiver. Each chirp has $N$ exponentially spaced packets, each of size $P$. There are three main advantages on using chirps. First, the chirp has $N - 1$ packet spacings that would normally require $2N - 2$ packets using packet pairs. Second, exponentially spaced packets require only $\log(G_2) - \log(G_1)$ packets to probe the network over the range of rates $[G_1, G_2]$ Mbps. Finally, chirps capture critical delay correlation information that packet pairs do not.
To better describe a chirp, Figure 2.3 shows what the authors call a "queuing delay signature" of a chirp \( m \). Bursts of cross traffic cause an "excursion" which ends when the queuing delay returns to zero (see excursion from 3 to 7.5 ms in the Figure). That occurs when the chirp rate \( R_k \) is less than the tight link capacity \( C_t \). An excursion can also end with increasing queuing delays as shown in the last excursion after 10 ms. in the Figure. That occurs when the chirp rate \( R_k \) is greater than the tight link capacity \( C_t \), which causes the chirp packets to fill up intermediate queues.

With every signature, Pathchirp makes an estimate \( E_k^{(m)} \) of the per-packet available bandwidth. To obtain the per-chirp available bandwidth \( D^{(m)} \), the per-packet values are averaged using the following equation:

\[
D^{(m)} = \frac{\sum_{k=1}^{N-1} E_k^{(m)} \Delta_k}{\sum_{k=1}^{N-1} \Delta_k}
\]  

(2.10)

where \( \Delta_k \) is the inter-spacing time between packets \( k \) and \( k + 1 \). The average of all per-chirp available bandwidth values is reported by the tool as the final estimation.
Chapter 3: Evaluation of Current Available Bandwidth Estimation Tools

Current available bandwidth estimation tools have been evaluated by different authors. However, the network scenarios and metrics used in the evaluations are limited and their analysis about the applicability of the tools in real network applications is absent. An additional issue is that these evaluations do not include the amount of experiments needed to provide statistically valid conclusions.

For example, in [17], Shriram et al. utilize a high-speed testbed to evaluate Spruce, Abing, Pathchirp, and Pathload. They use passive monitors to verify the actual load level of the generated traffic and test the tools using links of OC-48 and 1 Gbps capacities. The problem with these experiments is that they just provide a partial picture of the evaluation, as the researchers do not have the capability to work with links of different capacities. In [16], Lee et al. describe problems with some bandwidth estimation tools when used on the Planetlab [37] infrastructure. Since the capacity of the links is unknown to the researchers, they use Pathrate [38] to measure the end-to-end capacity of the links. The problem is that the associated error incurred by Pathrate in the estimation of the link capacities introduces errors in the final estimation of the available bandwidth. In [15], Angrisani et al. evaluate IGI, Iperf and Pathload over a local area network and use MGEN [39] as a traffic generator. This evaluation suffers similar flexibility problems like the ones found in [17]. Also, the authors do not measure the overhead generated by the tools.
Previous works do not analyze several issues that have to be considered before implementing any tool in a network application requiring the estimation. Are available bandwidth tools ready to be used in all network applications? Can they provide estimates at the granularity required by specific network applications? Further, can they be used regardless of whether the application is run over a low bandwidth wired network, or a wireless mobile ad hoc network, or a satellite network, or even a high bandwidth and clean optical network? For example, while an available bandwidth estimation with a 10% error could be considered within acceptable values for a routing protocol to make routing decisions, it may be completely unacceptable for SLA verification. Similarly, it may be just fine for a tool to take seconds or even minutes to provide an estimate for a network management system, but it would be useless for a transport layer protocol to make rate changing decisions. Finally, although it may not be a big issue to use a very intrusive active probing estimation tool in an optical network, it may consume very scarce and precious resources in a wireless mobile ad hoc network.

In order to answer the questions above, this chapter presents an evaluation of the main current available bandwidth estimation tools. The estimation provided by the tools is compared with the real value of the available bandwidth. Two approaches are used to perform the evaluation depending on the way this real available bandwidth is determined. The first approach uses an analytical and the second approach uses an experimental value.

This evaluation is novel in several ways. First, a flexible and low-cost testbed is built to include scenarios and network conditions not considered before such as using low, medium, and high link capacities; packet loss rates to simulate lossy links, or very clean links like optical fibers; cross traffic load and distribution to experiment with different levels of network congestion; propagation delays to simulate either local area, wide area, or satellite networks; etc. Second, the first evaluation approach presents an analytical
method never used before to determine the theoretical value of the available bandwidth and compare it with the estimation given by the tools. Third, the approach that uses an experimental value of the real available bandwidth performs a comprehensive set of experiments defined by a factorial design never considered before. Fourth, the evaluation includes a new metric called "reliability" not evaluated before. Finally, it is presented a unique analysis of the tools utility from the network applications point of view.

3.1 Performance Metrics

The three main metrics traditionally used and included in this chapter to evaluate available bandwidth estimation tools are: estimation error, overhead, and estimation time. The estimation error or accuracy metric provides a quantitative value that compares the estimation of the end-to-end available bandwidth, as provided by the tool under consideration, with the real value, which is in this work calculated analytically and experimentally. The estimation error is given by a percentage error. The overhead is related to the amount of probe packets that the tool needs to inject into the network in order to perform the estimation. Most available bandwidth estimation tools are active probing measurement mechanisms and as such they sample the system sending probe packets. The overhead is defined as the percentage of tool traffic respect to the capacity of the tight link. Finally, the estimation time says how long it takes the tool to provide the estimate, and it is usually given in seconds.

In this work, for the evaluation using an experimental obtained value of the real available bandwidth, a fourth metric is added: the reliability, which provides information about the robustness of the tool in providing estimations. The reliability is given by the percentage
of tests the tool succeeded to provide an estimate. It is calculated dividing the number of replications for a particular experiment by the final number of trials needed to perform to reach that number. Thus, a 100% reliable tool needed $N$ trials to provide $N$ estimations.

According to the performance metrics defined before, it could be said that an ideal tool should provide very accurate estimations, with very low or no overhead, in almost no time, and with a 100% reliability. However, not all applications need an ideal tool. Some requirements could be relaxed or tightened according to the application and the networking environment at hand. For example, an application that monitors the compliance of SLAs needs the available bandwidth estimation to have high accuracy, medium overhead, medium or low estimation time, and medium reliability. Although some of the metrics might be arguable, one strong requirement is high accuracy. Similarly, if the available bandwidth is used to drive the flow and congestion control mechanisms of a transport layer protocol, it better be fast and introduce almost no overhead. In this case, the estimation does not need to be very precise, a good estimation might work, but it has to be fast, so that the protocol can react on time to rapidly changing network conditions, and with no overhead because otherwise the huge number of transport layer protocol connections over the Internet will drive the goodput of the network to very low levels. As a final example, take an available bandwidth estimation tool to be used to detect security attacks. In this case, it is easy to see that the most important metrics are the estimation time and the reliability of the tool.
Figure 3.1: Testbed to evaluate bandwidth estimation tools. Intermediate computers act as routers and their names are to mimic locations in a wide area network. ©2006 IEEE.

3.2 Testbed

In order to experimentally evaluate the available bandwidth estimation tools, the testbed shown in Figure 3.1 is built. This is a fully controlled environment with parameterizable links in terms of link capacities, packet loss rates, queues sizes, and propagation delays. It can also be controlled the amount of cross traffic and its statistical distribution on a per link basis. The testbed utilizes low cost computers and open source software, as described next.

There are three main components in the testbed: one client (or sender), four intermediate routers, and one server (or receiver). They are six computers with AMD Athlon 64 3500+ processors, 1GB RAM, and 80 GB hard drive capacity interconnected through two private networks. A 1 Gigabit 192.168.X.X network is utilized to carry all the traffic related to the evaluation of the tools; a 100 Mbps 10.0.0.0 network is used to configure and run the experiments. These two networks are established to separate configuration data from evaluation data. The client and server are Linux-based machines running with 10000 Hz timer granularity. They host the available bandwidth estimation tools under investigation.
Intermediate routers are implemented by four FreeBSD machines emulating a multi-hop network path. The kernel has been recompiled to host a packet shaper called Dummynet [40] and to run at the same end nodes granularity. Dummynet allows to change the capacity of each link from 0 to unlimited (or limited by the physical capacity), and introduce packet losses and delays to emulate lossy and long links. Different queue sizes can also be established if so desired. To generate cross traffic the MGEN [39] traffic generator is used, which allows to choose the rate, packet size, and the statistical distribution of the cross traffic. In order to have independent queues, the cross traffic is introduced on a per link basis, i.e., enters at the input of each queue and leaves the network before the subsequent queue (see Figure 3.1).

The accuracy of this Dummynet-based testbed is experimentally verified. Using netperf and iperf it is observed that the testbed maximum throughput is close to 340 Mbps. According to a ping command, the testbed can emulate delays with 97.53% average precision; tcpdump traces also verify that the testbed emulates packet loss rates with a 99.84% average precision. Finally, tcpdump traces show that MGEN generated traffic with 96.19% accuracy. All the tools under evaluation use the gettimeofday function to query the system’s current time.

A Python client-server application is developed to automate the entire evaluation process. This application allows users to select the bandwidth estimation tool to be evaluated, the link bandwidths, the position of the tight link, the type and rate of the cross traffic and the number of experiments per scenario (to allow statistical significance in the results). The application automatically runs the experiments and collects the results; it reads testing files placed in a particular folder and writes the results in another folder after the experiments are finished. Running experiments remotely and sharing the testbed with others is much easier with this application.
3.3 Analytically-Based Available Bandwidth Evaluation

The first evaluation uses an analytical model to determine the real available bandwidth and compare it with the estimation given by the tools. The idea is to build a model to mimic the behavior of the network of queues in the testbed NICs and to determine from the model the real value of the available bandwidth. This model shown in Figure 3.2 consists of eight M/M/1 queues representing the network interface cards where the tight link will be set and evaluated (queues 1, 3, 5, and 7), where the cross traffic will be routed outside the system (queues 2, 4, and 6), and where output traffic of the system will be received (queue 8).

This network of queues was studied by Jackson [41, 42] in 1957. In Jackson’s model, if \( i \) is the number of nodes in the system \((i=1,2,...,K)\), it is assumed that node \( i \) contains \( n_i \) queues (servers). Also, items arrive from outside the system or from other nodes to node \( i \) at a Poisson rate and are served in turn at an exponential service rate. Once served at a node, an item goes (instantaneously) to node \( j \) \((j=1,2,...,K)\) with probability \( \theta_{ij} \), or out of the system. From these assumptions, in the steady state, the average arrival rate to node \( j \) \((\lambda_j)\) is given by Equation 3.1, where \( \theta_{ij} \) represents the routing probability of going from node \( i \) to node \( j \) and \( \gamma_j \) is the external traffic entering queue \( j \):

\[
\lambda_j = \gamma_j + \sum_{i=1}^{K} \lambda_i \theta_{ij}, \quad \text{for } 1 \leq j \leq K. \tag{3.1}
\]

The underlying stochastic process of the system is defined by

\[
X = \{X_t^i : t \in \mathbb{R}^+, i \in [1..8]\} \tag{3.2}
\]
where $X^t_i$ is the number of packets in the queue and server $i$ at time $t$. It is worth noticing that the probing packet traffic rate $\gamma_0$ to be obtained from experimentation is utilized in Equation 3.1 to calculate the input rates. This is the reason why different theoretical values are obtained for each tool in the same experiment as will be shown later.

The value of $\theta_{ij}$ corresponds to the routing matrix on each queue, which is different from the transition probability matrix of the underlying Markov model. In this evaluation, the routing matrix $\theta_{ij}$ is given by:

$$
\theta_{ij} = 
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{\gamma_0}{\lambda_2} & 0 & 0 & 0 & 0 & 1 - \frac{\gamma_0}{\lambda_2} \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{\gamma_0}{\lambda_4} & 0 & 0 & 1 - \frac{\gamma_0}{\lambda_4} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{\gamma_0}{\lambda_6} & 1 - \frac{\gamma_0}{\lambda_6} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} .
$$

(3.3)
Given all the M/M/1 queues input rates, we can estimate the available bandwidth corresponding to each queue can be estimated as the non-utilized capacity of the system as follows:

\[
\bar{A}_i = 1 - \left( \frac{\lambda_i}{\mu_i} \right) = 1 - \rho_i
\]  

(3.4)

where \(\rho_i\) is the calculated utilization of each queue.

It is worth noticing that this analysis assumes a Poisson distribution for the probing traffic generated by the tools. Although this assumption might not be true, the results obtained indicate that this is not a bad assumption. A reason for that is the low overhead introduced by the tool compared with the amount of cross traffic, which packet interarrival times are exponentially distributed.

3.3.1 Experiments

Using the testbed and the Jackson’s model described before, the tools analyzed under this first evaluation approach are Pathload, IGI and Spruce according to their estimation time, overhead and estimation error metrics. The estimation time in the case of Pathload and IGI is provided directly by the tool. In the case of Spruce, the estimation time is calculated by the difference of times before and after running the tool. The overhead is given by the ratio between the traffic generated by the tool and the capacity of the tight link. In other words, it represents the percentage of the tight link capacity utilized by the tool. The estimation error is calculated comparing the available bandwidth estimation given by the tool with the expected value from the mathematical model using \textit{Matlab}. Plots showing the estimation error on the estimation \(\beta\) use Equation 3.5 for the calculations.
Table 3.1: Parameters used by the estimation tools. ©2006 IEEE.

<table>
<thead>
<tr>
<th>TOOL</th>
<th>Packet Size</th>
<th>Packets/stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathload</td>
<td>Variable. Minimum: 96B</td>
<td>100</td>
</tr>
<tr>
<td>IGI</td>
<td>500B</td>
<td>60 to 256</td>
</tr>
<tr>
<td>Spruce</td>
<td>1500B</td>
<td>100</td>
</tr>
</tbody>
</table>

\[
\beta = \frac{m_A - \mu_A}{\mu_A} \tag{3.5}
\]

where \(m_A\) is the value calculated from experimentation and \(\mu_A\) is the value from the analytical model. The main parameters in the evaluated tools are given in Table 3.1

For each tool evaluated, 28 different scenarios are defined. Each scenario corresponds to variations in the capacity of the tight link from 1 Mbps to 9 Mbps at 1 Mbps intervals, and from 10 Mbps to 100 Mbps at 5 Mbps intervals. For each scenario, it is considered the situation where the links are completely empty of cross traffic and loaded at 25, 50 and 75 percent of the capacity. MGEN is used to generate Poisson processes with mean rates equal to the given desired amount of cross traffic. It is worth noticing that the cross traffic is generated on a node by node basis, i.e. the traffic generated at node \(i\) loads its output queue and the link from node \(i\) to node \(i+1\), but it does not load the output queue of nodes \(j \neq i\). In this manner no traffic correlations from node to node are included and all nodes are completely independent.

### 3.3.2 Results

The following sections describe the results after performing the experiments and evaluating the estimation error, overhead and estimation time metrics. Each point in the graphs is the average of running each experiment 35 times. That allows in the case of IGI and
Spruce, to calculate and plot a 95% confidence interval calculated by the normal distribution test. In the case of Pathload, it is plotted the average range given by the tool. A total of 11760 experiments are performed: 3 tools, 28 capacity variations, 4 cross traffic loads and 35 samples.

For the analytical results, $\gamma_0$ in Equation 3.1 is the probing traffic rate generated by each tool. This value is the result of dividing the amount of probing packet bytes calculated with `tcpdump` by the estimation time of the tool. This is the reason why the Jackson model behaves differently with each experiment and with each tool.

### 3.3.2.1 Estimation Error

Figures 3.3 to 3.14 present the estimation error of Pathload, IGI and Spruce when the tight link is loaded with 0%, 25%, 50% and 75% of cross traffic. Pathload provides the best approximation to the analytical value obtained from the Jackson’s model. Some Pathload measurements are not shown because the tool has convergence problems in low capacity links. However, as shown in Figure 3.15, when the tool converges, regardless of the amount of cross traffic and tight link capacity, it has a relative error of less than 20%. In most cases the tool overestimates the available bandwidth. It is well known that Pathload is one of the most accurate bandwidth estimation tools [17].

In the case of IGI, this tool presents high estimation errors. For example, in [27], the authors of IGI show that the tool has an error of less than 20% in scenarios with low round trip time values. Experiments in this evaluation also verify this conclusion although the results also indicate very high variability. Figure 3.16 shows that when the cross traffic is high, the accuracy of the tool is very low. This is also mentioned by the authors of IGI
Figure 3.3: Pathload estimation with 0% cross traffic.

Figure 3.4: IGI estimation with 0% cross traffic.

Figure 3.5: Spruce estimation with 0% cross traffic.
Figure 3.6: Pathload estimation with 25% cross traffic.

Figure 3.7: IGI estimation with 25% cross traffic.

Figure 3.8: Spruce estimation with 25% cross traffic.
Figure 3.9: Pathload estimation with 50% cross traffic.

Figure 3.10: IGI estimation with 50% cross traffic.

Figure 3.11: Spruce estimation with 50% cross traffic.
Figure 3.12: Pathload estimation with 75% cross traffic.

Figure 3.13: IGI estimation with 75% cross traffic.

Figure 3.14: Spruce estimation with 75% cross traffic.
Figure 3.15: Pathload relative error for 0%, 25%, 50% and 75% cross traffic.

Figure 3.16: IGI relative error for 0%, 25%, 50% and 75% cross traffic.

Figure 3.17: Spruce relative error for 0%, 25%, 50% and 75% cross traffic.
when they tested links with long round trip times. However, in contrast to the IGI paper, Pathload is still accurate in experiments with highly loaded links. Spruce, on the other hand, shows a relative error smaller than 30% in most scenarios, which also verifies the results presented by the authors in [29]. As in the case of IGI, Spruce also presents problems when estimating over high capacity links with high traffic loads. Its estimation variance is also high over low capacity links. It is worth noticing that IGI and Spruce belong to the same probe gap model category. Spruce shows a particular high under estimation value when the capacity is around 70 Mbps.

3.3.2.2 Overhead

Figure 3.18 to 3.20 show the overhead ratio of the tools for each cross traffic load. The overhead of Pathload does not exceed 10% of the tight link capacity. Pathload introduces more probe traffic when the cross traffic decreases. This is completely expected as it works based on the principle of induced congestion, so the emptier the channel the higher the amount of probe traffic that the tool needs to inject.

In Figure 3.19 it is shown that IGI has low overhead over high congested links. This is because IGI finds several packet trains in the Joint Queuing Region and does not need to send additional packets to determine the turning point. There are, however, some scenarios where the average overhead grows up to 30% or more, such as those points in Figure 3.19 where the capacity of the tight link is 25 Mbps and the cross traffic is 25 or 50% of the capacity.
Figure 3.18: Pathload overhead for 0%, 25%, 50% and 75% cross traffic.

Figure 3.19: IGI overhead for 0%, 25%, 50% and 75% cross traffic.

Figure 3.20: Spruce overhead for 0%, 25%, 50% and 75% cross traffic.
Spruce overhead is low and constant regardless of the amount of cross traffic. This is explained by the Poisson sampling method utilized by the tool. Another observation is the increase of traffic overhead when the tools based on the probe gap model operate in low bandwidth scenarios. In this case, IGI needs to send more probing packets to find the correct probing gap value and Spruce achieves only a small inter-pair gap in the Poisson sampling process, which results in a quite more intrusive sample. This is also reflected in the high estimation error variation shown by these two tools in low link capacities.

In the best case, when the network is highly loaded, the tools need to inject around 3% probe traffic of the narrow link capacity to perform the estimations. Although 3% sounds like a low value, in reality it may be a big number. For instance, in the case of a 50 Mbps narrow link, the tools would occupy 1.5 Mbps. This amount of overhead could limit the utilization of the tools in certain environments, such as wireless networks where link bandwidth is a scarce resource.

### 3.3.2.3 Estimation Time

Figures 3.21 to 3.23 depict the estimation time in seconds when the cross traffic varies from 0% to 75% of the narrow link. From Figure 3.21, it can be seen that Pathload needs less time to converge in the case of 0% cross traffic than in the 75% case. Tcpdump traces provide the explanation for this behavior. When the network is slightly loaded, the tool sends probe traffic more frequently and gets feedback about each sample faster. As a result, it injects more traffic and converges faster.

When the network is highly loaded, the tool needs to increase the gap between probe packets and the gap between trains, which reduces the amount of probe traffic. However,
Figure 3.21: Pathload time for 0%, 25%, 50% and 75% cross traffic.

Figure 3.22: IGI time for 0%, 25%, 50% and 75% cross traffic.

Figure 3.23: Spruce time for 0%, 25%, 50% and 75% cross traffic.
the tool has more problems finding the estimation, which translates into longer estimation times. Pathload can take more than 100 seconds to provide the estimation in some cases. This long estimation time may limit the applicability of Pathload in certain applications or may provide erroneous estimations in those environments with fast changing traffic patterns.

As it is shown in Figure 3.22, IGI needs considerably less amount of time to converge than Pathload. Spruce estimation time is directly associated to the amount of probing packets sent to the network, which is constant regardless the amount of cross traffic.

Regardless of the amount of cross traffic, the evaluated tools have more problems converging when the narrow link is a low capacity link. In the case of IGI and Spruce, their behavior can be explained by the difficulty of the tools to set the appropriate gap, which implies more measurements and more delay in the estimation. In the case of Pathload, the smaller the available bandwidth, the higher the number of iterations the tool needs to perform to detect the gray region. This is also the reason why in some of these points the tool is not able to converge.

3.4 Experimentally-Based Available Bandwidth Evaluation

The evaluation presented in the previous section evidences trade-offs regarded to the performance of the tools in the defined metrics. Specifically, Pathload is the most accurate tool but the slowest to converge. IGI, on the other hand, is the fastest tool but the least accurate. Spruce is the least intrusive tool with intermediate estimation error and estimation time. That evaluation based on the analytical calculation of the real available bandwidth triggers additional questions about the performance of these and other tools.
Table 3.2: Factors and levels in the $2^5$ factorial design. ©2006 IEEE

<table>
<thead>
<tr>
<th>Factor</th>
<th>Level (-1)</th>
<th>Level (+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tight Link Capacity</td>
<td>5 Mbps</td>
<td>100 Mbps</td>
</tr>
<tr>
<td>One Way Propagation Delay</td>
<td>10 ms</td>
<td>80 ms</td>
</tr>
<tr>
<td>Packet Loss Rate</td>
<td>0.01</td>
<td>0.07</td>
</tr>
<tr>
<td>Percentage of cross traffic</td>
<td>25 %</td>
<td>75 %</td>
</tr>
<tr>
<td>cross traffic Packet Size</td>
<td>512 Bytes</td>
<td>1408 Bytes</td>
</tr>
</tbody>
</table>

from the network applications point of view. Moreover, there are questions about how the tools perform in a variety of networks scenarios.

A second evaluation including additional tools, network scenarios, a new metric, and an experimental value of the real available bandwidth is presented in this section. Adding more tools and network scenarios increases the number of experiments that would have been necessary to run in order to evaluate each tool over the totality of network scenarios. For that reason, this evaluation is performed in two phases. In the first phase a $2^{k-p}$ factorial design [43] described in Appendix A, is utilized to reduce that large number of experiments. By using this method, it is possible to establish with enough confidence the most relevant factors and their combinations that affect the performance metrics under evaluation. The second phase utilizes the results of the factorial design to run specific experiments and analyze relevant cases deeper.

3.4.1 Phase One: The $2^5$ Factorial Design

Thirty-two sets of experiments are carried out in this section to perform a $2^5$ factorial design following standard statistical procedures found in the literature [43]. In a $2^5$ factorial design, experiments are performed using two extreme values for each of the five factors in order to collect one sample of each performance metric for each tool.
As many experiments as needed are performed to obtain ten valid results from each evaluation scenario. This is to calculate a 95% confidence interval using the Student’s t-distribution and present statistically confident results. As a result, a minimum of 1600 (32 \times 10 \times 5 \text{ tools}) experiments are run in this phase. Table 3.2 shows the five factors (tight link capacity, link propagation delay, packet loss rate, percentage of tight link capacity used by cross traffic, and cross traffic packet size) and the low and high values selected for each factor according to the methodology.

In this second evaluation, four performance metrics or response variables are tested: the estimation error of the estimation, given by the percentage error of the estimate compared with the real value; the overhead, given by the percentage of the tight link capacity utilized by the probe traffic; the estimation time in seconds; and the reliability of the tool, given by 10 estimations divided by the total number of times the tool had to be run to obtain those 10 values. In all these experiments and unless noted otherwise, each output queue in the path has a buffer size equal to 50 slots, the cross traffic type is Poisson sending 1408-Byte long packets, and all the link capacities are set to 200 Mbps. To obtain the value of the real available bandwidth, every output link in the testbed is sniffed and their packets classified to differentiate probing traffic from cross traffic. This classification is performed by a script written in \textit{Awk} that reads a tcpdump trace and filters the required values.

From the $2^5$ factorial design, the main effects of varying one or several factors on the response variables is determined. Table 3.3 shows the case in which only one factor is varied. It is worth noticing that these results do not say much about the absolute value of the metric. They only provide information about the average variation in a particular metric when the factor value is increased. For example, in the case of Pathload, the effect of increasing the tight link capacity from 5 to 100 Mbps on the estimation error is $-21.86\%$. 

49
Table 3.3: Main effect in the performance metrics when varying one factor.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Response Variables</th>
<th>Error (%)</th>
<th>Overhead (%)</th>
<th>Time (s)</th>
<th>Reliability (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pathload</td>
<td>Tight Link Capacity</td>
<td>-21.86</td>
<td>-3.12</td>
<td>-29.21</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Propagation Delay</td>
<td>-1.62</td>
<td>-1.35</td>
<td>28.97</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Packet Loss Rate</td>
<td>-88.55</td>
<td>-2.22</td>
<td>-31.92</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Percentage of cross traffic</td>
<td>6.06</td>
<td>-0.75</td>
<td>12.27</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>cross traffic Packet Size</td>
<td>4.24</td>
<td>-0.65</td>
<td>8.33</td>
<td>0.00</td>
</tr>
<tr>
<td>IGI</td>
<td>Tight Link Capacity</td>
<td>-80.43</td>
<td>-2.05</td>
<td>-1.74</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Propagation Delay</td>
<td>121.25</td>
<td>-0.23</td>
<td>1.88</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Packet Loss Rate</td>
<td>-58.13</td>
<td>-0.40</td>
<td>5.27</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Percentage of cross traffic</td>
<td>329.38</td>
<td>0.00</td>
<td>3.86</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>cross traffic Packet Size</td>
<td>110.51</td>
<td>0.03</td>
<td>-1.49</td>
<td>0.00</td>
</tr>
<tr>
<td>Spruce</td>
<td>Tight Link Capacity</td>
<td>-25.60</td>
<td>-2.14</td>
<td>-0.28</td>
<td>-7.33</td>
</tr>
<tr>
<td></td>
<td>Propagation Delay</td>
<td>0.80</td>
<td>-0.12</td>
<td>0.61</td>
<td>1.13</td>
</tr>
<tr>
<td></td>
<td>Packet Loss Rate</td>
<td>3.47</td>
<td>-0.29</td>
<td>4.33</td>
<td>-26.12</td>
</tr>
<tr>
<td></td>
<td>Percentage of cross traffic</td>
<td>26.80</td>
<td>-0.02</td>
<td>0.22</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>cross traffic Packet Size</td>
<td>-5.06</td>
<td>0.008</td>
<td>0.50</td>
<td>0.91</td>
</tr>
<tr>
<td>Abing</td>
<td>Tight Link Capacity</td>
<td>-636.00</td>
<td>-14.72</td>
<td>0.00</td>
<td>-1.50</td>
</tr>
<tr>
<td></td>
<td>Propagation Delay</td>
<td>-325.84</td>
<td>-2.04</td>
<td>0.28</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Packet Loss Rate</td>
<td>-64.18</td>
<td>-0.82</td>
<td>-0.02</td>
<td>-38.06</td>
</tr>
<tr>
<td></td>
<td>Percentage of cross traffic</td>
<td>744.01</td>
<td>0.01</td>
<td>-0.01</td>
<td>-3.06</td>
</tr>
<tr>
<td></td>
<td>cross traffic Packet Size</td>
<td>-311.76</td>
<td>-0.04</td>
<td>0.00</td>
<td>0.95</td>
</tr>
<tr>
<td>Pathchirp</td>
<td>Tight Link Capacity</td>
<td>-283.64</td>
<td>-4.72</td>
<td>0.16</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Propagation Delay</td>
<td>11.11</td>
<td>0.06</td>
<td>-0.03</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Packet Loss Rate</td>
<td>-111.77</td>
<td>-0.62</td>
<td>-0.87</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Percentage of cross traffic</td>
<td>229.73</td>
<td>-0.18</td>
<td>-1.33</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>cross traffic Packet Size</td>
<td>-40.67</td>
<td>0.09</td>
<td>1.66</td>
<td>0.00</td>
</tr>
</tbody>
</table>
This result means that Pathload negatively varies its estimation error downwards with an average 21.86% as the capacity of the tight link increases. However, this result does not mean that the estimation error improves or declines by 21.86%. Figure 3.24 summarizes the main effect of varying each factor on each response variable for each of the tools as having some impact, medium impact, and high impact. In addition, it is also calculated the average interaction of two, three, four, and five factors, which are shown in Tables A.2, A.3, A.4, and A.5.

From Figure 3.24, the following main conclusions can be made:

- The variation of any of the factors considerably impacts the estimation error variation for IGI, Abing, and Pathchirp. This is an indication of the inaccuracy of these tools. Therefore, the applicability of these tools may be limited to those applications that do not need to have very precise estimations. Spruce and Pathload are the least affected tools in the sense that their estimation errors do not vary much, independently of whether the estimates are accurate or not. The accuracy of Pathload is
mainly affected by changes in the packet loss rate and the capacity of the tight link. Therefore, this tool might not be good choice for those applications running over wireless networks, which usually have higher packet loss rates and low bandwidth channels. Spruce’s accuracy is affected by the amount of cross traffic (congestion) and the capacity of the tight link. Additional experiments need to be made to better analyze the estimation error of these tools. It is worth mentioning that available bandwidth estimation tools have not been analyzed under different packet loss rates before.

- The overhead that the tools insert into the network to perform the estimation is almost unaffected by the variation of any of the factors, except for Abing, which seems to be affected by the capacity of the tight link. Therefore, only experiments with variations in the capacity of the tight link are performed to observe the overhead shown by the tools in more detail. This conclusion does not mean that the tools do not insert a considerable amount of overhead but that the overhead that they introduce is fairly constant regardless of the factors.

- With the exception of Pathload, the estimation time of the tools is barely affected by any of the factors, meaning that the tools take a similar amount of time to converge. This is a good feature, as it provides predictability in the estimation time. The estimation time of Pathload is shown to be affected by the capacity of the tight link, the propagation delay, the packet loss rate, and the amount of cross traffic. Given these results, the applicability of Pathload may be limited to certain non-real time applications and certain network types. Again, it may not be a good idea to use Pathload in wireless and satellite networks, which have higher packet loss rates, low bandwidth, and long propagation delay links.
• The packet loss rate is the only factor that affects the reliability of the tools, and only affects Spruce and Abing. More experiments are needed to determine the reliability level of the tools and the packet loss rate at which the reliability becomes critical.

Similarly, the average effect that the combination of two, three, four, and all five factors have in the performance metrics is analyzed (results are shown in Appendix A). Here are the main observations from that analysis:

• The estimation error of IGI, Abing, and Pathchirp is very sensitive to variations in the capacity of the tight link and the amount of cross traffic in the network. More specifically, when these two factors are increased, a negative variation in the estimation error is obtained. A similar trend is observed when the packet loss rate and the amount of cross traffic are increased.

• When the capacity of the tight link and the cross traffic packet size increase, Pathload and Spruce tend to underestimate the available bandwidth.

• The estimation time of Pathload is highly affected by variations in the capacity of the tight link, the packet loss rate, and the amount of cross traffic.

• When all factors are combined, IGI’s estimation error is the most affected but its overhead is the most stable. The estimation time of Abing shows the most stable behavior.
3.4.2 Phase Two: Main Experiments

The second phase includes the results of the additional experiments performed according to the results of the factorial design. Looking at Figure 3.24, it can be seen that the capacity of the tight link affects the estimation error, overhead, and estimation time of the tools. As a result, the first set of experiments will look at these metrics while varying the capacity of the tight link from 10 to 200 Mbps, and setting a packet loss rate of 1%, a one way propagation delay of 10 ms, and a low congested (20% of cross traffic) and highly congested (75%) tight link. These results should provide some guidance as to which tools are better suited for low, medium, or high bandwidth networks.

3.4.3 Results

The following sections describe the results of performing the significant experiments and collecting data about the estimation error, overhead, estimation time, and reliability of the evaluated tools. Each point in the graphs is the average of running each experiment 10 times which provides statistically more significant results when confidence intervals are calculated using the Student’s t-distribution.

3.4.3.1 Variable Tight Link Capacity

Figures 3.25 to 3.30 show that, in general and for all the tools, the estimation error goes down with the capacity of the tight link. For the low congested scenario, the capacity of the tight link does not seem to have much effect on the accuracy of the tools, except in the
case of IGI in scenarios with low bandwidth links, and Spruce, which presents estimation problems beyond 100 Mbps. Spruce’s behavior in high capacity links was not evidenced in Figure 3.24 because the high level in the factorial design was selected to be 100 Mbps. Figures 3.25 and 3.26 show that the accuracy of the tools is mostly affected by the level of network congestion. It can be seen that the tools provide fairly stable (low variance) estimations in low congested scenarios regardless of the capacity of the tight link while they present highly variable estimations in highly congested scenarios.

Figure 3.25: Estimation error at 20% cross traffic with variable capacity, 10 ms OWD, 1% PLR.

Figure 3.26: Estimation error at 75% cross traffic with variable capacity, 10 ms OWD, 1% PLR.
Regardless of the congestion level, Pathload and Spruce (in that order) are shown to be the most accurate tools, with an estimation error of less than 25%. Pathload tends to over-estimate the available bandwidth in low congested scenarios and underestimate the bandwidth in highly congested ones. Spruce, as mentioned before, presents estimation problems when the link capacity goes beyond 100 Mbps. IGI and Abing, on the other hand, are shown to be highly inaccurate, especially in highly congested low capacity links (with estimation errors higher than 100%). Finally, Pathchirp is fairly accurate in low congested scenarios while it presents high estimation errors in highly congested ones, especially when low capacity links are used. Pathchirp estimation errors are usually less than 50% except in this case where the error can be as high as 200%.

As it is observed in Figures 3.27 and 3.28, the capacity of the tight link does not seem to have a strong impact on the convergence time of the tools. Only Pathload, under highly congested scenarios seems to be affected, increasing the convergence time with the capacity. Other than that, all tools present fairly low variability as the capacity of the tight link is varied. It can be seen that the convergence time of the tools also shows a well known trend. Pathload is the slowest tool to converge, followed by Pathchirp, Spruce,
Figure 3.28: Convergence time at 75% cross traffic with variable capacity, 10 ms OWD, 1% PLR.

IGI, and Abing in that order. Another important aspect is that the convergence time of the tools does not seem to be affected by the level of congestion in an appreciable way. Regardless of the congestion level, Spruce, Pathchirp, and Abing show low variation and lower convergence time than Pathload. Pathchirp gives a mean convergence time of around 13 seconds. Spruce’s convergence time is around 12 seconds while Abing is the fastest tool with an almost constant 1 second convergence time. IGI is the second fastest tool but its convergence time depends on the congestion level; in low congested scenarios IGI’s convergence time is around 4 seconds while in high congested cases it may reach 15 seconds and presents high variability.

With regard to the overhead, Figures 3.29 and 3.30 show that as the capacity of the tight link is increased, the overhead decreases. This is a clear indication that the overhead introduced by the tools is rather constant, i.e., the tools need a similar amount of probe packets to make the estimation regardless of the level of congestion and capacity of the networks.
From the results, it is clear that Pathload and Abing are the most intrusive tools. The amount of probe packets in Pathload takes approximately 6% of the tight link capacity in low congested scenarios and 1.5% in highly loaded networks. Pathload is less intrusive in highly congested scenarios because it works filling the available pipe capacity. However, as the capacity of the tight link is increased the amount of overhead needed by the tool decreases, i.e., the tool is able to find the available bandwidth more efficiently when the available capacity is big. The other tool that uses the same induced congestion principle is Pathchirp. However, as it is argued in [32], Pathchirp needs less than 10% of the probing traffic that Pathload uses. Pathchirp, sends the probe packets using a different approach.
Figure 3.31: Estimation error at 5 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

Figure 3.32: Estimation error at 100 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

gap distribution and algorithm that makes the tool to converge faster and, therefore, sends fewer probe packets. On the other hand, Abing utilizes a fixed and rather large amount of overhead to perform the estimations. This is why the overhead decreases with the capacity of the tight link and the level of congestion. IGI and Spruce are the tools that introduce the least amount of overhead (less than 0.5%). The results for this first set of experiments confirm what is already known about the accuracy of the tools from different authors [32], [27], [29], and [18].
3.4.3.2 Variable One-Way Propagation Delay

The second set of experiments look at the effect of the one-way propagation delay, which is varied from 10 to 80 msecs. This is another scenario that has not been studied before and should say whether current available bandwidth estimation techniques are appropriate or not in networks with long delay links, such as networks with satellite connections.

From Figure 3.24, it can be seen that variations of the one-way propagation delay only affect the estimation error and the convergence time of IGI and Abing, and Pathload, respectively. Overhead plots are added due to the particular situation of Abing, which shows a significant overhead (up to 20%) when the tight link capacity is low (5 Mbps). Experiments are performed with low (5 Mbps) and high (100 Mbps) tight link capacities and with an end-to-end packet loss rate of 1%. Only the experiment results related to highly congested scenarios (75% of cross traffic) are included because they are the most challenging scenarios to the tools.

With regard to the estimation error, when the tight link is set at 5 Mbps (Figure 3.31), Abing’s error is extremely high (up to 3000%) and increases with the one-way propa-
Figure 3.34: Estimation time at 100 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.

Pathchirp also has a very high estimation error (up to 500%). (This is why these results are not included in the graph.) Spruce and Pathload are the most accurate tools in this scenario with a 50% estimation error followed by IGI with an estimation error in the order of 150%.

Although the increment in the one-way delay affects the accuracy of the tools compared to the results obtained in Figure 3.26, most of the tools seem to be unaffected by additional increments. In general, all tools present overestimation problems in this particular scenario. When the tight link is set to 100 Mbps, Figure 3.32 shows that the estimation error of the tools, with the exemption of IGI, improves compared with the 5 Mbps scenario, but the one-way propagation delay seems to have no major impact either. In general, only Pathload and Spruce seem to be adequate in these cases, as the estimation error of the rest of the tools is very high. Here, Spruce is even more accurate than Pathload with an estimation error of less than 15%. (Notice that the results of Spruce and Pathload are very consistent with the ones shown in Figure 3.26 at 100 Mbps.)

Regarding the estimation time, the only problematic tool is Pathload. When the tight link is set to 5 Mbps (Figure 3.33), Pathload presents the worst estimation time, taking more
than 100 seconds to converge compared with less than 15 seconds for the rest of the tools. When the tight link is set to 100 Mbps, Figure 3.34 shows that, although Pathload improves its estimation time, it still is the worst performing tool. This last Figure shows how Pathload’s estimation time increases with the one-way propagation delay from 20 seconds to around 60 seconds. This behavior is expected since the tool adjusts the transmission rate according to the one-way delay variation shown by a probing packet train. Thus, the longer the propagation delay is, the slower the tool reacts. The other tools present steady estimation times regardless of the capacity of the tight link and the one-way propagation delay. Abing is definitively the fastest tool with an average 1 second estimation time while IGI, Pathchirp, and Spruce present estimation times of around 12 seconds or lower.

Figures 3.35 and 3.36 show that the propagation delay has no major impact in the overhead of the tools, i.e. bigger propagation delays do not translate into more probe packets. In the case of a tight link of 100 Mbps all tools experience an overhead below 1%; however, when the tight link has a capacity of 5 Mbps this value can reach up to 6% in some tools. The exemption in this last case is Abing, with an overhead of as high as 20%. This behavior of Abing can be easily explained. Abing sends the same high amount of fixed

Figure 3.35: Overhead at 5 Mbps for variable one way delay, 1% PLR, and 75% cross traffic.
size probing packets regardless to the path capacity, therefore, in low capacity links, it is expected to be more intrusive. It is worth noticing that the percentage values presented thus far do not indicate that the tools introduce low overhead. Notice that even a 1% of overhead in the case of the 100 Mbps tight link capacity, means that the tool introduces an overhead of 1 Mbps.
Figure 3.38: Estimation error at 100 Mbps for variable packet loss rate, 10 ms delay, and 75% cross traffic.

3.4.3.3 Variable Packet Loss Rates

The third set of experiments are meant to study the performance of the tools in scenarios with different packet loss rates. These experiments are important to conclude about the utilization of current available bandwidth estimation tools in networks with lossy links, such as wireless networks, and to study the reliability of the tools. In this case, experiments are performed with low (5 Mbps) and high (100 Mbps) tight link capacities with 75% of cross traffic and with one-way propagation delay of 10 ms while increasing the packet loss rate from 1% to 10%. Since Figure 3.24 shows that the estimation error, the estimation time, and the reliability of the tools are the response variables affected by the packet loss rate, these will be the only plots included in this part. With regard to the estimation error, when the tight link is set to 5 Mbps (Figure 3.37), Pathchirp and Abing go out of any normal range and, therefore, are not included in the Figure.

Pathload presents a steady behavior with an underestimation of 25% for packet error rates of 3% and higher. Spruce, on the other hand, presents a steady and accurate estimation regardless of the packet loss rate. When the tight link is set to 100 Mbps (Figure 3.38),
the tools present different behaviors. For example, the estimation error is above 100% for Abing and IGI while it is below −75% (underestimation) for Pathchirp and Pathload, especially for packet loss rates above 4-5%. Further, in the case of Pathload, the estimation error is even worse (close to 100%).

With regard to the estimation time, the results using tight links of 5 and 100 Mbps, depicted in Figures 3.39 and 3.40, show that the behavior of the tools is similar to previous scenarios, with Pathload taking the longest time. For the 5 Mbps link case, Pathload goes
from 131 seconds to 46 seconds when the packet loss rates go from 1% to 10%. In this case, as the packet loss rate increases, Pathload provides faster but worse estimates. As before, Spruce and Pathchirp each takes around 12 seconds to converge for both tight link capacities. On the other hand, IGI’s estimation time tends to increase with the packet loss rate and is higher than Spruce’s and IGI’s in the 5 Mbps case. As expected, Abing is the fastest tool with steady estimation times even lower than 1 second.
As shown in Figure 3.24, the reliability of Abing and Spruce is affected only when the packet loss rate is increased. The results of the reliability experiments are shown in Figures 3.41 and 3.42 for the 5 and 10 Mbps tight link capacity cases. From the two plots, it is clear that Abing and Spruce are the least reliable tools. Spruce seems to have problems providing an estimate with packet loss rates beyond 6% while Abing has problems regardless of the loss rate.

3.4.3.4 Variable Amount of Cross Traffic

The next factor varied is the amount of cross traffic in the tight link. Again, the experiments performed included low (5 Mbps) and high (100 Mbps) tight link capacities with a one-way propagation delay of 10 ms and a packet loss rate of 1%, but this time the amount of cross traffic (congestion) is varied in the tight link from 10% to 80%. According to Figure 3.24, variations in the amount of cross traffic only affect the estimation error and the estimation time of the tools.

However, additional plots are also included for the overhead (Figures 3.47 and 3.48), which again shows the same high overhead value of Abing in the 5 Mbps case, similar to the one shown in Figure 3.35. Figures 3.43 and 3.44 show the estimation error.

It is clear that regardless of the capacity of the tight link, the estimation error of the tools increases with the amount of cross traffic and with a tendency to overestimate the real available bandwidth. It is also clear that the tools have more difficulty estimating the available bandwidth accurately when the tight link capacity is 5 Mbps. In this case, it can be seen that Spruce and Pathload are the best tools, in that order. The rest of the tools present very inaccurate estimates in many cases. In particular, as is also presented in [17],

67
Figure 3.43: Estimation error at 5 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR.

Figure 3.44: Estimation error at 100 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR.

Abing’s accuracy shows also problems when the available bandwidth drops below 60% of the tight link capacity.

With regard to the estimation time, the results are similar to the ones obtained in past experiments. Pathload again is shown to have problems in the case of the 5 Mbps tight link, especially in the medium to high congestion range. As it is shown in [17], Pathload takes about 20 seconds to converge when the capacity of the tight link is high. The rest of the tools present stable behaviors regardless of the capacity of the tight link and the amount of cross traffic, and very consistent with past results. Again, Abing is the fastest tool.
Figure 3.45: Estimation time at 5 Mbps for variable % of cross traffic, 10 ms OWD, and 1% PLR.

followed by IGI and then Spruce and Pathchirp with estimation times of approximately 1, 6, 12, and 13 seconds, respectively.

The overhead introduced by the tools is shown in Figures 3.47 and 3.48. As explained before, the behavior of Pathload is the expected one: the overhead reduces with the amount of congestion. The other tools introduce the same amount of overhead regardless of the amount of congestion. However, the overhead can be significant if, as in the case of Abing, the tool inserts a constant but high amount of probing packets into the network. For exam-
In the case of the 5 Mbps tight link capacity, Figure 3.47 shows that Abing’s overhead is around 19.4% of the tight link capacity, or 950 kbps out of a 5 Mbps link.

3.4.3.5 Variable Cross Traffic Packet Size

Finally, the effect in the estimation error when varying the cross traffic packet size is studied. Here two different congestion levels (20% and 75% of cross traffic) and tight link
Figure 3.49: Estimation error at 5 Mbps and 20% cross traffic for variable cross traffic packet size, 10 ms OWD, and 1% PLR.

Figure 3.50: Estimation error at 5 Mbps and 75% cross traffic for variable cross traffic packet size, 10 ms OWD, and 1% PLR.

Figure 3.51: Estimation error at 100 Mbps and 20% cross traffic for variable cross traffic packet size, 10 ms OWD, and 1% PLR.
capacities (5 and 100 Mbps) are studied. The one-way propagation delay is of 10 ms and the packet loss rate of 1%. By looking at Figures 3.49 to 3.52, it can be seen that the cross traffic packet size has no major effect on the estimation error of the tools, which is fairly constant regardless of the packet size.

It is also clear that the error is smaller and more stable in low congested scenarios than in highly congested ones, which is also consistent with past results. Also, it can be seen that Pathchirp and Abing present estimation problems in the case of low capacity and congested tight links. This is also very consistent with other results presented so far, which lead us to conclude that neither Pathchirp nor Abing are good choices for low bandwidth and highly congested channels.

3.5 Applicability of Current Available Bandwidth Estimation Tools

This section presents general and specific conclusions to answer the original questions regarding the applicability of current available bandwidth estimation tools. The first general conclusion, and perhaps the most important one, is that current tools are still far from
having good performance for many, if not most, of the envisioned applications and networking environments. For example, the tool with the best estimation time is Abing, which provides estimations in around 1 second. Simply put, a estimation time of 1 second may leave out many real-time applications that would need to have estimates in the order of milliseconds. One example of this case may be the flow control mechanism of an available bandwidth-based transport layer protocol.

Second, none of the existing tools introduce low enough overhead to be used in those applications that will work on a per-connection basis. The example of the transport layer protocol comes to mind again; if TCP or any substitute transport layer protocol were to use an available bandwidth tool for flow control, the amount of overhead has to be considered given the large number of connections that currently go through the Internet. For these type of applications, new methods have to be devised to use the data packets as probe packets, reducing the overhead to acceptable levels.

Third, tools based on the PGM are usually less intrusive but less accurate than the ones based on the PRM. Fourth, all tools present their worst behavior and performance in highly loaded scenarios.

Finally, current available bandwidth tools are mostly “network-unaware” and accuracy problems may occur because the mechanisms of the underlying networking technology are not being modeled or included in the methodologies and tools. For example, there are well known medium access protocols and back-off algorithms in wireless networks that may introduce errors. More specific conclusions, drawn from the results of the performance evaluation, are as follows:

- Most of the tools are barely affected by the capacity of the tight link. Only IGI and Spruce present accuracy problems - IGI at low link capacities and Spruce be-
yond 100 Mbps. If the capacity of the tight link and the level of congestion are combined, the results show that Pathload and Spruce are the most accurate tools. These results indicate that they may be the best candidates for low, medium, and high capacity networks. On the other hand, IGI, Abing, and Pathchirp are highly inaccurate, especially in highly congested scenarios. Except for Pathload in highly congested scenarios, the estimation time of the tools is not impacted by variations in the capacity of the tight link. Pathload is the slowest tool to converge followed by Pathchirp, Spruce, IGI, and Abing, in that order. Abing provides an almost constant estimation time of approximately one second. In terms of overhead, Pathload and Abing are the most intrusive tools; therefore, these tools are not the best choices for low capacity and wireless networks. Pathload’s overhead varies and depends on the capacity and level of congestion of the network. This is due to the fact that Pathload utilizes the principle of induced congestion. Abing’s overhead is fixed but large. Spruce, Pathchirp, and IGI are the least intrusive tools.

- Current available bandwidth estimation tools behave differently to different end-to-end propagation delays. In general, further increases in propagation delays have no major impact on the estimation error of the tools. Pathchirp, Abing, and IGI present very high estimation errors regardless of the capacity of the network. On the other hand, Pathload and Spruce are barely affected by the propagation delay; both tools provide estimates with errors below 50% and low variability. With regard to the estimation time, the results show that with the exemption of Pathload, the tools seem to be unaffected by increases in the propagation delay. While most tools present estimation times below 20 seconds and without variation, Pathload has high and increasing estimation times. Finally, the propagation delay does not have a major impact on the overhead of the tools. Comparing all the tools, it can be
concluded that Spruce is the best tool for long propagation delay scenarios, as it presents the best combination of results: Spruce is the most accurate, one of the fastest, and one with the lowest overhead. Spruce might be a good choice for those applications running over satellite links or transatlantic fiber optic links. However, its performance in scenarios with links of more than 100 Mbps is still in question, given the results shown in Figure 3.26. In those cases, Pathload is the best second tool, if the high estimation time is not an issue.

- Spruce is the most accurate and immune tool to variations in the packet loss rate. The rest of the tools present high estimation errors as the packet loss rate is increased. The estimation time results of the tools are also similar to past results in terms of variation and absolute values, with Pathload presenting the longest estimation time followed by IGI, Pathchirp, Spruce, and Abing, in that order. Spruce’s estimation time does not seem to be affected by the packet loss rate since it stays at around 12 seconds. This estimation time is also seen in the other experiments, making Spruce a very predictable tool in this regard. The only problem is that in scenarios with high congestion and packet loss rates higher than 6%, Spruce fails to provide an estimate in many cases. Abing, although faster than Spruce, showed to be even more unreliable. These results suggest that Spruce is the best tool for networks with varying and high packet loss rates, such as wireless networks.

- In the analysis of the tools under different congestion levels, it is clear that regardless of the capacity of the tight link, the estimation error of the tools increases with the amount of cross traffic and with a tendency to overestimate the real available bandwidth. It is also clear that the tools have more difficulty estimating the available bandwidth accurately when the capacity of the tight link is 5 Mbps. In this case, it can be seen that Spruce and Pathload are the best tools, in that order. The
rest of the tools provide very inaccurate estimates in many cases. With regard to the estimation time and overhead, the results are similar to the ones obtained in past experiments. As a result, it can be concluded that Spruce and Pathload are the best choices for congested and low capacity networks, such as DSL and cable modem access networks.

- Results show that the size of the cross traffic packet has no major effect on the estimation error of the tools. The estimation error is fairly constant and consistent with past results. It is also clear that the error is smaller and more stable in low congested scenarios than in highly congested ones, which is also consistent with past results.

- Although previous conclusions are based on the average behavior of the tools under different scenarios, there are isolated results specially in the estimation error evaluation which are far from that average behavior. For example, IGI showed a very low estimation error (around 0.5%) when the link is set to 5 Mbps, 10 ms propagation delay, 1% packet loss rate, 20% of cross traffic in the tight link, and any cross traffic packet size. Therefore, any conclusion for a particular network scenario must be drawn from the detailed observation of Figures 3.25 through 3.52.

Table 3.4 summarizes all the results provided thus far and shows the best/worst performing tool for each of the factors and response variables studied. In terms of the applicability of the current available bandwidth estimation tools in the envisioned applications, Table 3.5 includes a qualitative assessment of their requirements and the best tools for each application. The assessment is qualified as low, medium, or high if the application needs low, medium, or high estimation error, overhead, estimation time, or reliability. From the Table, it can be seen that despite the current efforts in the design and development of
Table 3.4: Summary of best/worst performing tools.

<table>
<thead>
<tr>
<th>Main Factor</th>
<th>Sec. Factor</th>
<th>Level</th>
<th>Best/worst tools</th>
<th>Networking Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>T.Link</td>
<td>Low/High</td>
<td>XT</td>
<td>Pchirp-Abg/IGI</td>
<td>All types of wired networks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20%</td>
<td>IGI-Pload/Spr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>XT</td>
<td>Pload-Spr/IGI</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>75%</td>
<td>Pload/IGI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Pload</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Pload</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Abg</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Abg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XT:75%</td>
<td></td>
</tr>
<tr>
<td>OWD</td>
<td>Low/High</td>
<td>Cap.</td>
<td>Pload-Spr/Abg</td>
<td>LAN, WAN, satellite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 M</td>
<td>Pload-Spr/Abg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abg/Pload</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abg/Pload</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Pload/IGI</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Abg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Abg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>High speed</td>
<td></td>
</tr>
<tr>
<td>PLR</td>
<td>Low/High</td>
<td>Cap.</td>
<td>Pload-Spr/Abg</td>
<td>LAN and WAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 M</td>
<td>Pload-Spr/Abg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abg/Pload</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abg/Pload</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/IGI</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/IGI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IGI/Pload</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IGI/Pload</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IGI/Abg</td>
<td>All/Abg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IGI/Abg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IGI/Abg</td>
<td></td>
</tr>
<tr>
<td>XT Amount</td>
<td>Low/High</td>
<td>Cap.</td>
<td>Spr/Pchirp</td>
<td>DSL, cable access networks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 M</td>
<td>Spr/Abg-Pchirp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abg/Pload</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abg/Pload</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Pload/IGI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Abg</td>
<td>All 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Spr/Abg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XT:75%</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.5: Qualitative assessment of application requirements and best tools for the set of representative applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>Accuracy</th>
<th>Overhead</th>
<th>Time</th>
<th>Reliability</th>
<th>Best Tool</th>
<th>Main Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLA Compliance</td>
<td>High</td>
<td>Medium</td>
<td>Med/Hig</td>
<td>Medium</td>
<td>Spr/Pload</td>
<td>Overhead</td>
</tr>
<tr>
<td>N. Management</td>
<td>Medium</td>
<td>Medium</td>
<td>Med/Hig</td>
<td>Low/Med</td>
<td>Spr/Pload</td>
<td>Overhead</td>
</tr>
<tr>
<td>Traffic Eng.</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>None yet</td>
<td>Overhead and time</td>
</tr>
<tr>
<td>Flow Control</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>None yet</td>
<td>Overhead and time</td>
</tr>
<tr>
<td>Security</td>
<td>Low/Med</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>None yet</td>
<td>Estimation time</td>
</tr>
<tr>
<td>Admiss. Control</td>
<td>Med/Hig</td>
<td>Low/Med</td>
<td>Low</td>
<td>High</td>
<td>None yet</td>
<td>Overhead and time</td>
</tr>
</tbody>
</table>

new methodologies and tools to estimate the end-to-end available bandwidth, most of the tools are not entirely suitable for most of the envisioned applications. Current available bandwidth estimation tools only find applicability in very relaxed scenarios where the overhead and the estimation time of the tools are not big issues.
Chapter 4: HMM Approach to Available Bandwidth Estimation

Hidden Markov models (HMM) has been traditionally applied to speech, handwriting and gesture recognition, weather prediction, and in Bioinformatics to model DNA and protein sequences. These are problems where few and noisy information is used to estimate a state of the modeled system. The available bandwidth estimation problem also requires as few samples as possible so that the network is not congested with additional probing traffic. In addition, as it is shown in Section 1.2, there are several issues that affect the accuracy of each measurement obtained form every sample of the network. Building a hidden Markov model of the available bandwidth provides a mechanism to adjust the erratic measurements (not to avoid them).

Although this is the first work that applies HMM in the available bandwidth estimation problem, there are previous studies that use HMM’s in networking. In [44], a packet-level traffic HMM is used to model traffic sources and estimate Packet Size and Inter Packet Time. The model is validated using Internet traces with SMTP and HTTP traffic. A similar study performed in [45] builds HMM profiles to classify network applications from Internet traffic traces. Some of the applications studied are FTP, SMTP, HTTP, and Telnet. Another study [46] uses a HMM to model fading communication channels and to find closed-form solutions for the probability distribution of fade duration and number of level crossings.
This chapter introduces the concept and elements of a hidden Markov model, describes how the model is applied to the available bandwidth estimation problem, and presents the algorithms used to infer the available bandwidth state from well defined observations of the system.

### 4.1 Discrete Hidden Markov Models

A discrete Markov model is represented by a set of $N$ distinct states where the state at time $t$ is denoted by $q_t$ and the probability of going from one state to another depends on the values of the previous states [47]:

$$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, \ldots). \quad (4.1)$$

In a first-order Markov model, the state at time $t + 1$ depends only on the state at time $t$. That is,

$$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, \ldots) = P(q_{t+1} = S_j | q_t = S_i) \quad (4.2)$$

In an observable Markov model, at any time $t$, $q_t$ is known. Therefore, the sequence of states when the system moves from one to another is known. In a hidden Markov model, the state is not observable (it is hidden) but can be inferred from a given observation that is a probabilistic function of the state.
4.1.1 HMM Elements

As described in [48], a hidden Markov model is composed by the following five elements:

- N states in the model. This is a finite number that represents the number of states. The state at time $t$ is denoted as $q_t$. The set of states is denoted by:

$$S = S_1, S_2, \ldots, S_N$$  \hspace{1cm} (4.3)

- M distinct observation symbols per state. These are all the possible outcomes of a state. The set of observation symbols is denoted by:

$$V = v_1, v_2, \ldots, v_M$$  \hspace{1cm} (4.4)

- State transition probability matrix (A). Each element of this matrix has the probability of transitioning from one state to another. The sum of each row in the matrix has to be one. This matrix is denoted by:

$$A = [a_{ij}] \text{ where } a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \text{ for } 1 \leq i, j \leq N$$  \hspace{1cm} (4.5)

- Observation probabilities (B). This is a set of probabilities that indicates how likely is that at time $t$ an observation $O_t$ is generated by each state from the set $S$. This set defined for each state $1 \leq j \leq N$ is denoted by:

$$B = [b_j(m)] \text{ where } b_j(m) = P(O_t = v_m | q_t = S_j) \text{ for } 1 \leq m \leq M$$  \hspace{1cm} (4.6)
Figure 4.1: Available bandwidth Markov model. States represent levels of bandwidth availability.

- Initial state probabilities ($\Pi$). This is a vector with the probabilities that each state is the first in the state sequence that generated the observations:

$$\Pi = [\pi_i] \text{ where } \pi_i = P(q_1 = S_i) \text{ for } 1 \leq i \leq N \quad (4.7)$$

The last three set of probabilities are usually denoted as $\lambda=(A,B,\Pi)$ to indicate the complete parameter set of the model.

### 4.2 HMM to Estimate Available Bandwidth

The available bandwidth in an end-to-end path can be modeled by $N$ states, each one representing certain level of availability. For example, in the five-state representation shown in Figure 4.1, the available bandwidth could be in one of Low (L), Medium Low (ML), Medium (M), Medium High (MH), and High (H) states. That is, it could be located in any spare utilization range from [0,0.2), [0.2, 0.4), [0.4,0.6), [0.6,0.8), or [0.8,1]. By sampling the available bandwidth during time $T$, the sequence of states visited during that period can be determined. Then, the average state visited during $T$, calculated as the average of the middle points of each state range, is an estimation of the available bandwidth during that period of time.
Since the average timescale in Equation 1.2 is very small (milliseconds), it is assumed that transitions from one state to another during that period go no farther than one state apart. Therefore, the Markov chain representing the available bandwidth process, as it is shown in Figure 4.1, has a one-step transition probability matrix determining movements between available bandwidth levels.

However, available bandwidth states can not be directly observed, they are hidden, since the end-to-end estimators do not have information about bandwidth consumption in intermediate routers. Rather, available bandwidth estimators sample the network path with probing packets that convey packet dispersion information, which can be used by a hidden Markov model to infer the non-observable states.

### 4.2.1 Probing Sampling Method

In order to get information about the available bandwidth dynamics during the period $T$, the network is sampled using the probe gap model (see Section 2.1.1). Assuming the single tight link model shown in Figure 1.4, a probing packet pair enters the router with a $\Delta_{in}$ separation. Then, due to the interaction of the probing packet pairs with the cross traffic in the router’s output queue, they leave the link with a different separation, or dispersion. It is well known that this variation has a strong correlation with the amount of cross traffic in the queue during the sampling period, which can be used to estimate the available bandwidth. The relative dispersion suffered by the probing packets can be defined as:

$$
\varepsilon_t = \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}, \quad t = 1 \ldots T
$$

\[ (4.8) \]
which is a measure of the tight link utilization as seen by a probing packet pair at time \( t \). Then, knowing the capacity of the tight link \( C_t \), the end-to-end available bandwidth at time \( t \) can be estimated by:

\[
\overline{A_t} = C_t \times (1 - \varepsilon_t) = C_t \times \left( 1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}} \right)
\]  \hspace{1cm} (4.9)

Similar to other PGM-based tools, the value of the tight link capacity (\( C_t \)) can be calculated using well known and accurate tools, like Pathrate [38].

### 4.2.2 Model Description

Since in the available bandwidth model proposed in Figure 4.1 the states can not be directly observed, a HMM approach can be used to find the state sequence associated with the dispersions observed during the sampling period. The model, which is shown in Figure 4.2, is a hidden Markov model with discrete hidden states \( q \) representing the available bandwidth levels (ranges) and discrete observation variables \( \xi \) representing probing packet pair dispersions. A particular observation has associated a probability \( B \) of being generated by a particular hidden state. Available bandwidth transitions go from time \( t = 1 \) to time \( t = T \). Transitions between states are determined by probabilities specified in the transition probability matrix \( A \).

This model, which is refined with every new observation, is used to determine the most probable state sequence \( (Q = q_1, q_2, \cdots, q_T) \) responsible for what has been observed during \( T \). At the end, the average state in the estimated sequence of states will be the average available bandwidth during time period \( T \).
According to the components of a hidden Markov model described in Section 4.1.1, the available bandwidth estimation model has the following five elements:

- **Number of states in the model (N).**
  The bigger the number of states $N$, the longer the time needed to provide an estimation. The set of states is defined by $S = \{S_1, S_2, \ldots, S_N\}$ where the available bandwidth level grows from $S_1$ (low) to $S_N$ (high). The state at time $t$ is denoted by $q_t$. The default number of possible states in the estimation tool developed in this work is ten, representing available bandwidth ranges of $[0,0.1)$, $[0.1,0.2)$, ..., and $[0.9,1]$.

- **Number of distinct observation symbols per state (M).**
  These are all the possible outcomes of a state. That is, the set of symbols associated with observed dispersions from the probing sampling method. The default number of distinct observation symbols in the estimation tool presented in this work is ten, and it is denoted by $V = \{v_1, v_2, \ldots, v_{10}\}$. Each symbol is a decimal number from 1 to 10. These ten symbols represent observed relative dispersion values $\varepsilon$ in the ranges $v_1 \equiv [0,0.1)$, $v_2 \equiv [0.1,0.2)$, ..., and $v_{10} \equiv [0.9,1]$. Therefore, every single observation at time $t$ has to be converted to a discrete value $\xi_t$ associated with a symbol $v$ by:
\[ \xi_t = [M \times |1 - \varepsilon_t|] \]  

(4.10)

That is, Equation 4.10 determines the discrete observation value \( \xi_t \) that corresponds to a continuous observation \( \varepsilon_t \), where \( \xi_t \in V \). There is also a relation between states (available bandwidth) and observations (associated with \( \varepsilon \)) as it is defined in Equation 4.9.

- State transition probability matrix \( (A) \).

\[ A = [a_{ij}] \text{ where } a_{ij} = P(X_{t+1} = S_j | X_t = S_i), \ 1 \leq i, j \leq N. \]  

Since only one-step transitions between states are possible, the number of unknown elements in the matrix is reduced to the three main diagonals:

\[
A = \begin{bmatrix}
  a_{1,1} & a_{1,2} & 0 & \cdots & 0 \\
  a_{2,1} & a_{2,2} & a_{2,3} & 0 & \vdots \\
  0 & \ddots & \ddots & \ddots & 0 \\
  \vdots & 0 & a_{N-1,N-2} & a_{N-1,N-1} & a_{N-1,N} \\
  0 & \cdots & 0 & a_{N,N-1} & a_{N,N}
\end{bmatrix} \]  

(4.11)

- Observation probabilities \( (B) \).

As explained before, this is a set of probabilities that indicates how likely it is that at time \( t \) an observation symbol \( \xi_t \) is generated by each state from the set \( S \). More specifically, \( B = [b_j(m)] \) where \( b_j(m) = P(\xi_t = \nu_m | X_t = S_j) \) for \( 1 \leq m \leq M, \ 1 \leq j \leq N \), and \( \sum_{m=1}^{M} b_j(m) = 1 \):
It is expected that small values of $\xi$ are the result of a highly loaded network and therefore more likely generated by a low order state (one indicating low available bandwidth) and conversely. Based on this, probability values can be assigned and fixed in the model as shown below in the case of ten states and ten observation symbols. Note that 0.35 and 0.25 are high probability values assigned to more likely states:

$$B = \begin{bmatrix}
P(\nu_1|S_1) & P(\nu_2|S_1) & \cdots & P(\nu_M|S_1) \\
P(\nu_1|S_2) & P(\nu_2|S_2) & \cdots & P(\nu_M|S_2) \\
P(\nu_1|S_3) & P(\nu_2|S_3) & \cdots & P(\nu_M|S_3) \\
\vdots & \vdots & \vdots & \vdots \\
P(\nu_1|S_N) & P(\nu_2|S_N) & \cdots & P(\nu_M|S_N)
\end{bmatrix}$$

(4.12)

• Initial state probabilities ($\Pi$).

It has the probabilities for each state to be the first in the state sequence that generated the observations. $\Pi = [\pi_i]$ where $\pi_i = P(X_1 = S_i)$ for $1 \leq i \leq N$:

$$\Pi = [\pi_1 \pi_2 \cdots \pi_N] = [P(X_1 = S_1) P(X_1 = S_2) \cdots P(X_1 = S_N)]$$

(4.14)

Table 4.1 summarizes all the variables used in the estimation model.
Table 4.1: Available bandwidth HMM variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>Tight link capacity</td>
</tr>
<tr>
<td>$\Delta_{in}$</td>
<td>Packet pair separation before the tight link</td>
</tr>
<tr>
<td>$\Delta_{out}$</td>
<td>Packet pair separation after the tight link</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of states representing available bandwidth levels</td>
</tr>
<tr>
<td>$S$</td>
<td>Set of states (low to high): $S = S_1, S_2, \ldots, S_N$</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of distinct observation outcomes</td>
</tr>
<tr>
<td>$V$</td>
<td>Set of observations: $V = \nu_1, \nu_2, \ldots, \nu_M$</td>
</tr>
<tr>
<td>$\Pi$</td>
<td>Initial state probabilities</td>
</tr>
<tr>
<td>$T$</td>
<td>Sampling period. $t = 1 \ldots T$</td>
</tr>
<tr>
<td>$\epsilon_t$</td>
<td>Relative time dispersion at time $t$</td>
</tr>
<tr>
<td>$q_t$</td>
<td>State at time $t$</td>
</tr>
<tr>
<td>$Q$</td>
<td>State sequence: $Q = X_1, X_2, \ldots, X_T$</td>
</tr>
<tr>
<td>$\xi_t$</td>
<td>Observation symbol at time $t$</td>
</tr>
<tr>
<td>$O$</td>
<td>Observation sequence: $O = \xi_1, \xi_2, \ldots, \xi_T$</td>
</tr>
</tbody>
</table>

4.2.3 Parameter Estimation

Given an observation sequence $O = \xi_1, \xi_2, \ldots, \xi_T$, that is, a set of samples from the network during $T$, it is desired to estimate the model $\lambda$ that most likely generated that sequence, i.e. the model $\lambda = (A, B, \Pi)$ for which the $P(O | \lambda)$ is maximized. The solution to this problem is given by an iterative procedure formulated in the Baum-Welch algorithm [49].

The estimation tool presented in this work has implemented a modified version of the Baum-Welch algorithm written in C by Tapas Kanungo [50]. There are two main modifications to the algorithm. The first one is that the initial transition probability matrix $A$ is randomly generated to be a one-step transition matrix. Therefore, only the three main diagonals in the matrix have probability values. The second modification is that the
observation probability matrix $B$ is fixed so that the probabilities of observations being generated by the states do not change. This is due to the fact that it is expected that high congested links will increase the dispersion between packets and vice versa. Indeed, a link with zero cross traffic generates zero (or close to zero) dispersion between the pair of packets. The algorithm, which has a time complexity of $O(N^2T)$, runs as follows:

- Set the initial model $\lambda_0$ with a randomly generated one-step transition matrix $A_0$ and initial state probability vector $\Pi_0$. Matrix $B_0$ is initialized as explained in the previous section.
- Calculate a new $\hat{\lambda} = (\hat{A}, B_0, \hat{\Pi})$ based on $\lambda_0$ and the observation sequence $O$. See [48] for more details.
- if $\log P(O/\hat{\lambda}) - \log P(O/\lambda_0) < 0.001$ then stop
  else $\lambda_0 \leftarrow \hat{\lambda}$ and go to step 2.

Note that $\hat{B} = B_0$ all the time as explained before.

4.2.4 State Sequence Estimation

With an updated $\hat{\lambda}$, the next problem is to find the state sequence $Q = q_1, q_2, \ldots, q_T$ that maximizes the likelihood of $P(q_1, q_2, \ldots, q_T | O, \lambda)$. That state sequence is used to calculate the average available bandwidth during $T$. This is done by using another iterative algorithm, the Viterbi algorithm [51], which also has a $O(N^2T)$ time complexity. The algorithm selects the most likely path from a particular state to all possible paths and does the same for each state. See [48] or [50] for implementation details of the algorithm. The final most likely path represents the levels of available bandwidth that the probing
sampling packets have observed during the sampling time. As defined in Equation 1.1, the final estimation is based on the average utilization observed during $T$. Therefore, the available bandwidth is calculated as the average of all states in the sequence:

$$
\overline{A} = \frac{\overline{Q}}{N} \times C_t
$$

(4.15)
Chapter 5: Traceband: Monitoring Available Bandwidth

The hidden Markov model representation of the available bandwidth estimation process has been implemented in a new estimation tool called Traceband. This chapter describes and presents a performance evaluation of the tool. Traceband is compared with Spruce and Pathload which are the most representative tools of the probe gap and probe rate Model respectively. The performance evaluation is performed in a network testbed with Poisson, burst and self-similar synthetic cross traffic and in a real network path at University of South Florida.

5.1 Traceband Description

Traceband is a sender-receiver (client-server) tool written in ANSI C that uses the described hidden Markov representation of the available bandwidth dynamics to provide fast, continuous, low overhead, reliable, and accurate available bandwidth estimates. The tool has been developed for Linux-based operating systems and can be downloaded for evaluation purposes [52].

A train of probing packets pairs is sent from the sender application to the receiver at the end-to-end tight link rate. After interacting with cross traffic in the tight link, every packet pair in the train will provide a single dispersion value that constitutes one observation.
in the hidden Markov model sequence. The receiver application process the observation sequence and after estimating the sequence of states that generates the observations, provides a single averaged estimation of the available bandwidth. The detailed operation of the sender and receiver application is explained next.

5.1.1 Traceband Sender

The sender runs in cycles of ten estimations. In the first estimation the tool sends 50 UDP packet pairs 1498 bytes long. The nine remaining estimations are performed with 30 UDP packet pairs each. This reduction is possible since the HMM is able to learn the available bandwidth dynamics with an initial sample and keep the model updated with samples of reduced size. It is found from experimentation that re-learning every ten estimations is enough to maintain good accuracy with low overhead.

Traceband utilizes different values for the intra-gap and inter-gap times of packet pairs. The intra-gap refers to the time between the two packets of each packet pair. The intra-gap or $\Delta_{in}$ is set equal to the transmission time of a single probing packet in the tight link. In that way, the packet pair will be able to capture cross traffic in the queue, if any. The inter-gap refers to the time between pairs of probing packets, i.e. the time between the second packet of probing pair $i-1$ and the first packet of probing pair $i$. These times are obtained using the \texttt{gettimeofday()} function, and its values are sent to the receiver in the packet payload.

Similar to Spruce [29], Traceband performs a Poisson sampling process of the available bandwidth of the path by using exponentially distributed inter-gap times. In order to keep the overhead controlled and low, the mean inter-gap time value is calculated so that the
maximum overhead introduced by the tool is 5% of the tight link capacity. This allows the tool to be less intrusive with consistent accuracy.

5.1.2 Traceband Receiver

At the receiver side, the tool timestamps each received probing packet at the kernel level. This method helps to reduce delays generated by the `gettimeofday()` function as is mentioned in Section 1.2.

Packets are numbered to determine which packets are in the same pair and calculate the correct relative time dispersion ($\varepsilon$) between them. By applying Equation 4.10, the corresponding observation symbol for the HMM is determined for each packet pair.

The HMM module in Traceband receiver reads the values of N, M and B from a file to compute the model $\lambda$ based on the 50 (or less) observations. The model is used to determine the most likely sequence of states that generated the observations. For every new estimation, the initial model $\lambda_0$ is the output of the previous estimation. As it is defined in Equation 4.15, the sequence of states is then averaged and multiplied by the tight link capacity to provide a final available bandwidth estimation. The main Traceband algorithm running at the receiver is shown in Figure 5.1.

5.2 Performance Evaluation

The performance of Traceband is evaluated and compared with Pathload and Spruce, which are the current most representative tools for the probe rate model and probe gap model, respectively. These tools also show the best overall performance in the evaluation
Figure 5.1: Traceband receiver pseudo code.

performed in Chapter 3. Both tools and Traceband are implemented and used without any modification of their default parameters. The cross traffic is artificially generated in a testbed and taken from a real path at University of South Florida.

Similar to the evaluation performed in Chapter 3, the performance metrics used to evaluate Traceband are estimation error or accuracy, overhead, and estimation rate (similar to estimation time). The estimation error metric compares the estimation provided by the tool with the real average value obtained from the tcpdump trace, during the tool estimation period. The estimation error is given by the relative error according to Equation 3.5. The overhead is related to the amount of probing packets that the tool needs to inject into the network in order to perform the estimation. It is the percentage of tool traffic rate (tool traffic divided by the tool running time) with respect to the capacity of the tight link. Finally, the estimation rate shows how often the tool is able to provide an estimate. This rate is given in estimations per minute. The higher this value the better the estimation time of the tool. Pathload and Traceband directly report the estimation time. Spruce estimation
time is recorded using a script to calculate the difference of times before and after running the tool.

Experiments are performed using synthetic cross traffic generated over a testbed and with real network traffic transmitted over a path at University of South Florida.

### 5.2.1 Synthetic-generated Cross Traffic

The initial set of experiments are performed in the testbed shown in Figure 5.2. This is a controlled environment with a 10/100 Mbps tight link capacity. Cross traffic is generated from the host called US to the host called China and the estimation is performed from Sender to Receiver. The traffic generator MGEN is used to generate Poisson and burst cross traffic experiments; it allows to send cross traffic at different rates and with different probability distributions. Self-similar cross traffic is generated using a C application that sends packets from a trace file generated by [53]. A computer using tcpdump sniffs the output link in the router and records a trace with the joined cross and probing traffic. This trace is used to calculate the average link utilization every 1/10 seconds.
Table 5.1: Performance evaluation for 30% Poisson cross traffic.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Estimation Error</th>
<th>Estimations/min</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathload</td>
<td>6.71% ± 1.17%</td>
<td>1.754 ± 0.066</td>
<td>6.57% ± 0.20%</td>
</tr>
<tr>
<td>Spruce</td>
<td>7.77% ± 0.98%</td>
<td>5.579 ± 0.059</td>
<td>1.41% ± 0.02%</td>
</tr>
<tr>
<td>Traceband</td>
<td>8.83% ± 0.43%</td>
<td>11.645 ± 0.132</td>
<td>1.96% ± 0.03%</td>
</tr>
</tbody>
</table>

The tools are evaluated as if they were performing a continuous network monitoring task during a period of 200 seconds. In the case of Pathload and Spruce, it is necessary to run the tools in a loop. In the case of Traceband, the tool has an option to set the estimation period. For every experiment, the output of the tool is redirected to a log file that is processed to extract information about the time, amount, and values of the estimations. The tight link is loaded with 3 Mbps (30% of its capacity) with Poisson, bursty, and self-similar (Hurst parameter = 0.8) cross traffic.

5.2.1.1 Poisson Cross Traffic Experiments

Figures 5.3 to 5.5 show the tools’ estimations when the tight link is loaded with an average of 3 Mbps Poisson cross traffic. The mean value for the real available bandwidth is calculated as the average of all real available bandwidth values observed between two estimations of each tool. This is done in that way since the tools also provide an average over the estimation period. For comparison purposes, Pathload single points are calculated as the mid point of the range reported by the tool.

In the experiment results shown in Figure 5.3, Pathload makes 1.86 estimations per minute, inserts 6.86% of the tight link capacity as tool overhead, and presents an average estimation error of 6.92%. Spruce, as seen in Figure 5.4, performs 5.49 estimations per minute, inserts 1.42% of the tight link capacity as tool overhead, and has an average estimation
error of 8.54%. Finally, Traceband as shown in Figure 5.5 performs an average of 11.42 estimations per minute, inserts 1.90% of the tight link capacity as tool overhead, and presents an average estimation error of 8.40%. These results correspond to one single experiment. Table 5.1 shows 95% confidence intervals for each performance metric as a result of running the experiments five times.

In this Poisson cross traffic scenario, the three tools under evaluation show estimation errors below 10%, which according to evaluations performed by other authors like in [17] can be considered as of high accuracy. Compared with Spruce, Traceband performs twice
Figure 5.5: Traceband estimation for a 10 Mbps tight link with 30% of Poisson cross traffic.

Figure 5.6: Pathload estimation for a 10 Mbps tight link with 30% of bursty cross traffic.

the number of estimations per minute with similar total overhead. Pathload has shown to be more than three times more intrusive and more than six times slower than Traceband.

5.2.1.2 Bursty Cross Traffic Experiments

Figure 5.6 to 5.8 show the results of running the tools when the network is loaded with 3 Mbps of bursty cross traffic. The bursts occur at random intervals with an average interval...
Figure 5.7: Spruce estimation for a 10 Mbps tight link with 30% of bursty cross traffic.

Figure 5.8: Traceband estimation for a 10 Mbps tight link with 30% of bursty cross traffic.

from the start of one burst until the start of the next of 10.0 seconds. The duration of each burst is of exponential statistics with an average burst duration of 5.0 seconds.

In this case, as it is shown in Table 5.2, Traceband shows the minimum estimation error with the maximum number of estimations per minute. As in the Poisson cross traffic case, the amount of overhead introduced by Traceband is considerably lower than Pathload. As it can be observed from Figure 5.8, since Traceband performs more estimations per minute, the tool is able to accurately react to periods where the tight link has no cross
Table 5.2: Performance evaluation for 30% burst cross traffic.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Estimation Error</th>
<th>Estimations/min</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathload</td>
<td>12.06%</td>
<td>2.45</td>
<td>7.65%</td>
</tr>
<tr>
<td>Spruce</td>
<td>8.21%</td>
<td>5.46</td>
<td>1.34%</td>
</tr>
<tr>
<td>Traceband</td>
<td>4.12%</td>
<td>11.26</td>
<td>1.98%</td>
</tr>
</tbody>
</table>

traffic. Further, during those empty periods, the HMM provides 100% accuracy setting the estimations to the state representing the highest availability.

5.2.1.3 Self-similar Cross Traffic Experiments

Figures 5.9 to 5.11 show the results of running the tools when the network is loaded with 3 Mbps of self-similar cross traffic with Hurst parameter equal to 0.8. To generate this traffic, a Bounded Pareto burst size and exponential interburst time trace is created using syntraf1a.c [53]. The trace is generated with a target utilization of 30% for a 10 Mbps link capacity. It contains the interarrival time and packet size of each packet.

To playback the trace into the real network, an application called "udpreply" has been created. This application reads the trace file and sends the traffic to the destination.

Since it is assumed that the available bandwidth can be modeled by a hidden Markov model, the memoryless property holds when the cross traffic is Poisson but not self-similar. Therefore, as expected, Traceband shows a higher but still low estimation error. The 95% confidence intervals calculated for the three evaluated tools under self-similar cross traffic are shown in Table 5.3. Overhead and estimation rates are very similar to results obtained in the Poisson cross traffic scenario (Table 5.1). The estimation error is however higher but still in a low range (around 10%) with low variability.
Figure 5.9: Pathload estimation for a 10 Mbps capacity and 30% self-similar cross traffic.

Figure 5.10: Spruce estimation for a 10 Mbps capacity and 30% self-similar cross traffic.

Figure 5.11: Traceband estimation for a 10 Mbps capacity and 30% self-similar cross traffic.
Table 5.3: Performance evaluation for 30% self-similar cross traffic.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Estimation Error</th>
<th>Estimations/min</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathload</td>
<td>4.68% ± 1.29%</td>
<td>1.600 ± 0.092</td>
<td>6.31% ± 0.22%</td>
</tr>
<tr>
<td>Spruce</td>
<td>7.72% ± 1.57%</td>
<td>5.482 ± 0.060</td>
<td>1.36% ± 0.04%</td>
</tr>
<tr>
<td>Traceband</td>
<td>10.48% ± 1.33%</td>
<td>12.543 ± 0.051</td>
<td>2.09% ± 0.02%</td>
</tr>
</tbody>
</table>

Table 5.4: Performance evaluation with real cross traffic in a 100 Mbps path.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Estimation Error</th>
<th>Estimations/min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathload</td>
<td>13.89%</td>
<td>4.20</td>
</tr>
<tr>
<td>Spruce</td>
<td>11.07%</td>
<td>5.91</td>
</tr>
<tr>
<td>Traceband</td>
<td>10.95%</td>
<td>109.85</td>
</tr>
</tbody>
</table>

5.2.2 Internet-traffic Based Experiments

To perform experiments with Internet traffic, it is used a path connecting a computer from the Information Systems Lab [54] to a location in CUTR [55] through a layer-3 switch connected to Internet. The path has a 100 Mbps tight link. Since this is not a fully controlled environment, the traffic traces have been provided by a network administrator in the university and have a granularity of 10 seconds.

The average values plotted in Figures 5.12 to 5.14 are summarized in Table 5.4. Here Traceband is as expected faster than the other tools. Having a greater number of estimations per minute allows Traceband to have a better average accuracy when compared with a 10-seconds granularity real traffic trace.

5.2.3 Moving Average Algorithm

In this section, it is described a moving average algorithm meant to improve the estimation error and variability of Traceband. The idea of the algorithm is similar to the one
Figure 5.12: Pathload estimation for a 100 Mbps tight link with Internet cross traffic.

Figure 5.13: Spruce estimation for a 100 Mbps tight link with Internet cross traffic.

Figure 5.14: Traceband estimation for a 100 Mbps tight link with Internet cross traffic.
proposed in [56] to filter out abrupt changes in the received signal strength of wireless devices. It calculates the mean available bandwidth $\overline{A}$ and the standard deviation $S$ of five continuous estimations to calculate a 95% confidence interval using the t-student distribution:

$$\Delta A = \frac{S \times Q_{n=5,0.95}}{\sqrt{5}}$$  \hspace{1cm} (5.1)

$$Interval = [\overline{A} - \Delta A, \overline{A} + \Delta A]$$  \hspace{1cm} (5.2)

where $Q_{n=5,0.95}$ is the 95% quantile on the Student’s t-distribution for n=5 available bandwidth estimations. If the next single estimation lies above or below the upper or lower limits calculated using Equation 5.2, that estimation is considered a “peak” (a very rare sample) and it is changed to the interval upper or lower limit value. Then a new confidence interval is calculated with the last five estimations (a window of five estimations is continuously shifted once every time). The smoothed estimation is therefore the result of averaging the last five measurements after adjusting those out of the confident interval limits. The algorithm is shown in Figure 5.18. It is worth noticing that this technique is
This optional moving average algorithm available in Traceband is evaluated using Poisson and self-similar cross traffic. In the Poisson case, the results shown in Figures 5.3 to 5.5 are smoothed using the algorithm. The results are shown in Figures 5.15 to 5.17. From Figure 5.17, it can be observed that since Traceband estimations are more symmetric over the mean value than Pathload’s and Spruce’s, after applying the moving average, the tool
shows the best accuracy and the lowest variability. It is worth noticing that given the small estimation rate of Pathload, using this filtering algorithm the tool is not able to perform the first estimate before 150 seconds. As before, for the set of five experiments, a 95% confidence interval is calculated. The overhead and estimation rate are the same as in Table 5.1 but the estimation error results are shown in Table 5.5.

In the case of self-similar cross traffic, in which Traceband’s performance worsens with the Hurst parameter, the moving average algorithm makes important improvements in

Table 5.5: Estimation error after applying moving average to experiment results with Poisson traffic.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Estimation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathload</td>
<td>4.88% ± 2.13%</td>
</tr>
<tr>
<td>Spruce</td>
<td>3.84% ± 1.92%</td>
</tr>
<tr>
<td>Traceband</td>
<td>2.93% ± 1.42%</td>
</tr>
</tbody>
</table>

Figure 5.18: Moving average algorithm.
both, the estimation error and its variability. Figures 5.19 to 5.21 show the results of applying the moving average algorithm to the same data used to plot Figures 5.9 to 5.11 with a Hurst parameter of $H = 0.8$. As before, for comparison purposes, the technique is also applied to Pathload and Spruce. From the Figure, it can be observed that Traceband’s estimation error is reduced considerably using this methodology, as it is its variability. Further, the algorithm improves Pathload’s and Spruce’s performance as well. The 95% confidence intervals are shown in Table 5.6, which shows that now Traceband is the tool with the lowest variability.
5.3 Additional Experiments

This section presents the results of experiments performed to study the behavior of Traceband under variations in network conditions and variations in its own parameters related to the hidden Markov model. Results presented here motivate a factorial design analysis to be performed as part of the future work.

Table 5.6: Estimation error after applying the moving average to experiment results with self-similar traffic.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Estimation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathload</td>
<td>3.90% ± 1.72%</td>
</tr>
<tr>
<td>Spruce</td>
<td>4.67% ± 1.24%</td>
</tr>
<tr>
<td>Traceband</td>
<td>5.12% ± 0.49%</td>
</tr>
</tbody>
</table>
5.3.1 Hurst Parameter

One additional experiment is performed using the same testbed shown in Figure 5.2 to look at the effect of the Hurst parameter in the estimation error of the tools. Figure 5.22 shows these results with and without using the moving average algorithm and having the Hurst parameter varied from 0.5 to 0.8. For each point in the graph, a 95% confidence interval is plotted. From Figure 5.22(a), it is clear that Pathload is the most accurate tool followed by Spruce and Traceband, in that order. As expected, the self similarity level affects the accuracy of the evaluated tools, and in particular, the performance of Traceband, which not only increases its estimation error but also its variability.

When the moving average algorithm is applied, Figure 5.22(b) now shows that all the tools present fairly similar results, but better compared with the case without the moving average algorithm.
### Table 5.7: Traceband performance for high and low tight link capacities.

<table>
<thead>
<tr>
<th>Capacity</th>
<th>Estimation Error</th>
<th>Estimations per minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Mbps</td>
<td>10.48%</td>
<td>12.54</td>
</tr>
<tr>
<td>100 Mbps</td>
<td>10.95%</td>
<td>109.85</td>
</tr>
</tbody>
</table>

### 5.3.2 Tight Link Capacity

As it is shown in Table 5.7, Traceband estimation error has no significant variation when the tight link capacity changes. On the other hand, since the tool is able to send probing packets at a very fast rate, the number of estimations per minute increases with the tight link capacity.

### 5.3.3 Number of States in the HMM

The number of states is also varied from 5 to 20 with the corresponding number of observation symbols (that is, from 5 to 20). As expected, Figure 5.23(b) shows how the more states and observations, the receiving side of the application has to perform more calculations and therefore the number of estimations per minute decreases. With regards to the estimation error, as it is shown in Figure 5.23(a), there is no variation trend when the number of states changes.

This is an aspect that needs further research but it is probably due to the definition of the observation probability matrix $B$. Appendix B shown the defined matrices according to the following policy:

- 25% of the most likely states responsible for an observation accumulate 70% of the probabilities
Figure 5.23: Traceband estimation error and time for different number of states.

- 50% of the most likely states responsible for an observation accumulate 90% of the probabilities
Chapter 6: Conclusions and Future Work

This dissertation studies the problem of available bandwidth estimation and proposes a novel technique and tool to accurately and non-intrusively monitor the available bandwidth of an end-to-end path. This section presents the conclusions drawn from the contributions presented in Section 1.4 and outlines future directions of this work.

6.1 Conclusions

- Regarding the performance evaluation of available bandwidth estimation tools, the main conclusion is that current tools are still far to provide the high accuracy, low estimation time, low overhead, and reliability required by the network applications. Table 3.5 shows that only network management and SLA compliance applications could benefit from tools like Spruce and Pathload (in that order). Other network applications still need a better performance of the current tools.

On the other hand, Table 3.4 summarizes the tools that perform best and worst in specific network scenarios. This information is useful to decide which tool would be the most suitable for a network application running in a particular network scenario. In general, the study clearly shows which tools might be the best choices for particular applications, networks, and network conditions, and which aspects need
further research in this area, hoping that this will trigger more interest to push the state-of-the-art in this field.

- A hidden Markov model of the available bandwidth is able to adjust observations affected by the busy nature of cross traffic and by errors associated to the network infrastructure. This model allows to keep the number of observations low with acceptable accuracy in the estimation. By using a moving average technique it is possible to smooth the estimate and improve the accuracy even further.

- The hidden Markov model implemented in Traceband provides a novel approach to obtain accurate estimations with a considerable improvement in the estimation time of the tool. This improvement makes the tool unique since it is the only one able to accurately monitor the available bandwidth with a granularity never shown before. Traceband compared with Pathload and Spruce, not only provides better performance results overall, but it is also able to react and accurately estimate the available bandwidth under abrupt changes in cross traffic. Experimental results using Poisson, bursty, and self-similar cross traffic show that Traceband provides more estimations per unit time with comparable accuracy to Pathload and Spruce and with less probing traffic. The tool tested in a University of South Florida network path shows similar result as in the testbed experiments.

- The testbed infrastructure built for this work is able to accurately emulate different network scenarios and conditions as seen in the Internet. This infrastructure supports several research projects including class assignments in the graduate course of computer networks. The testbed can be accessed over the Internet which makes it available to other researchers.
6.2 Future Work

The research included in this dissertation can be extended in several ways. Some of them are:

- In the analytical evaluation of available bandwidth estimation tools a more precise estimation of the real available bandwidth requires to study the exact distribution of the tools’ probing traffic. This could imply the use of a G/M/1 network of queues if the distribution is different than Poisson.

- All the traffic in the network testbed is artificially generated. A more realistic approach is to use traces taken from different Internet sources [57] [58] [59] [60] and to replay them into the network testbed. This can be done using a tool called tcpreplay [61].

- Traceband requires a previous estimation of the tight link capacity. This estimation can be included in the tool by sending back-to-back packets and using those whose one way delays are minimum to estimate the rate variation they suffer because of the tight link capacity.

- There are additional network scenarios to evaluate and compare with other tools the performance of Traceband. One of them is over 1 or 10 Gbps links.

- Regarding the hidden Markov model approach and its impact on the estimation tool, more work has to be done to study the effect of variations in the definition of the observation probabilities for a given number of states and observation symbols. Given the large number of combinations, a factorial design analysis similar to the one shown in Appendix A can be conducted to reduce the number of experiments.
Another analysis can be performed with different values (fixed or learned) of the observation probabilities.
List of References


[34] ——, “Regression-Based Available Bandwidth Measurements,” 2002.


Appendices
Appendix A: Factorial Design for Available Bandwidth Evaluation

There are three main approaches used in performance evaluation: experimental (using a real system), analytic (using mathematics), and simulation (using computer tools implementing a system model). In any of these approaches, experiments have to be performed to evaluate, compare or analyze the behavior of the system. Experimental Design helps to obtain the maximum information using the minimum number of experiments.

The experimental design used in this work is meant to evaluate the performance of several bandwidth estimation techniques under variable path characteristics and cross traffic. Design an experiment imply to define the following:

- **Response variables or metrics**: they are the performance metrics or outcomes of the experiment. For the available bandwidth evaluation, these are the response variables: estimation error or accuracy of the available bandwidth [Mbps], estimation time or response time [s], overhead which is define as $(\text{tool traffic rate} \times 100) / (\text{Tight link capacity}) \%$, and reliability defined as $(\text{Number of estimations}) / (\text{Number of trials}) \%$

- **Experimental factors**: They are the variables that affect the response variables. For the available bandwidth evaluation, they are the following: capacity [Mbps], propagation delay [ms], packet loss rate [0..1], cross traffic amount [% of the link capacity], and cross traffic packet size [Bytes]

- **Factor levels**: they are the different values to be studied for the factors. For example, in the case of the capacity values of 10, 20, 30, ..., 200 Mbps.
Appendix A: (continued)

- Replication: Is the number of experiment repetitions for each set of levels. In the case of the available bandwidth evaluation 10 successful repetitions are used to calculate a 95% confidence interval using a t-test.

For a system with $k$ factors, with the $i^{th}$ factor having $n_i$ levels, then if $r$ replications are performed for each level and every combination of all levels of all factors is studied, the TNR (total number of repetitions) is given by this Full Factorial Design:

$$TNR = r \prod_{i=1}^{k} n_i$$  \hspace{1cm} (A.1)

This will lead to a large number of experiments because all the levels for every factor are considered. A more simplified model is denoted by $2^k$ factorial design. The idea is to use only 2 level values for each factor identified as -1 (for the smaller) and +1 (for the larger) and study the possible combinations. This will lead to a number of repetitions given by:

$$TNR = r2^k$$  \hspace{1cm} (A.2)

Table A.1: $2^5$ factorial design matrix.

<table>
<thead>
<tr>
<th>Exp.</th>
<th>Capacity</th>
<th>Delay</th>
<th>PLR</th>
<th>%XTraffic</th>
<th>XT packet size</th>
<th>Response (Error)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5000 (-1)</td>
<td>10 (-1)</td>
<td>0.01 (-1)</td>
<td>25% (-1)</td>
<td>512 (-1)</td>
<td>$y_1$</td>
</tr>
<tr>
<td>2</td>
<td>100000 (+1)</td>
<td>10 (-1)</td>
<td>0.01 (-1)</td>
<td>25% (-1)</td>
<td>512 (-1)</td>
<td>$y_2$</td>
</tr>
<tr>
<td>3</td>
<td>5000 (-1)</td>
<td>80 (+1)</td>
<td>0.01 (-1)</td>
<td>25% (-1)</td>
<td>512 (-1)</td>
<td>$y_3$</td>
</tr>
<tr>
<td>30</td>
<td>100000 (+1)</td>
<td>10 (-1)</td>
<td>0.01 (+1)</td>
<td>75% (+1)</td>
<td>1408 (+1)</td>
<td>$y_{30}$</td>
</tr>
<tr>
<td>31</td>
<td>5000 (-1)</td>
<td>80 (+1)</td>
<td>0.07 (+1)</td>
<td>75% (+1)</td>
<td>1408 (+1)</td>
<td>$y_{31}$</td>
</tr>
<tr>
<td>32</td>
<td>100000 (+1)</td>
<td>80 (+1)</td>
<td>0.07 (+1)</td>
<td>75% (+1)</td>
<td>1408 (+1)</td>
<td>$y_{32}$</td>
</tr>
</tbody>
</table>
Appendix A: (continued)

To do that, it has to be defined first a $2^k$ factorial design matrix as show in Table A.1 where $y_i$ is determined by averaging the $r$ replications of experiment $i$. Using the design matrix, the MEF (main effect of varying a factor) is defined as the average change in the response variable due to moving the factor from its -1 to its +1 level (increasing it) with all other factors being constant. It is defined as:

$$MEF_j = \frac{[\text{column} \, j]^T \times \text{response : column}}{2^{k-1}} \quad (A.3)$$

A low MEF means that increasing the factor level has no effect in the response variable. Table 3.3 shows the main effect in the performance metrics of five estimation tools when varying one factor.

It is also possible to determine the average interaction effect of two or more factors. In the case of two factors, IEF is defined as the average change in the response variable when the two factors are at the same level and when they are at opposite levels. It is defined as:

$$IEF_{ij} = \frac{[\text{column} \times \text{column} \, j]^T \times \text{response : column}}{2^{k-1}} \quad (A.4)$$

A low IEF means a poor interaction between the two factors. The average interaction of two, three, four, and five factors, which are shown in Tables A.2 to A.5.
Table A.2: Main effect in the performance metrics when varying two factors.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Response Variables</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Overhead (%)</td>
<td>Time (s)</td>
<td>Reliab. (%)</td>
</tr>
<tr>
<td>Pathload</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay</td>
<td>0.54</td>
<td>0.39</td>
<td>14.27</td>
<td>0.00</td>
</tr>
<tr>
<td>Capacity + PLR</td>
<td>6.24</td>
<td>1.13</td>
<td>14.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Delay + PLR</td>
<td>1.36</td>
<td>0.78</td>
<td>-16.25</td>
<td>0.00</td>
</tr>
<tr>
<td>Delay + %XTraffic</td>
<td>-1.61</td>
<td>0.45</td>
<td>-2.70</td>
<td>0.00</td>
</tr>
<tr>
<td>PLR + %XTraffic</td>
<td>0.30</td>
<td>0.62</td>
<td>-5.85</td>
<td>0.00</td>
</tr>
<tr>
<td>%XTraffic + XT packet size</td>
<td>4.63</td>
<td>-0.48</td>
<td>7.42</td>
<td>0.00</td>
</tr>
<tr>
<td>IGI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay</td>
<td>-133.04</td>
<td>0.20</td>
<td>-0.35</td>
<td>0.00</td>
</tr>
<tr>
<td>Capacity + PLR</td>
<td>69.82</td>
<td>0.36</td>
<td>0.58</td>
<td>0.00</td>
</tr>
<tr>
<td>Delay + PLR</td>
<td>-50.41</td>
<td>0.03</td>
<td>0.47</td>
<td>0.00</td>
</tr>
<tr>
<td>Delay + %XTraffic</td>
<td>122.91</td>
<td>0.00</td>
<td>-0.69</td>
<td>0.00</td>
</tr>
<tr>
<td>PLR + %XTraffic</td>
<td>-58.10</td>
<td>-0.05</td>
<td>0.05</td>
<td>0.00</td>
</tr>
<tr>
<td>%XTraffic + XT packet size</td>
<td>110.88</td>
<td>-0.03</td>
<td>-0.38</td>
<td>0.00</td>
</tr>
<tr>
<td>Spruce</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay</td>
<td>6.32</td>
<td>0.12</td>
<td>-0.50</td>
<td>5.19</td>
</tr>
<tr>
<td>Capacity + PLR</td>
<td>-0.88</td>
<td>0.27</td>
<td>-0.43</td>
<td>-7.95</td>
</tr>
<tr>
<td>Delay + PLR</td>
<td>2.19</td>
<td>0.06</td>
<td>-0.48</td>
<td>-4.69</td>
</tr>
<tr>
<td>Delay + %XTraffic</td>
<td>1.18</td>
<td>-0.02</td>
<td>0.14</td>
<td>0.78</td>
</tr>
<tr>
<td>PLR + %XTraffic</td>
<td>3.72</td>
<td>-0.05</td>
<td>-0.05</td>
<td>-5.78</td>
</tr>
<tr>
<td>%XTraffic + XT packet size</td>
<td>3.67</td>
<td>0.10</td>
<td>-1.18</td>
<td>9.94</td>
</tr>
<tr>
<td>Abing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay</td>
<td>324.09</td>
<td>1.86</td>
<td>-0.01</td>
<td>-2.64</td>
</tr>
<tr>
<td>Capacity + PLR</td>
<td>94.92</td>
<td>0.76</td>
<td>-0.01</td>
<td>1.57</td>
</tr>
<tr>
<td>Delay + PLR</td>
<td>83.72</td>
<td>0.16</td>
<td>0.01</td>
<td>-0.75</td>
</tr>
<tr>
<td>Delay + %XTraffic</td>
<td>-317.66</td>
<td>0.10</td>
<td>0.01</td>
<td>-1.81</td>
</tr>
<tr>
<td>PLR + %XTraffic</td>
<td>-76.44</td>
<td>-0.06</td>
<td>0.01</td>
<td>-2.26</td>
</tr>
<tr>
<td>%XTraffic + XT packet size</td>
<td>-307.68</td>
<td>-0.04</td>
<td>0.00</td>
<td>-0.25</td>
</tr>
<tr>
<td>Pathchirp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay</td>
<td>-11.06</td>
<td>-0.07</td>
<td>1.10</td>
<td>0.00</td>
</tr>
<tr>
<td>Capacity + PLR</td>
<td>33.52</td>
<td>0.55</td>
<td>-0.56</td>
<td>0.00</td>
</tr>
<tr>
<td>Delay + PLR</td>
<td>-0.58</td>
<td>-0.07</td>
<td>0.81</td>
<td>0.00</td>
</tr>
<tr>
<td>Delay + %XTraffic</td>
<td>9.38</td>
<td>0.06</td>
<td>-0.52</td>
<td>0.00</td>
</tr>
<tr>
<td>PLR + %XTraffic</td>
<td>-55.18</td>
<td>0.17</td>
<td>-1.21</td>
<td>0.00</td>
</tr>
<tr>
<td>%XTraffic + XT packet size</td>
<td>-42.69</td>
<td>-0.02</td>
<td>0.77</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table A.3: Main effect in the performance metrics when varying three factors.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Response Variables</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Overhead (%)</td>
<td>Time (s)</td>
<td>Reliab. (%)</td>
<td></td>
</tr>
<tr>
<td>Pathload</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + PLR</td>
<td>-1.24</td>
<td>-0.07</td>
<td>-2.62</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + %XTraffic</td>
<td>-0.27</td>
<td>-0.18</td>
<td>0.66</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + XT packet size</td>
<td>4.25</td>
<td>-0.31</td>
<td>0.29</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay + PLR + XT packet size</td>
<td>-3.01</td>
<td>-0.12</td>
<td>-3.28</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay + %XTraffic + XT packet size</td>
<td>2.35</td>
<td>0.30</td>
<td>-5.49</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>PLR + %XTraffic + XT packet size</td>
<td>0.20</td>
<td>0.28</td>
<td>-4.35</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>IGI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + PLR</td>
<td>46.34</td>
<td>-0.03</td>
<td>-0.23</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + %XTraffic</td>
<td>-131.97</td>
<td>0.00</td>
<td>-0.58</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + XT packet size</td>
<td>-141.01</td>
<td>0.02</td>
<td>-0.89</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay + PLR + XT packet size</td>
<td>-57.43</td>
<td>0.03</td>
<td>-1.25</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay + %XTraffic + XT packet size</td>
<td>133.34</td>
<td>-0.05</td>
<td>0.77</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>PLR + %XTraffic + XT packet size</td>
<td>-55.95</td>
<td>0.01</td>
<td>-0.60</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Spruce</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + PLR</td>
<td>-5.20</td>
<td>-0.06</td>
<td>-0.17</td>
<td>1.64</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + %XTraffic</td>
<td>4.78</td>
<td>0.03</td>
<td>-0.95</td>
<td>-4.00</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + XT packet size</td>
<td>-0.93</td>
<td>0.00</td>
<td>-0.09</td>
<td>8.59</td>
<td></td>
</tr>
<tr>
<td>Delay + PLR + XT packet size</td>
<td>-4.49</td>
<td>-0.03</td>
<td>0.01</td>
<td>-1.49</td>
<td></td>
</tr>
<tr>
<td>Delay + %XTraffic + XT packet size</td>
<td>-0.48</td>
<td>-0.06</td>
<td>0.69</td>
<td>-5.43</td>
<td></td>
</tr>
<tr>
<td>PLR + %XTraffic + XT packet size</td>
<td>-0.33</td>
<td>0.05</td>
<td>-1.05</td>
<td>13.49</td>
<td></td>
</tr>
<tr>
<td>Abing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + PLR</td>
<td>-97.70</td>
<td>-0.16</td>
<td>0.01</td>
<td>-3.44</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + %XTraffic</td>
<td>323.01</td>
<td>-0.11</td>
<td>0.01</td>
<td>6.26</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + XT packet size</td>
<td>-176.39</td>
<td>0.01</td>
<td>0.00</td>
<td>-1.65</td>
<td></td>
</tr>
<tr>
<td>Delay + PLR + XT packet size</td>
<td>-9.60</td>
<td>0.01</td>
<td>0.00</td>
<td>-1.53</td>
<td></td>
</tr>
<tr>
<td>Delay + %XTraffic + XT packet size</td>
<td>177.93</td>
<td>-0.03</td>
<td>0.00</td>
<td>5.34</td>
<td></td>
</tr>
<tr>
<td>PLR + %XTraffic + XT packet size</td>
<td>-30.55</td>
<td>-0.02</td>
<td>0.00</td>
<td>-7.11</td>
<td></td>
</tr>
<tr>
<td>Pathchirp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + PLR</td>
<td>0.99</td>
<td>0.07</td>
<td>0.65</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + %XTraffic</td>
<td>-10.57</td>
<td>-0.08</td>
<td>-0.48</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Capacity + Delay + XT packet size</td>
<td>7.49</td>
<td>-0.02</td>
<td>0.83</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay + PLR + XT packet size</td>
<td>7.14</td>
<td>0.07</td>
<td>1.10</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay + %XTraffic + XT packet size</td>
<td>-10.45</td>
<td>-0.04</td>
<td>-1.36</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>PLR + %XTraffic + XT packet size</td>
<td>-1.94</td>
<td>-0.11</td>
<td>0.27</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>
## Table A.4: Main effect in the performance metrics when varying four factors.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Response Variables</th>
<th>Error (%)</th>
<th>Overhead (%)</th>
<th>Time (s)</th>
<th>Reliab. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathload</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-1.39</td>
<td>-0.02</td>
<td>6.40</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-6.52</td>
<td>-0.01</td>
<td>5.25</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>0.84</td>
<td>-0.21</td>
<td>3.11</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>10.60</td>
<td>-0.28</td>
<td>11.16</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay+PLR+XT psizesize</td>
<td>-3.51</td>
<td>-0.13</td>
<td>1.78</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>IGI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>48.62</td>
<td>-0.01</td>
<td>-0.49</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>57.29</td>
<td>-0.01</td>
<td>-0.32</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-138.66</td>
<td>0.05</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>50.22</td>
<td>-0.01</td>
<td>-0.72</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay+PLR+XT psizesize</td>
<td>-59.39</td>
<td>0.00</td>
<td>0.82</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Spruce</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-5.58</td>
<td>0.01</td>
<td>-0.56</td>
<td>3.72</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-5.48</td>
<td>0.03</td>
<td>-0.08</td>
<td>14.41</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-1.60</td>
<td>0.06</td>
<td>-0.31</td>
<td>3.90</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>0.59</td>
<td>-0.05</td>
<td>0.27</td>
<td>8.71</td>
<td></td>
</tr>
<tr>
<td>Delay+PLR+XT psizesize</td>
<td>-3.85</td>
<td>0.00</td>
<td>0.27</td>
<td>-4.81</td>
<td></td>
</tr>
<tr>
<td>Abing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-99.97</td>
<td>-0.09</td>
<td>-0.01</td>
<td>-0.60</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-1.43</td>
<td>-0.02</td>
<td>0.00</td>
<td>0.67</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-172.15</td>
<td>0.02</td>
<td>0.00</td>
<td>-0.96</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>19.39</td>
<td>0.02</td>
<td>0.00</td>
<td>-1.95</td>
<td></td>
</tr>
<tr>
<td>Delay+PLR+XT psizesize</td>
<td>-5.69</td>
<td>0.06</td>
<td>0.00</td>
<td>6.14</td>
<td></td>
</tr>
<tr>
<td>Pathchirp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>2.78</td>
<td>-0.01</td>
<td>-0.03</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-4.93</td>
<td>-0.09</td>
<td>1.31</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>11.30</td>
<td>0.04</td>
<td>-1.75</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Cap.+Delay+PLR+XT psizesize</td>
<td>-1.52</td>
<td>0.11</td>
<td>-0.23</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Delay+PLR+XT psizesize</td>
<td>6.01</td>
<td>0.00</td>
<td>-1.53</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>
Appendix A: (continued)

Table A.5: Main effect in the performance metrics when varying five factors.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Response Variables</th>
<th>Error (%)</th>
<th>Overhead (%)</th>
<th>Time (s)</th>
<th>Reliab. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pathload</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>-0.64</td>
<td>0.161</td>
<td>-0.20</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>IGI</td>
<td>56.67</td>
<td>0.002</td>
<td>1.14</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spruce</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>-5.67</td>
<td>0.008</td>
<td>-0.26</td>
<td>-6.09</td>
<td></td>
</tr>
<tr>
<td>Abing</td>
<td>3.39</td>
<td>-0.074</td>
<td>0.00</td>
<td>-0.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pathchirp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>-4.56</td>
<td>0.007</td>
<td>-1.62</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B: Observation Probability Matrices

Tables B.1 and B.2 show the observation probabilities as described in Section 4.2.2 and chosen by Traceband according to the number of states and observation symbols.

Table B.1: Observation probability matrix for 5 states and 5 observation symbols.

<table>
<thead>
<tr>
<th></th>
<th>0.50</th>
<th>0.25</th>
<th>0.15</th>
<th>0.05</th>
<th>0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.50</td>
<td>0.15</td>
<td>0.05</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>0.20</td>
<td>0.50</td>
<td>0.20</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>0.05</td>
<td>0.15</td>
<td>0.50</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>0.05</td>
<td>0.15</td>
<td>0.25</td>
<td>0.50</td>
<td></td>
</tr>
</tbody>
</table>

Table B.2: Observation probability matrix for 10 states and 10 observation symbols.

<table>
<thead>
<tr>
<th></th>
<th>0.35</th>
<th>0.25</th>
<th>0.15</th>
<th>0.10</th>
<th>0.05</th>
<th>0.03</th>
<th>0.03</th>
<th>0.02</th>
<th>0.01</th>
<th>0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.20</td>
<td>0.35</td>
<td>0.20</td>
<td>0.10</td>
<td>0.05</td>
<td>0.03</td>
<td>0.03</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>0.08</td>
<td>0.20</td>
<td>0.35</td>
<td>0.20</td>
<td>0.07</td>
<td>0.03</td>
<td>0.03</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>0.03</td>
<td>0.08</td>
<td>0.20</td>
<td>0.35</td>
<td>0.20</td>
<td>0.07</td>
<td>0.03</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>0.03</td>
<td>0.08</td>
<td>0.20</td>
<td>0.35</td>
<td>0.20</td>
<td>0.07</td>
<td>0.02</td>
<td>0.02</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>0.07</td>
<td>0.20</td>
<td>0.35</td>
<td>0.20</td>
<td>0.08</td>
<td>0.03</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.07</td>
<td>0.20</td>
<td>0.35</td>
<td>0.20</td>
<td>0.08</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.03</td>
<td>0.07</td>
<td>0.20</td>
<td>0.35</td>
<td>0.20</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.03</td>
<td>0.05</td>
<td>0.10</td>
<td>0.20</td>
<td>0.35</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.03</td>
<td>0.05</td>
<td>0.10</td>
<td>0.15</td>
<td>0.25</td>
<td>0.35</td>
<td></td>
</tr>
</tbody>
</table>
Mr. Cesar Guerrero is a Ph.D. candidate in the department of Computer Science and Engineering at the University of South Florida. He received his M.S. degree in Computer Science from ITESM (Mexico) and UNAB (Colombia) in 2002. He also received his M.S. degree in Computer Engineering from USF (US) in 2007. He is a Fulbright scholar who works with Universidad Autonoma de Bucaramanga (Colombia). His research interests include Bandwidth Estimation and Network Measurement.