The Hong Kong Polytechnic University
Department of Computing

Design and Analysis of Robust Techniques for Inferring Network Path Properties

by

Edmond Wun-Wah Chan

A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

June 2010

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Edmond Wun-Wah Chan (Name of student)

To my wife and my family.

Abstract

Exploring network path properties is useful not only for both consumers and providers to verify service level agreement, choose the best route, and diagnose performance problems; but also for network applications and services to adapt to network path characteristics and improve their performance. However, the design and implementation of a reliable measurement method is very challenging for the Internet landscape today. It is difficult to obtain accurate measurement in the midst of interference from cross traffic which can intervene measurement traffic and cause packet loss. Another challenge is to characterize asymmetric-path (i.e., forward and reverse paths) properties between a measuring node and a remote endpoint, where acquiring the remote endpoint's cooperation (in terms of setting up additional software) is usually impracticable. Although many methods can seamlessly perform non-cooperative measurement, most of them measure either round-trip path properties or restricted configurations of asymmetric paths.

In this research, we first propose a fast and efficient method, called minimum delay difference (MDDIF), for path capacity measurement using packet pairs. The path capacity is defined as the smallest transmission rate of a set of links forming a network path. Measuring the path capacity is difficult, because the accuracy and speed can be adversely

affected by cross traffic present on the path. Moreover, minimizing the amount of overheads, including the amount of measurement traffic and the storage and computation requirement, are also important to make the measurement feasible for a practical network. Unlike the classic methods based on packet-pair dispersion filtering, the MDDIF method only requires a minimal possible delay for the first packet and a minimal possible delay for the second packet, where the two delays generally come from different packet pairs. Our proofs and first-passage-time analysis show that the MDDIF method is correct and that it takes less time to obtain accurate samples than the minimum delay sum (MDSUM) method. We have incorporated the MDDIF method into HTTP/OneProbe, a non-cooperative probing method based on TCP data probes and HTTP/1.1, and conducted extensive testbed and Internet experiments to evaluate the MDDIF method.

While most attention and effort have been put on the path capacity measurement for one-way or round-trip network paths, the development of robust methods for measuring capacity asymmetry is still in its infancy. As the second main contribution, we propose TRIO, a non-cooperative method for measuring capacity asymmetry. By using three minimum round-trip times (minRTTs), TRIO can measure both forward-path capacity and reverse-path capacity at the same time. Since TRIO does not measure packet dispersion directly, it removes the packet size limitation and therefore can measure any degree of capacity asymmetry. TRIO also mitigates the cross-traffic interference using the minRTT information and two capacity estimates. Both analytical models and empir-

ical evaluation conducted in a testbed and the Internet report accurate measurement results obtained by TRIO.

One of the fundamental effects of cross traffic is network congestion at routers that will cause packet loss and defeat the measurement of various network path properties (e.g., round-trip delay and path capacity). Notwithstanding this adverse impact, on the bright side, cross traffic can help the formation of loss pairs, which was proposed a decade ago, for discovering other network path properties such as a router's buffer size. A packet pair is regarded as a loss pair if exactly one packet is lost. Therefore, the residual packet's delay could be used to infer the lost packet's delay. Despite this unique advantage shared by no other methods, no loss-pair measurement in actual networks has ever been reported. We first characterize the residual packet's delay by including other important factors (such as the impact of the first packet in the pair) which were ignored before. As a consequence, we invalidate a previous claim that measurement based on the second packet gives the same result as that based on the first. Second, we employ HTTP/OneProbe to measure from a single endpoint all four possible loss pairs for a round-trip network path. We have conducted loss-pair measurement for 88 round-trip paths continuously for almost three weeks. Being the first set of loss-pair measurement in the Internet, we have obtained a number of original results, such as prevalence of loss pairs, distribution of different types of loss pairs, and effect of route change on the paths' congestion state.

Acknowledgments

Many people offered me their wisdom, advice, encouragement, and friendship along my journey into networking research. I would like to give my heartfelt thanks to all of them.

It has been my very good fortune to have had Prof. Rocky Chang as my teacher and advisor over eight years of study at HK PolyU. I am here because of Rocky, who encouraged me to consider the PDoS research for my final year undergraduate project. My deepest respect and most sincere gratitude must go to him, for his patience, professional guidance, unending enthusiasm, and useful critiques and recommendations. He also spared no effort to help me order and organize my every piece of writing. I have learnt from his expertise, rigorous research attitude, and dedication to high-quality scientific research. Rocky, I sincerely thank you.

I am particularly grateful for Daniel Xiapu Luo's mentorship and support throughout my research studies. Daniel has provided me many useful suggestions about doing research and always inspires me to keep moving forward. My grateful thanks are also extended to the past and current members in the Internet Infrastructure and Security Research Group: Grace Xie, Samantha Lo, Kathy Tang, Sam Lam, Steve Poon, Richard Ng, Marko Barthel, Wenchao Zhou, John Kong, Terry Chan, Waiting Fok, Weichao Li,

Sampson Tan, Ricky Mok, Brent Zhou, Ang Chen, and Piotr Dylis.

Moreover, I would like to thank the members of my thesis committee: Prof. Constantine Dovrolis of Georgia Tech, Prof. Yunhao Liu of the Hong Kong University of Science and Technology, and Prof. Henry C. B. Chan of the Hong Kong Polytechnic University for their insightful comments and suggestions on this dissertation and my research.

I am deeply indebted to my dearest wife Summer and my family for their love and understanding throughout these years, and my friends—Daniel Lam, Gloria Hung, Peggy Wong, Ryan Yu, Wallace Chau, and William Li—for their unconditional support.

Last but not least, I thank the anonymous reviewers from the ACM CoNEXT 2009 and ACM/USENIX IMC 2010 for their critical reviews and Paolo Giaccone, in particular, for shepherding my CoNEXT paper. This work is supported by a number of external research grants: two grants (ref. no. ITS/152/08 and ITS/355/09) from the Innovation Technology Fund in Hong Kong, a grant from the Cisco University Research Program Fund, and a grant from the Area of Excellence in Information Technology.

Contents

List of Figures

List of Tables

List of Abbreviations

ADR	Average dispersion rate
ADSL	Asymmetric digital subscriber line
AN	Acknowledgement number in TCP
AQM	Active queue management
BTC	Bulk transfer capacity
CDF	Cumulative distribution function
CWND	Congestion window in TCP
DOCSIS	Data over cable service interface specification
DSL	Digital subscriber line
DWRED	Distributed weighted RED
FAH-ACK	Filling-a-hole ACK
FCFS	First come first serve
FPT	First passage time
HTTP	Hypertext transfer protocol
HTTP/OneProbe	Implementation of OneProbe based on HTTP/1.1
i.i.d.	Independent and identically distributed
ICMP	Internet control message protocol
IP	Internet protocol
IPID	IP identification
IPPM	Internet protocol performance metrics

ISN	Initial sequence number in TCP
ISP	Internet service provider
LBH	Local bottleneck hop
LRD	Long range dependence
MDDIF	Minimum delay difference
MDSUM	Minimum delay sum
MIB	Management information base
minDelay	Minimal possible packet delay
minRTT	Minimum RTT
MSS	Maximum segment size in TCP
MTU	Maximum transmission unit
NAT	Network address translation
ns	Network simulator
OECD	Organisation for Economic Co-operation and Development
OneProbe	Non-cooperative probing method
OOP-ACK	Out-of-ordered-packet ACK
OWAMP	One-way ping
pdf	Probability density function
PLR	Packet loss rate
PNCM	Post-narrow capacity mode
PPD	Packet-pair dispersion
PTD	Packet-train dispersion
RBPP	Receiver-based packet pair
RED	Random early detection

ROHC	Robust header compression
ROPP	Receiver-only packet pair
RTP	Round-trip probe
RTT	Round-trip time
RWND	Receive window in TCP
SBPP	Sender-based packet pair
SCDR	Sub-capacity dispersion range
SLA	Service level agreement
SN	Sequence number in TCP
SNMP	Simple network management protocol
TCP	Transmission control protocol
TCP ACK	Pure TCP acknowledgement packet
TCP FIN	TCP packet with the FIN flag set
TCP RST	TCP packet with the RST flag set
TCP SYN	TCP packet with the SYN flag set
TRIO	Non-cooperative measurement method for asymmetric capacity
TTL	IP time to live
TWP	Two-way probe
UDP	User datagram protocol
VDSL	Very-high-bitrate digital subscriber line
VoIP	Voice over IP
VPS	Variable packet size
WRED	Weighted RED

1

Introduction

Measuring *network path properties* is vital for understanding Internet paths' performance and stability. Network operators measure network capacity and available bandwidth for network planning and design [78, 95] to minimize the discrepancy between unfulfilled customers and underused network resources. Internet Service Providers (ISPs) measure packet loss and delay to monitor the compliance of network service level agreement (SLA) with their customers [208]. End users can also benefit by analyzing their network traffic to identify potential causes of perceived network instability [54, 145] and detecting ISP service discrimination [76, 114, 135, 218, 235].

One of the important network path properties is the *path capacity*

(i.e., smallest link transmission rate of an end-to-end path [78]). Knowing path capacity can improve the performance of many Internet applications and protocols, like multicast and overlay network protocols [34, 65, 107], content delivery applications [195], media streaming applications [62, 166], and those requiring large file transfer [169, 170]. Several available bandwidth measurement methods (e.g. Delphi [190] and Spruce [214]) also require the *a priori* knowledge of the path capacity for computing the available bandwidth estimate.

Unfortunately, designing a robust and reliable path capacity measurement method is very challenging for the Internet landscape today. Since intermediate network devices (e.g., switches and routers) of a network path do not disseminate the link information and it is impracticable to deploy passive monitors to every network link in the path, we often resort to an active measurement method by injecting a sequence of probes to the path and analyzing the subsequent responses. Unlike the measurement for some performance metrics (e.g., available bandwidth [106, 210] and per-hop queueing delay [79]) that estimates the cross-traffic interference from measurement traffic, the path capacity measurement should mitigate the interference to improve the measurement accuracy [78, 115]. Minimizing the overheads, including the amount of measurement traffic and the measurement time, is also essential to make the measurement method usable for a practical network.

Even worse, cross traffic can cause network congestion at routers that will result in packet loss and defeat the measurement of various network path properties (e.g., round-trip delay and path capacity). Notwithstand-

ing this adverse impact, on the bright side, cross traffic can help the formation of *loss pairs* [133], which was proposed a decade ago, for discovering other network path properties such as a router's buffer size. A packet pair is regarded as a loss pair if exactly one packet is lost. Therefore, the residual packet's delay could be used to infer the lost packet's delay. Despite this unique advantage shared by no other methods, no loss-pair measurement in actual networks has ever been reported.

Another challenge is to characterize *asymmetry* properties of forward and reverse paths between a measuring node and a remote endpoint. The forward and reverse paths can possess different characteristics due to many reasons, such as asymmetric data links [75, 213, 223], asymmetric routes [173, 175], and load balancing devices [37, 89]. Measuring asymmetric-path properties is critical for diagnosing performance problems for network applications, because the relative importance of their traffic flowing in each direction can be different [40]. In particular, file sharing and video streaming applications could be dominated by the path capacity and packet loss behavior in one direction [196, 198, 220].

Although measuring asymmetric-path properties can be performed by deploying a measurement tool (e.g., [78, 175, 208]) at both endpoints of a network path, this cooperative approach requires the remote endpoint's cooperation and therefore lacks flexibility to measure arbitrary paths. On the other hand, using *non-cooperative* methods for the measurement offers a potential advantage to scale the measurement to a large number of paths from a single measuring node. However, the capability of existing non-cooperative methods is very limited, because most of

them measure either round-trip path properties or restricted configurations of asymmetric paths [137].

This thesis argues that a measurement method, if carefully devised, can allow a measuring node to achieve a *sound* measurement for network path properties even without the remote endpoint's cooperation. In particular, we first propose a fast and efficient method, called *minimum delay difference* (MDDIF) [50], for path capacity measurement in the midst of cross-traffic interference. Our proofs, analysis, and testbed and Internet experiments show that the MDDIF method is both correct and fast.

While most of the previous research efforts have been put on devising methods for measuring capacity of one-way or round-trip network paths, the development of robust methods for measuring capacity asymmetry is still in its infancy. As the second main contribution, we present a novel method, called *TRIO*, for measuring both forward-path capacity and reverse-path capacity at the same time based on only three minimum round-trip times (minRTTs). TRIO supports any degree of capacity asymmetry, and also mitigates the cross-traffic interference by using the minRTT information and two capacity estimates.

Furthermore, we revisit the loss-pair measurement [51] and present a new method to measure all four possible loss pairs (i.e., two forward-path loss pairs and two reverse-path loss pairs) on a round-trip path. We develop an analytical model to study the information obtained from residual packets. We also present a number of original loss pair results obtained from a recent Internet measurement study.

In the rest of this chapter, we introduce the path capacity measurement and its related problems in Section 1.1 to motivate our designs of the MDDIF method and TRIO. We then present the motivations for the loss-pair measurement in Section 1.2. We finally summarize the contributions of this thesis and its organization in Sections 1.3 and 1.4.

1.1 Path capacity measurement

Path capacity measurement (e.g., [43, 47, 50, 58, 59, 60, 70, 78, 115, 121, 126, 197, 236]) focuses on the smallest link transmission rate of an end-to-end network path from a source to a destination [78]. Besides the path capacity, previous works have also studied the link capacity (e.g., [79, 95, 104, 127, 144, 163]), available bandwidth (e.g., [78, 158, 191, 210]), and bulk transfer capacity (BTC) (e.g., [29, 153]).

Many network applications can benefit from the knowledge of path capacity [95, 197]. Existing path capacity measurement methods mostly exploit packet pairs [43, 47, 50, 58, 78, 115, 121, 197] or packet trains [59, 60, 70, 75, 78, 82, 236]. To compute a capacity estimate, these methods obtain the packet dispersion [43, 47, 58, 70, 78, 95, 115, 121, 126, 128, 164, 197], or delay variation [50, 163] from the measurement traffic. Methods proposed for measuring path capacity are mostly based on active measurement (e.g., [43, 47, 58, 70, 78, 115, 163, 197]), and only a handful based on passive measurement (e.g., [82, 126, 128]). The majority of the active methods are designed to measure one-way capacity (e.g., [78, 174]), round-trip capacity (e.g., [43, 47]), and sub-path capacity [95].

Moreover, methods for the one-way path measurement are usually cooperative, whereas others are mostly non-cooperative.

1.1.1 Mitigating cross-traffic impact

Measuring path capacity is, however, a challenging task in practice, because the accuracy and speed can be adversely affected by cross traffic present on the path under measurement, introducing capacity underestimation or overestimation [78, 115, 164]. To deal with the cross-traffic interference, existing methods usually involve a denoising component to filter out measurement samples that have been biased by cross traffic. The common filtering methods estimate the minimum [58, 115] or mode [43, 47, 78, 126, 128, 163, 164] from the measurement data. The minimum estimator determines the smallest value from a sequence of packet delays as the unbiased data, whereas the mode estimator obtains the unbiased packet dispersion or delay variation that could be neither minimum nor maximum among the measurement data [78, 164].

In particular, Carter and Crovella propose Bprobe [47] which filters inaccurate estimates using union or intersection of packet-pair measurements with different packet sizes. Lai and Baker [126] use a kernel density estimation method to filter capacity estimates. Pásztor and Veitch [164] analyze several types of components embedded in the packet-pair dispersion and select the capacity mode from the high-resolution histogram. Pathrate proposed by Dovrolis *et al.* [78] exploits the capacity distributions obtained by packet pairs and packet trains to infer the cor-

rect capacity mode. Kapoor *et al.* [115] propose the minimum delay sum (MDSUM) method that uses the minimum of packet-pair delay sum to remove distorted packet-pair dispersions. However, these existing filtering methods suffer from slow speed and a large overhead. Notice that injecting a large amount of measurement traffic unnecessarily not only prolongs the estimation process, but also affects the normal traffic and introduces additional processing burdens to both the measuring and remote endpoints.

We therefore propose the MDDIF method for path capacity measurement using packet pairs. Unlike the classic methods that admit a packet pair as the basic unit for capacity measurement, the MDDIF method admits a packet as a basic unit. The MDDIF method obtains minimal possible delay of a first probe packet and a second probe packet, but these two packets do not necessarily belong to the same packet pair. By exploiting useful information in a single packet, the MDDIF method requires less time to obtain accurate capacity estimates and has very low computation and storage costs. The MDDIF method only needs to store and update two minimum packet delay values for the capacity computation.

There are some apparent similarity between the variable-packet-size (VPS) methods [79, 104, 144] and the MDDIF method in terms of requiring minimal possible packet delay. The VPS methods require the minimal possible delay for a sequence of variable-sized packets for measuring a link capacity; the MDDIF method, however, requires the minimal possible delay only for a pair of packets with the same size. As a result, the MDDIF method achieves a faster capacity estimation with a lower

storage requirement.

1.1.2 Measuring capacity asymmetry

It has been reported [75] that the smallest transmission rate is often contributed by the xDSL and cable broadband links. Incidentally, 89% of the broadband subscriptions in the Organisation for Economic Co-operation and Development (OECD) were based on xDSL and cable in October 2009 [23]. Table 1.1 shows the common xDSL and cable technologies with their typical downstream (C_{dn}) and upstream (C_{up}) data rates [22, 223]. Their downstream and upstream data rates are generally asymmetric with C_{dn}/C_{up} ranging between 1.41 and 20. Furthermore, their actual data rates may vary due to the wire quality, transmission distance, and different broadband offerings (e.g., $C_{up}/C_{dn} = 0.512/20$ in [19]). Therefore, we also include in the last two rows the average data rates offered in 30 countries [23].

Measuring asymmetric (instead of round-trip) capacity is useful for many existing applications whose performance is dominated mainly by the capacity of a one-way path [196, 220]. Although the measurement can be conducted by deploying a measurement tool on both endpoints of a path (e.g., [43, 59, 60, 78, 174]), this cooperative approach lacks the flexibility to measure arbitrary paths. We therefore focus on non-cooperative methods that do not require the remote endpoint's cooperation (in terms of setting up additional software). However, among the existing non-cooperative methods (e.g., [47, 70, 104, 197]), very few of them [70, 197]

Table 1.1: Upstream and downstream data rates (in Mbits/s) of the common xDSL and cable technologies, and the average rates reported by the OECD in October 2009.

Technologies	C_{up}	C_{dn}	C_{up}/C_{dn}	C_{dn}/C_{up}
xDSL [223]:				
ADSL (ITU-T G.992.1)	0.8	8	0.1	10
ADSL2 (ITU-T G.992.3/4)	1	12	0.08	12
ADSL2+ (ITU-T G.992.5)	1	20	0.05	20
VDSL (ITU-T G.993.1)	6.48	55.2	0.12	8.52
Cable [22]:				
DOCSIS 1.0	9	38	0.24	4.22
DOCSIS 2.0	27	38	0.71	1.41
DOCSIS 3.0 (8 Channels)	108	304	0.36	2.81
Average rates [23]:				
xDSL	2.27	25.47	0.09	11.22
Cable	3.06	14.41	0.21	4.71

can measure capacity asymmetry.

SProbe [197] and DSLprobe [70], both non-cooperative methods, measure capacity asymmetry using the packet dispersion method which was initially proposed for measuring round-trip capacity [121]. The packet dispersion method relies on setting the probe packet much larger (or smaller) than the response packet. This approach, however, introduces two serious limitations to their measurement capability. First, they cannot measure any degree of capacity asymmetry, because the maximum packet size should be limited by the path MTU to avoid packet fragmentation [58]. Second, they generally cannot support all measurement scenarios, because they may not be able to elicit the required packet size from the remote endpoint. For instance, DSLprobe elicits only small TCP RSTs (but not large response packets) from remote ADSL endpoints. Al-

though AsymProbe [58] also uses the packet dispersion method to measure capacity asymmetry, its current implementation [57] requires running a UDP server on cooperative remote endpoints.

In this thesis, we propose a new non-cooperative method, called TRIO, for measuring capacity asymmetry. By exploiting the minRTTs obtained from three specially crafted probe packets, TRIO obtains the forward-path and reverse-path capacity estimates at the same time and mitigates cross-traffic impact on the capacity estimates. Since TRIO does not measure the dispersion directly, it eliminates the packet size restriction encountered by the existing methods and can measure any degree of capacity asymmetry. Moreover, TRIO incorporates three self-diagnosis tests to further improve the measurement accuracy.

1.2 Loss pair

Packet loss behavior in network paths has been studied extensively for the last twenty years. Most of the efforts focus on packet losses as a result of congestion at routers. The packet loss behavior has been characterized by loss rates (e.g., [43, 174]), loss stationarity (e.g., [234]), loss episodes (e.g., [209, 234]), and loss correlation (e.g., [161, 231]). Both active (e.g., ZING [26], Sting [198], Badabing [207, 209], OneProbe [137], and Queen [225]) and passive (e.g., [30, 168]) measurement methods have been proposed for measuring losses on end-to-end paths. For active methods, the probing process is an important consideration for minimizing measurement errors [38, 39, 217]. Moreover, a number of network tomography

techniques have been proposed for measuring packet losses on the link level [68, 74, 80].

Besides the packet loss measurement, it is also useful to study the correlation between loss and other important metrics. However, the correlation problem has so far received much less attention. A notable exception is using a packet pair to correlate a packet loss event and the delay that would have been experienced by the lost packet. A packet pair is referred to as a loss pair [132, 133] if exactly one packet (the first or second) in the pair is lost. If the two packets traverse the path close to each other, then the residual packet's delay could be used to infer the lost packet's delay. Thus, a correlation between the packet loss and the lost packet's delay could be measured. The loss-pair analysis was originally motivated by the problem of estimating buffer size of the congested node responsible for dropping the packet. Other possible applications of the lost-pair measurement include characterizing packet dropping behavior [133], classifying the type of packet loss [134], detecting dominant congestion links [227], and detecting common congestion points [93, 194].

Although loss pairs could be considered rare events in typical network paths, they can be detected by many existing measurement methods without extra cost. For example, the path capacity measurement methods send a sequence of packet pairs to capture the packet dispersion from the bottleneck link. But they usually discard the loss pairs, which fail to provide the dispersion information. Other measurement methods for packet loss (e.g., [145, 198]), packet reordering (e.g., [41, 138]), Internet traffic characterization (e.g., [61]), and path fingerprinting (e.g.,

[205]) also send packet pairs for their measurement. Therefore, loss-pair measurement is considered a bonus feature for these tools, and the previously regarded useless probes can now be exploited for discovering additional path properties.

However, loss pairs have not been reported in actual network path measurement. In [132, 133, 134], only ns-2 simulation and emulated testbed experiments were performed to evaluate the effectiveness of using loss pairs to measure path properties. As a result, the behavior of loss pairs in Internet paths is largely unknown. Moreover, some important delay components, such as the impact of the first packet on the second, have not been taken into consideration.

On the other hand, a number of methods have been proposed for monitoring congested network links, including Pathload [106] and Pong [73] that detect network congestions by observing increasing queueing delays of its probe packets. Besides, various methods [93, 194, 227] have been proposed to detect the shared network congestion point in the paths. However, these methods are either based on simulation or cooperative measurement.

In this thesis, we revisit loss-pair measurement and show that the delay of the residual packet includes other important factors (such as the impact of the first packet in the pair). Second, we conducted the first set of loss-pair measurement in the Internet for 88 round-trip paths continuously for almost three weeks. As a consequence, we obtained a number of original results from the measurement, including the prevalence of loss pairs, distribution of different types of loss pairs, and effect of route

change on network paths' congestion state.

1.3 Contributions

The contributions of this thesis are as follows:

1. We have investigated the cross-traffic impact on packet pairs for path capacity measurement and proposed the MDDIF method to estimate packet-pair dispersions (PPDs) accurately and efficiently.

   We have established a deterministic model for deriving necessary and sufficient conditions for the first and second packets' minimal possible packet delay (minDelay). Based on these conditions, we have proved that the MDDIF method correctly estimates both forward-path and reverse-path PPDs on a round-trip path with any number of hops in the forward and reverse paths.

   We have derived an analytical model to compare first passage times (FPTs) between the MDDIF method and the MDSUM method. We have proved that the MDDIF method is always faster than the MD-SUM method, when it is possible to obtain the two minDelays from different packet pairs. We have confirmed our findings by obtaining analytical results under a multiple-hop stochastic network model.

   We have incorporated the MDDIF method into HTTP/OneProbe [137], a non-cooperative probing method based on TCP data probes and HTTP/1.1 [87], and conducted extensive testbed and Internet experiments to evaluate the MDDIF method. The experiment re-

sults show that the MDDIF method can yield a correct estimate with a shorter time than the MDSUM method.

2. We have proposed round-trip probes (RTPs) and two-way probes (TWPs) to generalize the measurement methods used by SProbe, AsymProbe, and DSLprobe for capacity asymmetry. Besides, we have articulated two types of interferences—probe interference and response interference—encountered by RTPs due to the packet size restriction.

   We have presented a new non-cooperative method, called TRIO, for measuring capacity asymmetry. We show that TRIO skillfully incorporates the individual probe packets from RTPs and TWPs for measuring the forward-path capacity and reverse-path capacity at the same time. Unlike the existing methods, TRIO performs the measurements by using only three minRTTs (and another minRTT for self-validation) obtained from the RTPs and TWPs. Our analysis based on a deterministic model shows that TRIO removes the probe and response interferences and correctly estimates both the forward-path and reverse-path PPDs.

   Moreover, we have implemented TRIO in HTTP/OneProbe for measuring all asymmetric-capacity scenarios. By incorporating three self-diagnosis tests, TRIO can filter out artifacts due to packet reordering and loss, and invalid minRTTs and capacity estimates. Our empirical evaluation conducted in a testbed and the Internet shows that TRIO is correct under different capacity-asymmetric paths.

3. We have conducted a more detailed analysis for the residual packets' delays by including the impacts of cross traffic and the first packet of the packet pair. The new analysis invalidates the previous claim that the first and second residual packets give the same result [132, 133]. We instead show that using the first packet's delay is generally more accurate than the second packet's delay on inferring the congested router's queueing delay upon packet loss. Moreover, we show that the delay variation of the first and second residual packets can be used to estimate the capacity of a link preceding the congested router.

   We have exploited OneProbe's capability [137] of detecting path events from a single endpoint to measure all four possible loss pairs on a round-trip path: two for the forward path and the other two for the reverse path. To the best of our knowledge, OneProbe is the first non-cooperative method capable of performing comprehensive loss-pair measurement. Previous loss-pair measurement considered only two possible loss pairs on a round-trip path [132, 133]. We have also utilized OneProbe's facility of packet size configuration to validate that a smaller packet size generally increases the accuracy of delay inference.

   We have conducted loss-pair measurement using HTTP/OneProbe for 88 round-trip paths between eight universities in Hong Kong and 11 PlanetLab nodes in eight countries. Our measurement shows that loss pairs were prevalent in the packet pairs that suffered packet

loss, and a loss-pair analysis can help infer additional properties about the lossy paths. Moreover, we show that loss pairs' delays provide path signatures for correlating multiple path measurements.

1.4 Organization

The rest of this thesis consists of five chapters: Chapter 2 on background and related work, Chapter 3 on mitigating cross-traffic impact on path capacity measurement, Chapter 4 on measuring asymmetric capacity, Chapter 5 on using loss pairs to characterize network path properties, and finally Chapter 6 on conclusions and future work.

In Chapter 2, we will first present relevant background on network measurement and performance metrics. Next, we will review previous works on the packet loss measurement. After that, we will provide a detailed survey on the network capacity measurement and propose a new taxonomy for capacity measurement methods. Lastly, we will discuss the probing methods for existing measurement methods and introduce OneProbe, a reliable probing methods to implement our proposed measurement techniques.

Chapter 3 is devoted to mitigation of cross-traffic impacts on path capacity measurement. In this chapter, we will first present the model and assumptions, and review the classic PPD technique. We will then introduce the MDDIF method and compare its performance with the MDSUM method based on their first passage times. Finally, we will evaluate

the MDDIF method's performance based on testbed and Internet experiments.

Chapter 4 is devoted to non-cooperative asymmetric-capacity measurement. We will first present a measurement model and two types of probes for measuring capacity asymmetry. We will then analyze the properties of the two types of probes, and introduce TRIO and an implementation based on HTTP/OneProbe. We will finally evaluate TRIO's accuracy based on testbed and Internet measurement, and compare TRIO with the existing methods for measuring capacity asymmetry.

In Chapter 5, we will explore the loss pair's capabilities on characterizing network path properties. We will first review the loss-pair measurement method and describe how OneProbe detects the loss-pair events in the non-cooperative measurement environment. We will then analyze the residual packets' delays and relate the results to the problem of estimating the queueing delay at the congested router upon packet drop. We will also report our findings of measuring 88 Internet paths continuously for almost three weeks.

2

Background and Related Work

In this chapter, we discuss prior work related to this thesis. We begin with some background on measuring network path properties and packet loss measurement. Next, we survey how previous studies achieved the network capacity measurement, and introduce a taxonomy of existing capacity measurement methods. In particular, we show that the existing methods can be classified based on their measurement traffic, measurement data, and filtering methods. We finally discuss existing probing methods and introduce OneProbe, a reliable probing method to implement our proposed measurement techniques.

2.1 Measuring network path properties

The ability of measuring network path properties is important for network operators to monitor network SLA [27, 52, 66, 208, 237], choose the best route [69, 103, 146], diagnose performance problems [41, 90, 142, 228], and many others. Knowing network path performance is also a desired goal for end users to quantify the ISP's performance [75, 147], optimize the end-to-end performance [34], deal with identified faults with their providers [145], and detect ISP service discrimination [76, 114, 135, 218, 235]. Researchers have also performed the measurement to characterize Internet paths' properties (e.g., [28, 34, 36, 53, 75, 101, 147, 175, 234]). Moreover, it is important to establish concrete and well-defined performance metrics and to design sound network measurement methods.

2.1.1 Performance metrics

Performance metrics are carefully specified quantities related to the performance and reliability of the network [177]. IETF IP Performance Metrics (IPPM) working group [8] has developed a number of performance metrics, including connectivity [148], packet delay [31, 32, 72], packet loss [33, 124], packet reordering [157], network capacity [63, 152], and packet duplication [222]. Moreover, the group has established three notions for defining samples and statistics on different metrics [177]. A singleton metric is atomic in order to define a measured quantity obtained from a single instance of measurement. A sample metric refers to met-

rics derived from a given singleton metric by taking a number of measurement instances. A statistical metric is statistic derived from a given sample metric.

The research community has also studied a variety of performance metrics, including packet loss rate (e.g., [43, 209, 234]), packet reordering rate (e.g., [41, 90, 138, 171]), packet delay (e.g., [151, 156, 172, 187, 192, 200, 226]), network capacity (e.g., [29, 59, 60, 130, 236]), available bandwidth (e.g., [106, 221]), and bulk transfer capacity (e.g., [29, 153]). Some studies have considered correlations among different performance metrics (e.g., [133, 156, 232]), and correlations of a performance metric among network paths (e.g., [160, 201, 230]).

2.1.2 Network measurement methods

A suite of network measurement methods have been proposed during the last several ten years. The methods can be classified into three categories: *active* methods (e.g., [75, 209]), *passive* methods (e.g., [64, 82, 118, 128]), or *hybrid* methods by using a combination of both (e.g., [27, 45, 167, 187]). Detailed surveys on network measurement methods and their pros and cons can be found in [2, 24, 179].

Active methods (e.g., [16, 102, 197, 198, 202]) launched by a measuring endpoint dispatch specially crafted probe packets into a network path under measurement, and capture subsequent responses from a receiving endpoint. The active measurement may involve one or more intermediate nodes to relay probe packets to the destination. Although ac-

tive methods are limited by the measurement resolution (e.g., the probing rate), they can give end-to-end perspective for a target network path. Some previous works (e.g., [186, 199, 203]) provide detailed comparisons of various active methods.

There are a few large-scale measurement methods conducting active measurement, such as Skitter [46], Surveyor [111], AMP [14], NIMI [178], RON [34], Scriptroute [211], Monarch [92], iPlane [142, 143], DipZoom [228], Netdiff [147], neighbor-cooperative measurement system [55], and reverse traceroute [119]. Moreover, PlanetLab [17], Measurement Lab [10], OneLab [21], and GENI [6] are some large-scale testbeds for performing active measurement.

Passive methods, on the other hand, (e.g., [27, 52, 204, 237]) attach monitors to network links and perform passive capture of network traffic passing through the links. The methods collect data from Management Information Base (MIB) counters via Simple Network Management Protocol (SNMP) [48], or capture raw packet traces using `tcpdump` [18] (e.g., [11]). Passive methods can gain very detailed information for the local traffic behavior, at the cost of high storage and processing requirements. However, it is hard to gain end-to-end perspective for a particular network path based on the passive methods, because information exchange among different administrative parties is often infeasible.

Moreover, end-to-end measurement methods (referring to Tables 2.1 and 2.3) can be classified into *cooperative* methods (e.g., [128, 191, 208, 209]) and *non-cooperative* methods (e.g., [71, 150, 198]). The cooperative methods require the control of both a local and a remote measure-

ment endpoints to measure various one-way metrics [212], but the non-cooperative methods measure the round-trip path between the two endpoints without controlling the remote one. The non-cooperative methods have been used in characterizing residential broadband networks [75], Internet coordinate system [131], Internet tomography [67], and studying the impact of routing events on path performance [224].

A major problem with existing non-cooperative methods is that they usually support a very limited number of performance metrics. Since the expected performance from network paths could be different for various applications, it is necessary to measure the network path using as many metrics as possible. There are two specific shortcomings responsible for the current limitation. First, many methods, such as ICMP ping [182], can only measure round-trip path properties. Only few non-cooperative methods can support the measurement of forward-path and reverse-path properties (e.g., sting [198] and POINTER [138]) for a round-trip path. Second, the probing methods for almost all non-cooperative methods (with the exception of tulip [145]) only support one or two types of metrics (e.g., sting for packet loss and POINTER for packet reordering).

A number of previous studies discuss principles for achieving accurate network measurement. Paxson points out a number of strategies to calibrate network measurement to avoid measurement errors in different forms [176]. Sommers proposes a framework and calibration techniques to increase confidence in the validity and accuracy of network measurement methods and the measurements they produce [206]. Roughan [193] derives bounds on accuracy for delay measurements in theoretical

point of view, and shows how these bounds can be used to design both passive and active network measurement experiments.

2.2 Packet loss measurement

Packet loss is a common occurrence in the Internet that is caused by, for example, packets rejected by saturated routers' queue or network intermediates (e.g., firewall), packet corruption [56], denial-of-service attacks (e.g., shrew attacks [125] and pulsing DoS attacks [139]), and loss discrimination mechanisms [114]. Packet loss can produce substantial impact on different protocols or applications. For instance, previous studies show that TCP is sensitive to even a small number of packet losses [140, 165]. Moreover, packet loss can induce users' perceptions of performance degradation. For instance, packet loss has the most severe impact when it occurs at the beginning of compressed voice segments [215], and is also detrimental to compressed video because errors potentially propagate across different frames [85].

As a result, packet loss behavior has been extensively studied. Both active (e.g., ZING [26], OWAMP [16], Sting [198], Tulip [145], Badabing [207, 209], OneProbe [137], and Queen [225]) and passive (e.g., [30, 168]) measurement methods have been proposed for measuring losses on end-to-end paths. On the other hand, various tomography techniques have been proposed for measuring losses on the link level [68, 74, 80]. For active methods, the probing process is an important consideration for minimizing measurement errors [38, 39, 141, 209, 217]. Previous studies

perform active measurement and show that packet loss occurs in bursts of more than one consecutive loss in the Internet [43, 175]. Moreover, the packet loss behavior has been characterized by loss rates [43, 114, 135, 174, 198], loss stationarity [234], loss episodes [209, 234], and loss correlation [161, 231].

While it is useful to study the correlation between loss and other performance metrics, it has so far received much less attention. A notable exception is that Liu and Crovella used packet pairs to correlate a packet loss event and the delay that would have been experienced by the lost packet [133]. A packet pair is referred to as a *loss pair* [133] if exactly one packet (the first or second) in the pair is lost, and a correlation between the packet loss and the lost packet's delay could be measured. Assuming that the pair traverses the path close to each other, they may observe similar states of the congested hop before a packet is discarded. Therefore, the residual packet in the loss pair has been used to estimate the packet dropping mechanism of the congested hop governed by various Active Queue Management (AQM) schemes (e.g., droptail, RED [88], and BLUE [86]) and a droptail queue's buffer size [133], and to classify the causes for packet loss [134]. Other possible applications of the lost-pair measurement include characterizing packet dropping behavior [133], classifying the type of packet loss [134], and detecting dominant congestion links [227] and common congestion points [93, 194].

2.3 Network capacity measurement

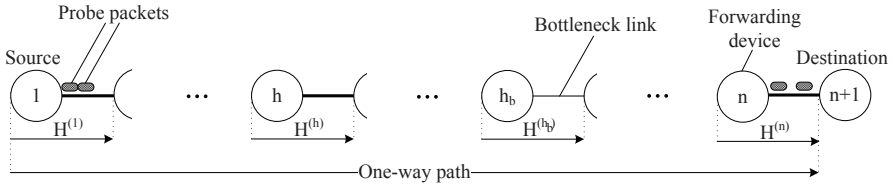
Previous works have studied the following network capacity metrics: *link capacity* (e.g., [79, 104]), *path capacity* (e.g., [43, 47]), *available bandwidth* (e.g., [106, 154, 158]), and *bulk transfer capacity* (BTC) (e.g., [29, 152]). The path capacity is the upper bound of the available bandwidth that defines the maximum bandwidth a network path can provide to a network flow [106]. BTC, on the other hand, defines the amount of data a transport protocol (e.g., TCP) can transfer in the path per unit of time [152]. In the ensuing discussion, we focus on the link capacity and path capacity measurements.

2.3.1 Measurement models

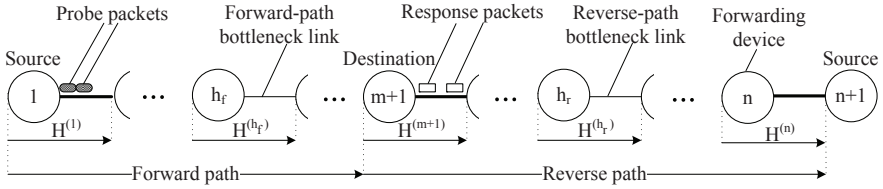
We consider a n-hop network path for capacity measurement. We assume that the path is static and unique throughout the measurement, and use $H^{(h)}$ to denote the h^{th} hop in the path. Each hop consists of a store-and-forward device with a First-Come-First-Serve (FCFS) queue and an outgoing link connecting to the next hop. For convenience, we label the hops on the path sequentially, starting from one at the source node. Therefore, measurement traffic is dispatched at $H^{(1)}$ and traverses towards the end of the path.

The n-hop network path can be either a *one-way path* (where $n \geq 1$) depicted in Figure 2.1(a) or a *round-trip path* (where $n > 1$) in Figure 2.1(b). The one-way path is a single uni-directional (forward) path between the source node and the destination node, whereas the round-

trip path consists of two uni-directional one-way paths—a *forward path* and a *reverse path*—between the two nodes and the destination node at $H^{(m+1)}$ rebounds the measurement traffic back to the source node. Therefore, the first m hops (where $1 \leq m < n$) belong to the forward path and the remaining $(n - m)$ hops to the reverse path.



(a) An n-hop one-way path.



(b) An n-hop round-trip path.

Figure 2.1: Two network path models for network capacity measurement.

2.3.1.1 Capacity metrics

An outgoing link can transfer data at a nominal link capacity depending on the underlying medium for the data transmission [63]. For instance, an 100BASE-T Ethernet link can carry traffic at a nominal link capacity of 100 Mbits/s. However, the actual link capacity depends on the protocol layer in which we are interested (e.g., the IP layer) and is usually *lower* than the nominal link capacity due to additional overheads (e.g., the Ethernet header). If we use IP probe packets of size S_{ip} bits to measure an

Ethernet link at rate C_{eth} bits/s, then each probe packet will be encapsulated by an Ethernet header of size S_{eth} bits. As a result, the IP-layer link capacity is given by [78, 186]

$$C_{ip} = C_{eth}\left(\frac{1}{1+\frac{S_{eth}}{S_{ip}}}\right). \tag{2.1}$$

Going back to our network path models in Figure 2.1, we define $C^{(h)}$ bits/s as the $H^{(h)}$'s link capacity (with respect to the target protocol layer). The path capacity, denoted by $C_b^{(n)}$, is the minimum link capacity in a one-way (round-trip) path:

$$C_b^{(n)} = \min_{h=1,\ldots,n} C^{(h)}. \tag{2.2}$$

We also refer the path capacity to as the *one-way capacity* and *round-trip capacity* for the one-way path and round-trip path, respectively. In the case of a round-trip path, we further define *forward-path capacity* ($C_f^{(n)}$) and *reverse-path capacity* ($C_r^{(n)}$) as:

$$C_f^{(n)} \equiv C^{(h_f)} = \min_{h=1,\ldots,m} C^{(h)}, \tag{2.3}$$

$$C_r^{(n)} \equiv C^{(h_r)} = \min_{h=m+1,\ldots,n} C^{(h)}. \tag{2.4}$$

As Figure 2.1(b) shows, the forward-path capacity and reverse-path capacity are introduced by the bottleneck links at $H^{(h_f)}$ and $H^{(h_r)}$, respectively. Based on Equation (2.2), the round-trip capacity is given by the

minimum of the forward-path capacity and reverse-path capacity:

$$C_b^{(n)} \equiv C^{(h_B)} = \min\left\{C_f^{(n)}, C_r^{(n)}\right\}, \tag{2.5}$$

$$h_B = \begin{cases} h_f, & \text{if } C^{(h_f)} < C^{(h_r)}, \\ h_r, & \text{otherwise.} \end{cases} \tag{2.6}$$

Moreover, when $C_f^{(n)} \neq C_r^{(n)}$, we refer the round-trip path to as a *capacity-asymmetric path*.

In practice, per-hop link capacity measurement (e.g., [104]) estimates the link capacity of each hop along a one-way path. Some previous works (e.g., [113, 118]) propose methods for measuring congested-link capacity. In contrast, path capacity measurement focuses on the bottleneck link's capacity on a one-way path (e.g., [78]), a round-trip path (e.g., [47]), a path segment (i.e., a sub-path) [95], or a forward/reverse path (e.g., [197]). It should be noticed that the path capacity measurement does not attempt to identify the bottleneck link's location, but the per-hop link capacity measurement could infer the location.

2.3.2 Measurement methods

Table 2.1 shows a list of methods for measuring per-hop link capacity, congested-link capacity, sub-path capacity, one-way capacity, round-trip capacity, and asymmetric capacity; and also our methods proposed in the following chapters. To achieve the measurement, the methods require (i) *measurement traffic* (i.e., probe packets introduced by the active methods or legitimate packets exploited by the passive methods) to

probe target network paths, (ii) *measurement data* to compute capacity estimates, and (iii) *filtering methods* to mitigate measurement errors. We propose a taxonomy of capacity measurement methods in Figure 2.2.

As shown in Table 2.1 and Figure 2.2, existing methods exploit four types of measurement traffic—*hop-limited probe* [79, 104, 144], *tailgating probe* [94, 95, 127, 163], *packet pair* [43, 47, 50, 58, 77, 78, 115, 121, 126, 197], and *packet train* [59, 60, 70, 77, 78, 82, 113, 236]—to gain three types of measurement data—*packet delay* [79, 104, 127, 144], *packet dispersion* [43, 47, 58, 70, 77, 78, 82, 95, 115, 121, 126, 128, 164, 197], and *delay variation* (i.e., difference of a pair of packet delays) [163]—for computing capacity estimates. For instance, packet delay of hop-limited probes has been used for measuring link capacity, and dispersion of packet pairs for measuring path capacity. Since dispersion of any two packets is equivalent to their delays' difference [164], we group the packet dispersion and delay variation together in Figure 2.2.

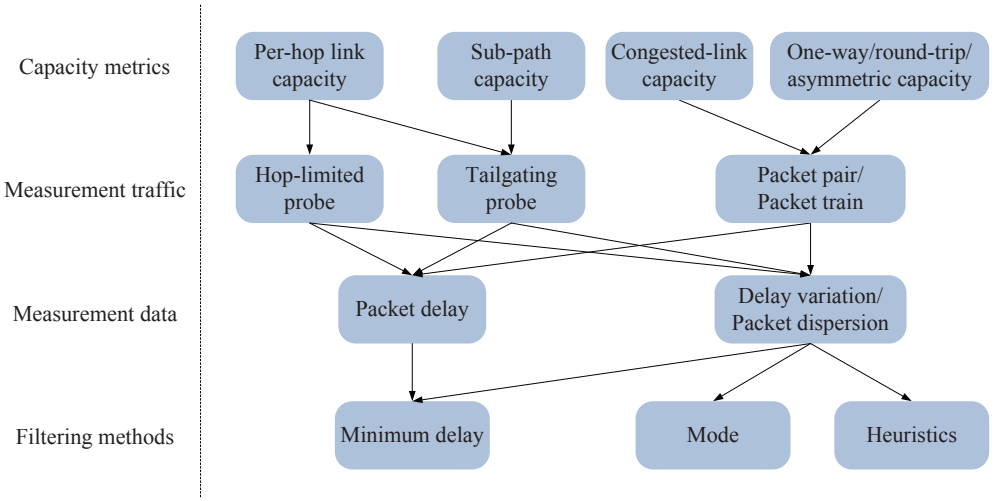


Figure 2.2: A taxonomy of capacity measurement methods.

Table 2.1: A list of capacity measurement methods (A–active, P–passive; NC–non-cooperative, C–cooperative; S–simulation only).

Measurement methods	Measurement traffic	Measurement data	Filtering methods	Probing methods
Per-hop link capacity:				
ACCSIG [163]	Hop-limited probe	Delay variation	Mode (high-resolution histogram)	A; NC
Clink [79]	Hop-limited probe	Delay	Minimum packet delay	A; NC
Nettimer (tailgating) [127]	Tailgating probe	Delay	Minimum packet delay	A; NC
Packet quartet [163]	Tailgating probe	Delay variation	Mode (high-resolution histogram)	A; C
Pathchar [104]	Hop-limited probe	Delay	Minimum packet delay	A; NC
Pchar [144]	Hop-limited probe	Delay	Minimum packet delay	A; NC
Congested-link capacity:				
Envelope [113]	Packet train (envelope)	Dispersion	–	A; C; S
OneProbe (Chapter 5)	Packet pair (loss pair)	Delay variation	Mode (histogram)	A; NC
multiQ [118]	Packet pair	Dispersion	Mode (histogram)	P; NC
Sub-path capacity:				
BBscope [95]	Tailgating probe (cartouche)	Dispersion	Mode (histogram) & Heuristics	A; NC
One-way capacity:				
Nettimer [128]	Packet pair	Dispersion	Mode (kernel density)	P; C/NC
Pathrate [78]	Packet pair & packet train	Dispersion	Mode (histogram) & Heuristics	A; C
Pásztor's method [164]	Packet pair	Delay variation	Mode (high-resolution histogram)	A; C
PBM [174]	Packet pair & packet train	Dispersion	Mode & Heuristic	A; C
PPrate [82]	Packet pair & packet train	Dispersion	Mode (histogram) & Heuristics	P; NC
Round-trip capacity:				
Bolot's method [43]	Packet train	Delay variation	Heuristics	A; C
Bprobe [47]	Packet pair	Dispersion	Mode (union/intersection of histograms)	A; NC
CapProbe [115]	Packet pair	Dispersion & delay	Minimum packet delay	A; NC
PingPair [180]	Packet pair	Dispersion & delay	Minimum queueing delay & mode (histogram)	A; NC
Asymmetric capacity:				
AsymProbe [57, 58]	Packet pair	Dispersion & delay	Minimum packet delay	A; C
DSLprobe [70]	Packet train	Dispersion	Heuristics	A; NC
MDDIF (Chapter 3)	Packet pair	Delay	Minimum packet delay	A; NC
SProbe [197]	Packet pair	Dispersion	Heuristics	A; NC
TRIO (Chapter 4)	Packet pair	Delay	Minimum packet delay	A; NC

The measurement accuracy could be deteriorated by various forms of errors in the measurement data. Such errors can be produced by cross traffic or other measurement artifacts such as packet loss and reordering, thus introducing additional queueing delay to the measurement traffic [141] or causing misconceptions about the measurement data [176]. To mitigate the errors, existing methods are usually augmented by a filtering component to estimate *minimum* (e.g., [104, 115, 127]) or *mode* (e.g., [47, 78, 82, 126, 163, 164]) from the measurement data, or to analyze the data based on *heuristics* (e.g., [70, 197]).

The minimum estimator [79, 104, 127, 144] is commonly found in the methods based on hop-limited probes and tailgating probes. The methods obtain the minimum delay from a sequence of probes, assuming that the minimum delay has precluded the cross-traffic-induced queueing delay. However, the same technique cannot be applied directly to measured packet dispersions or delay variations, because their unbiased values are usually non-minimum [78, 164].

Only a handful of packet-pair methods have exploited the minimum estimator to mitigate the cross-traffic interference. To measure path capacity, the minimum-delay-sum (MDSUM) method [115] employed by CapProbe [115] and AsymProbe [58] acquires minimum packet-pair delays and computes their sum to validate a sequence of observed packet-pair dispersions. PingPair [180] alternatively uses minimum packet-pair queueing delays for the validation. On the other hand, only the MDDIF method (Chapter 3) and TRIO (Chapter 4) derive the path capacity from the minimum packet-pair delays.

The mode estimator [43, 47, 77, 78, 82, 126, 128, 163, 164] filters out biased packet dispersions, delay variations, or corresponding capacity estimates by using histograms [47, 77, 78, 82, 163, 164] or kernel density estimators [126, 128]. Such technique assumes that the unbiased data will tend to cluster closely together [47, 128]. In practice, since the mode representing the path capacity may not prevail in the (multimodal) distribution, Pathrate [78] also applies a heuristic method to infer the path capacity from a set of local modes. Other heuristic methods tackle deceptive probes by analyzing the dispersion, sequence number, and IP identifier (IPID) of response packets [70, 197].

Next, we discuss the four types of measurement traffic and the corresponding measurement methods. Our discussion is based on a network path scenario illustrated in Figure 2.3.

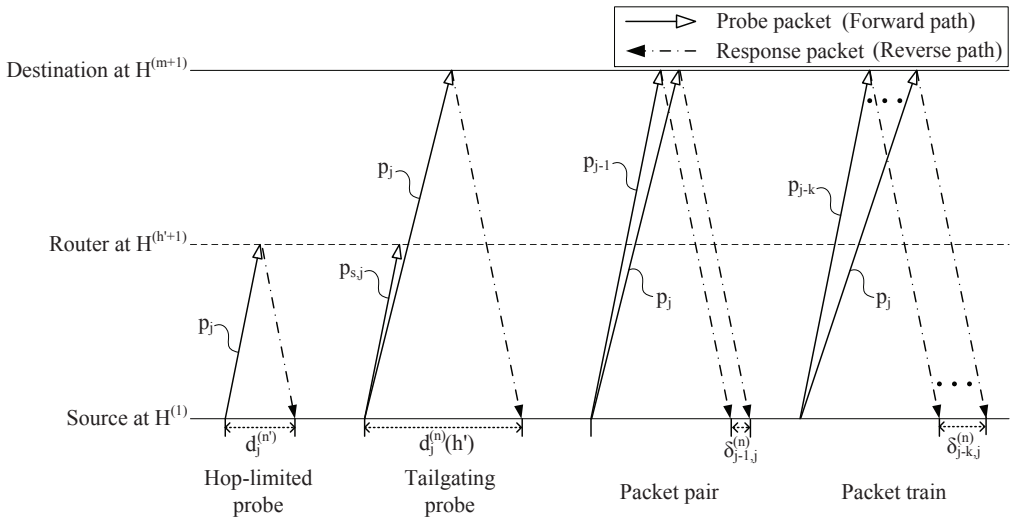


Figure 2.3: Four types of measurement traffic for capacity measurement.

2.3.2.1 Hop-limited probe

The *hop-limited probe* is proposed for measuring per-hop link capacity along a one-way (forward) path from a source node to a destination node [104]. As shown in Figure 2.3, the source node dispatches a hop-limited probe p_j to the path, where p_j is an IP packet with packet size S_j and a time-to-live (TTL) value of h' (where $1 \leq h' \leq m$). Similar to traceroute, the TTL value determines the maximum number of router hops the probe packet can traverse along the path. Therefore, the router at $H^{(h'+1)}$ drops the probe packet and returns an ICMP time-exceeded packet of size S_{icmp} back to the source node. For convenience, we denote the hops on the ICMP packet's returning path as $\{H^{(h'+1)}, \ldots, H^{(n')}\}$ (where $n' \leq n$).

Accordingly, the round-trip time (RTT) of p_j is given by

$$d_j^{(n')} = \sum_{h=1}^{h'} \frac{S_j}{C^{(h)}} + \sum_{h=1}^{h'} w_j^{(h)} + \sum_{h=h'+1}^{n'} w_{icmp}^{(h)} + \zeta, \tag{2.7}$$

where $S_j/C^{(h)}$ is the transmission delay of p_j at $H^{(h)}$, $w_j^{(h)}$ ($w_{icmp}^{(h)}$) is the queueing delay of p_j (the ICMP packet) at $H^{(h)}$, and $\zeta = \sum_{h=1}^{n'} T^{(h)} + \sum_{h=h'+1}^{n'} S_{icmp}/C^{(h)}$ contains two constant terms: (i) propagation delay ($\sum_{h=1}^{n'} T^{(h)}$) on the round-trip path and (ii) transmission delay of the ICMP packet ($\sum_{h=h'+1}^{n'} S_{icmp}/C^{(h)}$) on the returning path. It has been assumed that the transmission delay linearly varies with the packet size and the propagation delay remains constant for various packet sizes.

Variable packet size (VPS) methods (also known as one-packet meth-

ods [127, 197])—Pathchar [104], Clink [79], and Pchar [144]—exploit the linear relationship between S_j and $d_j^{(n')}$ in Equation (2.7) to estimate the link capacity. The methods send a sequence of hop-limited probes with different probe packet sizes (e.g., 45 for pathchar with the MTU of 1500 bytes [126, 163]), and plot the packet sizes against the measured RTTs. To avoid the cross-traffic-induced queueing delay (i.e., $w_j^{(h)}$ and $w_{icmp}^{(h)}$), the methods acquire the minimum RTT (minRTT) for each packet size, and then estimate the slope $\sum_{h=1}^{h'} 1/C^{(h)}$ using the least squares line approximated from the minRTTs. The estimate of $C^{(h')}$ can be computed using the slope difference for h' and $h'-1$.

Unlike the VPS methods, ACCSIG [163] performs the per-hop link capacity measurement using delay variation of adjacent, alternative-sized hop-limited probes. Based on Equation (2.7), we can derive the expression of the delay variation for a pair of hop-limited probes $\{p_{j-1}, p_j\}$:

$$d_j^{(n')} - d_{j-1}^{(n')} = \sum_{h=1}^{h'} \frac{S_j - S_{j-1}}{C^{(h)}} + \sum_{h=1}^{h'} \left(w_j^{(h)} - w_{j-1}^{(h)} \right) + \sum_{h=h'+1}^{n'} \left(w_{icmp,j}^{(h)} - w_{icmp,j-1}^{(h)} \right), \tag{2.8}$$

where $w_{icmp,j}^{(h)}$ is the $H^{(h)}$'s queueing delay experienced by the ICMP packet induced by p_j. Adjacent probes are sufficiently spaced out to ensure that the succeeding probes will never queue behind the preceding probes. By alternating the packet sizes for adjacent probes with a minimum value and a maximum value, the delay variation distribution will show symmetric peaks at $\pm(S_j - S_{j-1}) \sum_{h=1}^{h'} 1/C^{(h)}$, with symmetric noise surrounding the peaks due to cross-traffic-induced queueing delay. Accordingly, ACC-

SIG estimates the quantity $\sum_{h=1}^{h'} 1/C^{(h)}$ by producing a high-resolution delay variation histogram, and computes the difference between the quantities for h' and $h'-1$ to estimate $C^{(h')}$.

There are a number of known issues about the methods using the hop-limited probe. First, TTL-limited IP packets are susceptible to store-and-forward layer-2 devices which introduce additional serialization delay to the packets [163, 184, 185]. Besides, the (long) measurement duration depends on the number of hops on the forward path, the per-hop round-trip delays, the number of probe packet sizes, and the number of repeated probe packets for a particular packet size. Therefore, a significant amount of measurement traffic can be introduced to the network path. For example, Pathchar sends 10 MB of measurement traffic and utilizes more than 60% of the available bandwidth to probe a single-hop 10 Mbits/s Ethernet path with latency of 1 ms [126]. Though Downey [79] shows that the adaptive data collection could save many probe packets for each particular size, the methods still require a large number of packet sizes to yield a better estimate. Thorough discussions of their weaknesses have been given in [110, 126, 185].

2.3.2.2 Tailgating probe

The *tailgating probe* is proposed by Lai and Baker [127] to reduce the number of variable-sized probe packets for the per-hop link capacity measurement. As shown in Figure 2.3, a tailgating probe involves a probe packet $p_{s,j}$ of size $S_{s,j}$ closely followed by another probe packet p_j of size S_j, where $S_{s,j}/S_j = S_{max}/S_{min}$. By setting a TTL value of h' for $p_{s,j}$ and

$S_{max} \gg S_{min}$, p_j is likely to queue behind $p_{s,j}$ at $H^{(h')}$ and at no later hops on the forward path. Assuming that p_j can elicit a response packet with the same size S_j from the destination node, the RTT of p_j (denoted by $d_j^{(n)}(h')$) is given by

$$d_j^{(n)}(h') = \frac{S_{s,j}}{C^{(h')}} + (S_{s,j} - S_j) \sum_{h=1}^{h'-1} \frac{1}{C^{(h)}} + S_j \sum_{h=1}^{n} \frac{1}{C^{(h)}} + \sum_{h=1}^{n} T^{(h)}. \qquad (2.9)$$

Equation (2.9) assumes the same arrival time for $p_{s,j}$ and p_j at $H^{(1)}$ and the absence of cross-traffic interference on the round-trip path.

To compute the estimate of $C^{(h')},\ h' = 1, \ldots, m$ using the packet tailgating technique, Nettimer [127] involves a sigma phase and a tailgating phase to estimate $\sum_{h=1}^{n} 1/C^{(h)}$, $\sum_{h=1}^{n} T^{(h)}$, $d_j^{(n)}(h')$, and $\sum_{h=1}^{h'-1} 1/C^{(h)}$ in Equation (2.9). The sigma phase employs variable-sized probe packets (similar to the VPS methods) to estimate the end-to-end quantities $\sum_{h=1}^{n} 1/C^{(h)}$ and $\sum_{h=1}^{n} T^{(h)}$. The tailgating phase obtains the minimum $d_j^{(n)}(h')$ for $h' = 1, \ldots, m$. Moreover, $\sum_{h=1}^{h'-1} 1/C^{(h)}$ can be determined based on the capacity estimates for previous links.

Packet quartet [163] exploits delay variation of successive, sufficiently spaced out tailgating probes to measure the per-hop link capacity. This method involves five variants—PQ1, PQ2, PQ3, PT1, and PT2—based on different h', $S_{s,j}$, and S_j. The delay variation expressions for the five variants can be derived using Equation (2.9). To avoid the cross-traffic interference, Packet quartet obtains the mode from a high-resolution delay variation histogram for computing the capacity estimate.

To achieve a valid tailgating probe (such that p_j queues behind $p_{s,j}$ at

$H^{(h')}$), the following inequality must be satisfied [95]:

$$\frac{C^{(h')}}{\min_{h=1,\ldots,h'-1} C^{(h)}} \leq \frac{S_{max}}{S_{min}}. \tag{2.10}$$

For instance, given $S_{max}/S_{min} = 1500/40$ (where S_{max} is upper bounded by an MTU of 1500 bytes) and disregarding lower-layer overheads (e.g., Ethernet header size), the probe becomes invalid if $C^{(h')} = 1$ Gbit/s and the path capacity for $\{H^{(1)}, \ldots, H^{(h'-1)}\}$ is less than 26.7 Mbits/s.

BBscope [95] employs *cartouches*, a variation of the tailgating probe, to measure sub-path capacity. We denote $C_b^{(u,v)} = \min_{h=u,\ldots,v} C^{(h)}$ as the sub-path capacity for $\{H^{(u)}, \ldots, H^{(v)}\}$ (where $1 \leq u < v \leq m$) of length $l = v - u + 1$. A cartouche consists of $r + 1$ back-to-back tailgating probes $\{p_{s,j-r}, p_{j-r}, \ldots, p_{s,j}, p_j\}$, in which p_{j-r} and p_j can reach the remote endpoint and the others with a TTL value of h' (where $h' \leq m$) can traverse up to $H^{(h')}$. For each j, $p_{s,j}$ and p_j are of sizes S_{max} and S_{min}, respectively. If p_{j-r} and p_j elicit a response packet respectively from the remote endpoint and there is no cross traffic on the round-trip path, the response packets' dispersion observed by BBscope will be given by

$$\delta_{j-r,j}^{(n)} = \frac{r(S_{max} + S_{min})}{C_b^{(1,h')}}. \tag{2.11}$$

BBscope dispatches a sequence of cartouches and computes corresponding estimates for $C_b^{(1,h')}$ based on Equation (2.11). It obtains the mode from the capacity distribution to avoid the cross-traffic interference. To measure $C_b^{(u,v)}$, BBscope computes both $C_b^{(1,u-1)}$ and $C_b^{(1,v)}$ and

obtains $C_b^{(u,v)} = C_b^{(1,v)}$ if $C_b^{(1,u-1)} > C_b^{(1,v)}$. Otherwise, BBscope exploits a cartouche train [95], which is a sequence of l overlapping cartouches, for the measurement.

Moreover, the following must be fulfilled in order for Equation (2.11) to become valid [95]:

$$\frac{C_b^{(1,h')}}{C_b^{(h'+1,n)}} \leq \frac{r(S_{max} + S_{min})}{S_{min}}. \tag{2.12}$$

Otherwise, $\delta_{j-r,j}^{(n)}$ will be expanded by the bottleneck link in the sub-path $\{H^{(h'+1)}, \ldots, H^{(n)}\}$, thus underestimating $C_b^{(1,h')}$.

2.3.2.3 Packet pair and packet train

Table 2.1 shows that the existing methods mostly use *packet pairs* [47, 58, 115, 128, 164, 180, 197], *packet trains* [70], or both [78, 82, 174] to measure path capacity. According to Figure 2.3, the source node dispatches $k+1$ back-to-back probe packets $\{p_{j-k}, \ldots, p_j\}$ of fixed size S_f to the network. We denote the sequence of packets as a packet pair when $k = 1$, and a packet train when $k > 1$. For the round-trip path, each probe packet also elicits a response packet of size S_r from the destination node. It is convenient to regard the j-th response packet as the probe packet p_j "bounced back" from the destination node towards the source node. Therefore, we also use p_j to refer to the j-th response packet. We assume $S_f = S_r = S$ unless stated otherwise.

We first consider the measurement using packet pairs. Without any cross traffic in the path, *packet-pair dispersion* (PPD) of $\{p_{j-1}, p_j\}$ at the

outgoing link of $H^{(n)}$ is unbiased and equal to the transmission delay introduced by the bottleneck link [78]:

$$\delta_{j-1,j}^{(n)} = \frac{S}{\min_{h=1,\ldots,n} C^{(h)}} = \frac{S}{C^{(h_B)}} \equiv \frac{S}{C_b^{(n)}}. \tag{2.13}$$

Figure 2.4 shows a graphical illustration of the packet-pair measurement using a similar fluid model as in [78, 105]. Without any cross traffic, the packet pair must queue one after another at the bottleneck hop, leave this hop with the largest time gap of $S/C_b^{(n)}$, and preserve this gap in the remaining hops. Therefore, the capacity estimate can be computed as $S/\delta_{j-1,j}^{(n)}$.

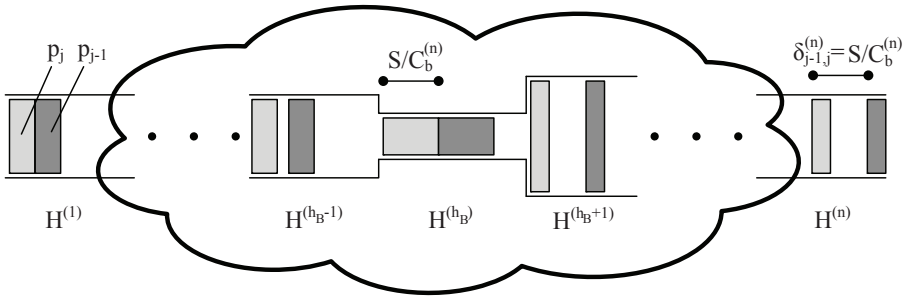


Figure 2.4: Measuring path capacity $C_b^{(n)}$ with a packet pair $\{p_{j-1}, p_j\}$.

Moreover, the PPD can be derived from *delay variation* of p_{j-1} and p_j, because [43, 164]

$$\begin{aligned} d_j^{(n)} - d_{j-1}^{(n)} &= t_j^{(n+1)} - t_{j-1}^{(n+1)} - (t_j^{(1)} - t_{j-1}^{(1)}), & (2.14) \\ &= \delta_{j-1,j}^{(n)} - \delta_{j-1,j}^{(0)}. & (2.15) \end{aligned}$$

where $t_j^{(h)}$ is the arrival time of the p_j's last bit at $H^{(h)}$, or equivalently, the departure time of the last bit at $H^{(h-1)}$.

The seminal work by Bolot [43] shows that $S/C_b^{(n)}$ can be estimated using a scatter plot for adjacent RTTs $d_{j-1}^{(n)}$ and $d_j^{(n)}$, because some data points on the scatter plot will form a line given by Equation (2.15). As a result, we can obtain the x-intercept (i.e., $\delta_{j-1,j}^{(0)} - S/C_b^{(n)}$) of the line to estimate the round-trip capacity.

Bprobe [47] conducts non-cooperative round-trip capacity measurement using the PPD method. Bprobe injects a sequence of ICMP echo requests and measures the PPDs of adjacent ICMP echo responses (where the request and response packet sizes are equal). Since the PPD is likely distorted by cross traffic on the path, Bprobe obtains capacity histograms with various packet sizes, and produces a histogram based on either union or intersection of the capacity histograms. Bprobe finally obtains a capacity estimate by locating the mode from the histogram.

Nettimer [126, 128] captures packets from a legitimate sender to measure one-way capacity. When the sender injects packets with a sending rate lower than the path capacity, the packets will not experience queueing at the bottleneck. Therefore, the capacity estimates derived from their PPDs will be similar to the measured sending rate. By using sender-based packet pairs (SBPPs) [126, 174], Nettimer can detect such bias by computing the ratio between the capacity estimate and the sending rate. To mitigate the cross-traffic interference, Nettimer employs a kernel density estimator to estimate the capacity mode. To ease the measurement deployment, Nettimer also implements receiver-only packet pairs (ROPPs) which do not measure the sending rate.

While Bprobe and Nettimer assume that the actual path capacity is

represented by the greatest density in a capacity distribution, various studies [78, 164, 174] have shown that the distribution can be multimodal and the actual value may not have the greatest density. Dovrolis *et al.* [78] also show that the *sub-capacity dispersion range* (SCDR) and the *post-narrow capacity modes* (PNCMs), both of which deviate from the actual path capacity, can prevail in the distribution depending on the link utilization, cross-traffic packet size, and probe packet size.

Pásztor and Veitch derive a general expression for $\delta_{j-1,j}^{(n)}$ [164]:

$$\delta_{j-1,j}^{(n)} = \frac{S}{C^{(h^*)}} + q_{j-1,j}^{(h^*)} + \sum_{h=h^*+1}^{n} \left(w_j^{(h)} - w_{j-1}^{(h)} \right), \tag{2.16}$$

where $H^{(h^*)},\ 1 \leq h^* \leq n$, is the last hop where p_j arrives before the fully departure of p_{j-1} (i.e., both belong to the same busy period of the queue at $H^{(h^*)}$). The PPD consists of a signature $S/C^{(h^*)}$, which is the transmission delay of a probe packet at $H^{(h^*)}$, and additional "error terms". The first error term $q_{j-1,j}^{(h^*)}$ is the queueing delay experienced by p_j at $H^{(h^*)}$ due to intervening cross traffic between p_{j-1} and p_j, and the second depends on the packets' queueing delay at the hops after $H^{(h^*)}$.

In general, cross traffic can introduce the following impact on the PPD measurement:

1. Cross traffic can expand the dispersion of p_{j-1} and p_j from a *preceding congested hop*, such that they cannot experience the same busy period at the bottleneck hop.

2. Even if the packet pair can experience the same busy period at the

bottleneck hop,

i. cross traffic can increase the queueing delay of p_j (p_{j-1}) at any subsequent hops and therefore expand (compress) the PPD, thus introducing capacity underestimation (overestimation); and

ii. cross traffic can cause the PPD to be over-written by the transmission delay of subsequent hops (i.e., second bottlenecks [78, 126, 164]), thus $h^* > h_B$.

Dovrolis *et al.* [78] show that a lower bound of the path capacity can be derived from the dispersion of a sufficiently long packet train. Denote *packet train dispersion* (PTD) of a packet train $\{p_{j-k}, \ldots, p_j\},\ k > 1$, as $\delta^{(n)}_{j-k,j}$. The path capacity derived from the PTD is referred to as the *average dispersion rate* (ADR) [78] given by

$$R^{(n)}_{j-k,j} = \frac{kS}{\delta^{(n)}_{j-k,j}}. \tag{2.17}$$

Without any cross traffic in the path, it is easy to see that $R^{(n)}_{j-k,j}$ is equivalent to $C^{(n)}_b$ because the sequence of packets will queue one after another at the bottleneck hop, thus leaving with a dispersion of $kS/C^{(n)}_b$. The packet train, however, becomes more susceptible to the cross-traffic interference as k increases. In the case of one-hop persistent cross traffic and a sufficiently large k, the ADR (denoted by $R^{(h)}$) observed from the outgoing link of $H^{(h)},\ h = 2, \ldots, n$, does not depend on the train length, but on the link capacity and the link utilization (denoted by $\rho^{(h)}$) at $H^{(h)}$,

and the ADR from the previous hop. Let $A^{(h)} = C^{(h)}\left(1-\rho^{(h)}\right)$ be the available bandwidth of $H^{(h)}$. Then, $R^{(h)}$ is given by

$$R^{(h)} = \begin{cases} R^{(h-1)}\frac{C^{(h)}}{C^{(h)}\rho^{(h)}+R^{(h-1)}}, & \text{if } R^{(h-1)} \geq A^{(h)}, \\ R^{(h-1)}, & \text{otherwise,} \end{cases} \tag{2.18}$$

and $R^{(1)} = C^{(1)}$ [78]. Moreover, $R^{(n)}$ is a lower bound for the path capacity and an upper bound for the available bandwidth.

Pathrate [77, 78] combines the PPD and PTD methods for measuring one-way capacity. To mitigate the effects of SCDR and PNCMs, Pathrate uses variable-sized packet pairs to probe the target path and obtains a capacity distribution. If the distribution is multimodal, Pathrate probes the path using packet trains with increasing length until a unimodal ADR distribution is observed. Since the ADR mode represents the lower bound of the path capacity, Pathrate chooses the final estimate based on the highest mode after the ADR mode from the capacity distribution.

CapProbe [115] exploits the *minimum delay sum* (MDSUM) method to filter out the biased PPDs. If any packet in a packet pair is queued behind some cross traffic, we will observe a longer delay for this packet (due to the additional queueing delay). Accordingly, the MDSUM method avoids the packet pairs whose delay sums are greater than the minimum. However, such method also discards those pairs in which only *single* packets have been affected by cross traffic. Therefore, we propose the MDDIF method (Chapter 3) which exploits the information obtained from unaffected probe packets to speed up the measurement process.

We notice that cross traffic is sometimes useful. Bolot [43] relies on queueing of the probe packets in the bottleneck hop to measure the bottleneck link's capacity. MultiQ proposed by Katti *et al.* [118] exploits the cross-traffic interference to measure the congested-link capacity. The authors show that the density distribution of the PPDs extracted from TCP packet traces can be modulated by equally-spaced sharp spikes, as a result of various numbers of maximum-sized (i.e., 1500 bytes) packets intervening the measurement traffic at one or more congested links. Therefore, the time gaps between the spikes correspond to the intervening packets' transmission delay at these links and can be used to compute their link capacity. Besides, Kang *et al.* [113] propose the *envelope packet train* to obtain the packet dispersion at a congested link for the measurement. On the other hand, by using the residual packets' delay obtained from loss pairs, we can infer the capacity of a link preceding the congested router (Chapter 5).

Finally, capacity-asymmetric paths are prevalent in the Internet as a result of asymmetric data links (e.g., ADSL [40, 75]), asymmetric routes [173, 175], and using different links for sending and receiving (e.g., load balancing [37, 89] and multi-homing [189]). As shown in Table 2.1, only SProbe [197] and DSLprobe [70] can measure capacity asymmetry with a non-cooperative remote endpoint; both of them exploit the PPD method with different S_f/S_r ratios. AsymProbe [58], while claiming as a sender-only round-trip procedure [58], requires software setup on the remote endpoint [57] to perform the measurement. We leave a more detailed discussion of their mechanisms and weaknesses in Chapter 4.

2.4 Probing methods

Performing an active network measurement in a sound fashion requires a *reliable* probing method. By reliability, we mean the following specific requirements. First, the probing method should work correctly even without controlling the remote endpoint; response packets elicited from different remote endpoints should also be consistent to avoid potential pitfalls that arise from the measurement data obtained from the packets. Second, the measurement result should characterize network path properties experienced by legitimate data packets, rather than by *network control packets* (e.g., ICMP) or *protocol exceptions* (e.g., sending a TCP SYN to a closed remote TCP port to induce a TCP RST). Finally, the method should support the measurement for asymmetric-path (i.e., forward-path and reverse-path) properties with various probe and response packet sizes, sampling rates, and sampling patterns.

2.4.1 Existing probing methods for active measurement

The last column of Table 2.1 reveals that existing capacity measurement methods commonly perform active measurement (e.g., [43, 47, 58, 70, 75, 77, 78, 79, 104, 115, 127, 144, 163, 164, 197]), and only a handful performs passive measurement [82, 118, 126, 128] (see [83] for the comparison of the existing passive methods for capacity measurement). Besides, the probing methods for measuring one-way capacity always require the remote endpoint's cooperation, but the methods for round-trip capacity, sub-path capacity, and per-hop link capacity are mostly non-cooperative.

Moreover, we summarize existing active and non-cooperative capacity measurement methods with their corresponding probing methods in Table 2.2. As shown, the existing methods are mostly based on the probe packets used by ICMP ping (i.e., ICMP echo requests) and traceroute (i.e., TTL-limited IP packets) to induce control packets from the non-cooperative endpoint, or various types of TCP protocol exceptions to induce TCP RST packets.

Table 2.2: Existing active and non-cooperative capacity measurement methods with their corresponding probing methods.

Measurement methods	Probing methods	
	Probe packets	Response packets
Bprobe [47], CapProbe [115], PingPair [180], BBscope [95]	ICMP echo	ICMP echo reply
Pathchar [104], Clink [79], Pchar [144], ACCSIG [163], Packet quartet [163]	TTL limited	ICMP time exceeded
SProbe (forward-path) [197]	TCP SYN	TCP RST
Nettimer (tailgating) [127]	TCP FIN	TCP RST
DSLprobe [70]	TCP data	TCP RST
SProbe (reverse-path) [197]	TCP data	TCP data

Network control packets and protocol exceptions can, however, produce anomalous and unreliable measurements. It has been reported that routers and end hosts do not always respond to ICMP echo and traceroute [84, 136]. According to our recent measurement studies [137] based on a set of 37,874 web servers randomly selected from 241,906 domains, only around 82.7% of them responded to the ICMP echo request. Even when ICMP packets are returned, the measurement results may not be trustworthy, because the ICMP packets and TCP data packets can be processed on different paths in routers [198, 229]. Similarly, while the protocol exceptions worked well in the past, many network intermediaries,

particularly firewalls and IDS/IPS, may now treat these probes and response packets as threats and therefore drop or modify them [91].

Table 2.2 also shows that SProbe [197] is the only method that exploits *legitimate* TCP application sessions for measuring reverse-path capacity. Such approach is more reliable, because it measures the network path experienced by the TCP data packets and can elicit consistent response from the remote endpoint. On the contrary, the SProbe's forward-path measurement is still based on the protocol exception technique.

Besides, the most practiced probing methods for other performance metrics are not reliable according to our definition. As shown in Table 2.3, while the research community has proposed a suite of active measurement methods to diagnose network path performance, many of them [9, 16, 29, 100, 102, 106, 208, 209] support only cooperative measurement. For those non-cooperative methods, almost all of them rely on control packets (e.g., TReno [153] and tulip [145]), protocol exceptions (e.g., SYN test [41] and ABwProbe [71]), or TCP ACKs (pure TCP acknowledgement packets, e.g., Sting [198], abget [35], and POINTER [138]). TCP ACKs are unreliable, because they may not trigger consistent response from the remote endpoint [137] and their packet size cannot be changed. On the other hand, although httping [98] (disregarding the TCP connection time measurement) and TCP data transfer test [41] manipulate TCP data for both probe and response packets, both of them do not aim for asymmetric-path measurement.

We also notice that some methods (e.g., tulip [145] and DSLprobe [70]) manipulate the response packet's IPID for their measurements. Such

Table 2.3: Probing methods for common active measurement methods (NC–non-cooperative, C–cooperative; F–forward path, R–reverse path, RT–round-trip path). The column for "asymmetric-path measurement" is only applicable to non-cooperative methods.

Measurement methods	Probing methods	Probe packets	Response packets	Asymmetric-path measurement
Available bandwidth:				
abget [35]	NC	TCP ACK	TCP data	× (only R)
ABwProbe [71]	NC	TCP data	TCP RST	× (only F)
Cprobe [47]	NC	ICMP echo	ICMP echo reply	× (only RT)
IGI/PTR [100, 102]	C	TCP or UDP	–	–
ImTCP [149, 221]	NC	TCP data	TCP ACK	× (only F)
LinkWidth [49]	NC	TCP SYN and RST	TCP RST	× (only F)
pathChirp [191]	C	UDP	–	–
pathload [106]	C	UDP	–	–
Pathneck [101]	NC	TTL-limited	ICMP time exceeded	× (only F)
TOPP [154]	C	IP	–	–
Bulk transfer capacity:				
cap [29]	C	UDP	UDP	–
Iperf [9]	C	TCP or UDP	TCP or UDP	–
TReno [153]	NC	TTL-limited	ICMP time exceeded	× (only F)
Packet loss:				
Badabing [209]	C	UDP	–	–
NetPolice [235]	NC	TTL-limited	ICMP time exceeded	× (only F)
Sting [198]	NC	TCP data	TCP ACK	✓
ZING [26]	C	UDP	–	–
Packet reordering:				
Dual conn. test [41]	NC	TCP data	TCP ACK	✓
POINTER [138]	NC	TCP ACK and data	TCP ACK and data	✓
Single conn. test [41]	NC	TCP data	TCP ACK	✓
SYN test [41]	NC	TCP SYN	TCP SYN/ACK and RST	✓
TCP data transfer test [41]	NC	TCP data	TCP data	× (only R)
Delay and packet loss:				
ICMP ping [182]	NC	ICMP echo	ICMP echo reply	× (only RT)
DiffProbe [114]	C	UDP	–	–
httping [98]	NC	TCP SYN and data	TCP SYN/ACK and data	× (only RT)
OWAMP [16]	C	UDP	–	–
UDP ping [15]	NC	UDP	ICMP port unreachable	× (only RT)
Delay, delay variation, and packet loss:				
SLAM [208]	C	UDP	–	–
Delay, IP routing path, and packet loss:				
traceroute	NC	TTL-limited	ICMP time exceeded	× (only RT, F)
Packet loss, packet reordering, and queueing delay:				
tulip [145]	NC	ICMP timestamp	ICMP timestamp reply	✓

method requires the remote endpoint to sequentially increment the IPID for each packet sent. However, it has been reported that some operating systems (e.g., FreeBSD and OpenBSD) use randomized IPIDs to avoid an attacker to infer the number of NATted hosts [42]. Likewise, our measurement study found that around 30% of the tested web servers failed to return consecutive IPIDs [137].

2.4.2 OneProbe

We therefore design OneProbe [137] to achieve our measurements. OneProbe is a non-cooperative probing method which uses only TCP data packets as the probe and response for path measurement. OneProbe can tackle the reliable path measurement problem by sending two customized TCP data packets to induce at most two TCP data packets from the remote endpoint in a legitimate TCP application session. Its probing method is capable of measuring multiple path metrics—round-trip delay, probe and response packet loss events, and probe and response packet reordering events—all from the same probe.

The implementation of OneProbe, namely HTTP/OneProbe, is based on HTTP/1.1 [87] due to the prevalence of HTTP servers in the Internet [1, 13]. To perform the path measurement, HTTP/OneProbe sends a sequence of probes in a persistent HTTP connection (over a single TCP connection). Each probe packet contains a legitimate HTTP request, and each response packet contains legitimate data requested by HTTP/OneProbe. HTTP/OneProbe uses HTTP/1.1's request pipelining to facilitate contin-

uous measurement in a persistent HTTP connection, and employs concurrent TCP connections (managed by the POSIX Threads (`pthreads`) library) to support a higher sampling rate and different sampling patterns.

In the remaining of this section, we review the OneProbe's probing method, and refer readers to [137] for HTTP/OneProbe's implementation details.

2.4.2.1 The probing process

We use Figure 2.5 to explain the OneProbe's probing process. Denote a probe packet by $Cm|n$ and a response packet by $Sm|n$. Both packets are TCP data packets, and m and n are the TCP data segment's sequence number (SN) and acknowledgement number (AN), respectively. All the TCP data segments are of full size (i.e., maximum segment size, MSS). Therefore, we simply use $m = 1, 2, \ldots$ to enumerate the server's TCP data segments and $1', 2', \ldots$ OneProbe's TCP data segments. For example, OneProbe sends its fourth data segment in $C4'|2$ that also acknowledges the first two data segments from the server. Moreover, when the AN is not important, we just use Cm and Sm.

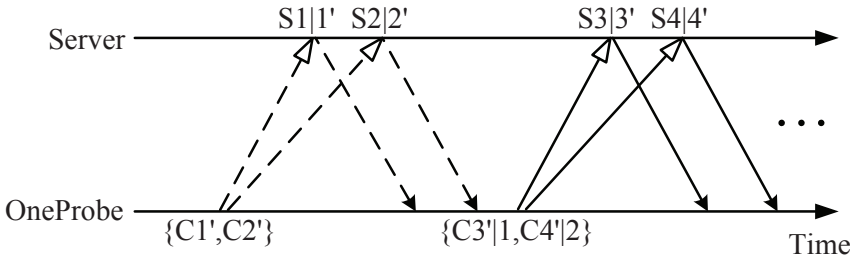


Figure 2.5: Two successive probe rounds in OneProbe.

OneProbe customizes and dispatches the successive probes according to the following three principles:

P1. (Dispatching a new probe) A new probe is dispatched only after receiving two new data segments from the server and the acknowledgment for the data segments in the probe.

P2. (Acknowledging one data segment) Each probe packet acknowledges *only one* data segment from the server, although both have been received by the time of sending the first probe packet.

P3. (Controlling the send window size) The probe packets advertise a TCP receive window (RWND) of two segments in an attempt to constrain the server's TCP send window size to two segments.

Figure 2.5 depicts two successive probe rounds (the first round denoted by dotted lines and the second by solid lines). According to P1, OneProbe sends a new probe of $\{C3'|1, C4'|2\}$ (for a new probe round) after receiving $S1|1'$ and $S2|2'$. We use $\{\cdot\}$ to mean that the packets are consecutively dispatched by the sending host. Therefore, the packet transmissions in the first round do not overlap with that in the next. Moreover, if the server's congestion window size (CWND) is at least two segments, P3 will ensure that its send window size is set to two segments. Finally, based on P2 and P3, the server can send only one new data segment after receiving a probe packet if the probe packets are received in the original order.

2.4.2.2 Detecting packet loss and reordering events

There are five possible path events regarding the two probe packets on the forward path:

F0. Both probe packets arrive at the server with the same order.

FR. Both probe packets arrive at the server with a reverse order.

F1. The first probe packet is lost, but the second arrives at the server.

F2. The first probe packet arrives at the server, but the second is lost.

F3. Both probe packets are lost.

There are also five similar events for the two new response packets on the reverse path: R0, RR, R1, R2, and R3 (by replacing "probe" with "response" and "the server" with "OneProbe" in the list above). As a result, there are 18 possible loss-reordering events, as shown in Table 2.4: the 17 events indicated ✓and one event for F3 (there is no ✓, because this is a forward-path-only event). Others indicated by – are obviously not possible.

Table 2.4: The 18 possible loss-reordering events for the two probe packets and two response packets.

	R0	RR	R1	R2	R3
F0	✓	✓	✓	✓	✓
FR	✓	✓	✓	✓	✓
F1	✓	✓	✓	✓	✓
F2	✓	–	✓	–	–
F3	–	–	–	–	–

OneProbe can detect almost all the 18 path events based on the response packets. Considering the $\{C3'|1, C4'|2\}$ probe in Figure 2.5, Table 2.5 summarizes the response packets induced for the 18 cases based on RFC 793 [183]. In addition to the new data segments 3 and 4, the server may retransmit old data segments 1, 2, and 3, and we use $\widehat{S}m|n$ to refer to a data retransmission. Since the server responses are based on TCP's two basic mechanisms: acknowledgment-clocked transmissions and timeout-based retransmissions, all operating systems are expected to produce the same responses.

Figure 2.5 has already illustrated the event F0×R0; Figure 2.6 ($C1'$ and $C2'$ are omitted) illustrates four other cases: FR×R0, F1×R0, F2×R0, and F3. The rest can be easily constructed from the illustrations for these five events. Note that, because of P1, the server retransmits old data segments in all four cases. The main purpose for withholding a new probe, even after receiving two new data segments (e.g., in the events FR×R0 and F1×R0), is to induce retransmissions for path event differentiation.

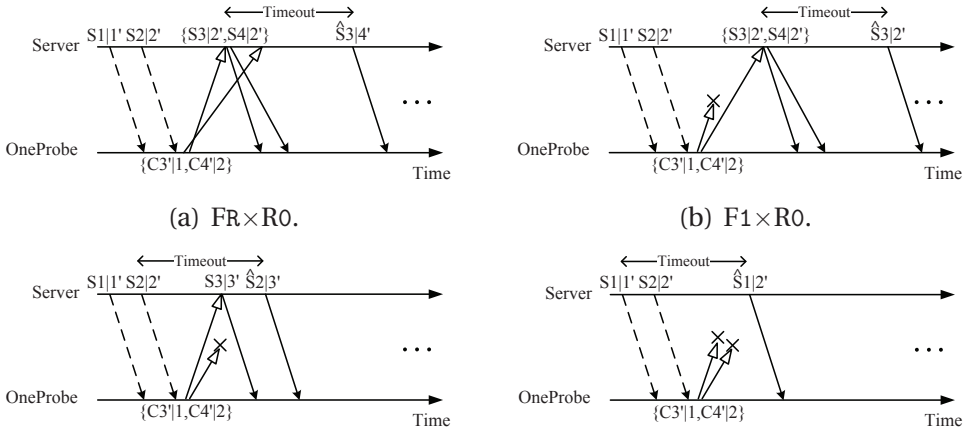
2.4.2.3 Distinguishability of the path events

The different combinations of the SN and AN in the response packets enable OneProbe to distinguish almost all the 18 path events. It is not difficult to see, by sorting Table 2.5 according to the three response packets, that each sequence of the response packets matches uniquely to a path event, except for the following three cases:

A1. F1×R2 and F1×R3: These two events cannot be distinguished based

Table 2.5: The response packets induced by the $\{C3'|1, C4'|2\}$ probe for the 18 path events according to RFC 793.

Path events	1st response packets	2nd response packets	3rd response packets
1. F0×R0	$S3\|3'$	$S4\|4'$	–
2. F0×RR	$S4\|4'$	$S3\|3'$	–
3. F0×R1	$S4\|4'$	$\widehat{S}3\|4'$	–
4. F0×R2	$S3\|3'$	$\widehat{S}3\|4'$	–
5. F0×R3	$\widehat{S}3\|4'$	–	–
6. FR×R0	$S3\|2'$	$S4\|2'$	$\widehat{S}3\|4'$
7. FR×RR	$S4\|2'$	$S3\|2'$	$\widehat{S}3\|4'$
8. FR×R1	$S4\|2'$	$\widehat{S}3\|4'$	–
9. FR×R2	$S3\|2'$	$\widehat{S}3\|4'$	–
10. FR×R3	$\widehat{S}3\|4'$	–	–
11. F1×R0	$S3\|2'$	$S4\|2'$	$\widehat{S}3\|2'$
12. F1×RR	$S4\|2'$	$S3\|2'$	$\widehat{S}3\|2'$
13. F1×R1	$S4\|2'$	$\widehat{S}3\|2'$	–
14. F1×R2	$S3\|2'$	$\widehat{S}3\|2'$	–
15. F1×R3	$\widehat{S}3\|2'$	–	–
16. F2×R0	$S3\|3'$	$\widehat{S}2\|3'$	–
17. F2×R1	$\widehat{S}2\|3'$	–	–
18. F3	$\widehat{S}1\|2'$	–	–



(a) FR×R0. (b) F1×R0.

(c) F2×R0. (d) F3.

Figure 2.6: OneProbe's packet transmissions for the path events FR×R0, F1×R0, F2×R0, and F3.

on the response packets, because $S3|2'$ and $\widehat{S}3|2'$ are identical, and the server may retransmit more than once.

A2. F1×RR and F1×R1: The reasons for their indistinguishability are similar to that for A1.

A3. F0×R3 and FR×R3: Both events have the same response packet $\widehat{S}3|4'$.

The ambiguities in A1 and A2 make the delivery status of $S3|2'$ uncertain. The ambiguity in A3, on the other hand, makes the probe's order of arrival uncertain. Our current implementation disambiguates A1 and A2 by measuring the time required for $S3|2'$ (or $\widehat{S}3|2'$) to arrive. It usually takes a much longer time to receive $\widehat{S}3|2'$, the retransmission of $S3|2'$.

2.4.2.4 Assistance from TCP ACKs

Recall that an important design choice for OneProbe is not to rely on TCP ACKs. However, some ACKs, if received by OneProbe, can assist in detecting the path events. There are two such ACKs: out-of-ordered-packet ACK (OOP-ACK) and filling-a-hole ACK (FAH-ACK). Referring to Figure 2.6(a), the early arrival of $C4'|2$ could immediately trigger an OOP-ACK, whereas the late arrival of $C3'|1$ could immediately trigger an FAH-ACK. According to our measurement, some systems did not return the OOP-ACK, but all the systems tested returned the FAH-ACK.

Even though the system responses regarding the FAH-ACK are uniform, OneProbe still does not rely on it for measurement, because it could be lost. Instead, OneProbe exploits these ACKs, if received, to enhance its measurement. The first is using the FAH-ACK to accelerate the detection

of the forward-path reordering events (i.e., FR$\times*$) without waiting for the data retransmissions. The second is using the FAH-ACK to disambiguate A3 that is the only unresolved case. An arrival of FAH-ACK, in addition to $\widehat{S}3|4'$, clearly signals an FR$\times$R3 event.

2.4.2.5 Starting a new probe round

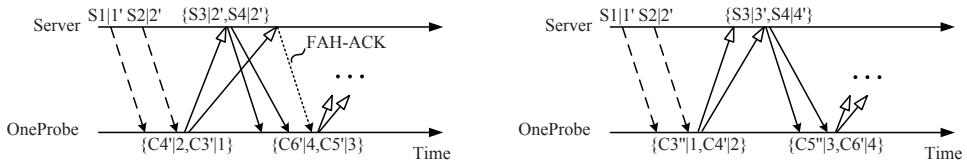
Out of the 18 path events, only the path events 1-2 fulfill the conditions for dispatching a new probe in P1 immediately after receiving two response packets. Moreover, path events 3 and 6-8 fulfill the conditions immediately after receiving a data retransmission. However, the condition is not met for the rest (i.e., events 4-5 and 9-18). Another related problem is that the server's CWND is dropped to one segment for all the path events that involve timeout-based retransmissions (i.e., path events 3-18).

To address the two problems that prevent OneProbe from starting a new probe round, OneProbe will first send one or more new TCP ACKs to increase the server's CWND back to two for path events 3-18. After receiving two new data segments, OneProbe dispatches a new probe: $\{C5', C6'\}$ for events 3-10, $\{C4', C5'\}$ for events 16-17, and $\{C3', C4'\}$ for event 18. Handling events 11-15 is more complicated. If a new probe of $\{C3', C4'\}$ were used, the server will drop $C4'$, because it has already been received. The current implementation restarts the connection when encountering these path events. A better approach is to retransmit $C3'$ with the respective ANs and to use a new probe of $\{C5', C6'\}$.

2.4.2.6 Inducing a pair of back-to-back response packets

Sometimes it is desired to probe the reverse path using a pair of back-to-back response packets dispatched by the server. To this end, OneProbe provides two methods depicted in Figure 2.7. Consider the scenario after receiving $S1|1'$ and $S2|2'$. For the first method depicted in Figure 2.7(a), OneProbe sends a reordered probe of $\{C4'|2, C3'|1\}$ to simulate the event F2×R0 (in Figure 2.6(a)). Moreover, OneProbe can immediately dispatch a new probe after receiving an FAH-ACK (which acknowledges both $C3'$ and $C4'$).

As shown in Figure 2.7(b), OneProbe can also dispatch a probe of $\{C3''|1, C4'|2\}$, in which their SN and AN are same as the $\{C3'|1, C4'|2\}$'s, but $C3''|1$ now has an RWND of zero bytes (which deviates from P3) in an attempt to suppress the server to send $S3|3'$. Therefore, in the absence of packet loss and reordering, the arrival of $C4'|2$ with the RWND of two segments will elicit the server to dispatch $\{S3|3', S4|4'\}$. Moreover, OneProbe can immediately dispatch a new probe after receiving the two response packets.



(a) The $\{C4'|2, C3'|1\}$ probe. (b) The $\{C3''|1, C4'|2\}$ probe.

Figure 2.7: Two probing methods for inducing a pair of back-to-back response packets.

3

Mitigating Cross-Traffic Impact on Path Capacity Measurement

The knowledge of path capacity is useful for many network applications to improve their performance. Measuring path capacity is, however, a challenging task in practice, because the accuracy and speed can be adversely affected by cross traffic, packet loss and reordering events, packet sizes, time resolution supported by measurement endpoints, probing methods, and others. As discussed in Chapter 2, most of the existing path capacity measurement tools are based on the dispersion approach, in particular, the packet-pair dispersion (PPD) method that sends a pair of back-to-back probe packets to measure their dispersion.

To deal with the cross traffic present on the path under measurement, the basic PPD method is usually augmented by a component to filter measurement samples that have been biased by cross traffic. A notable example is the minimum delay sum (MDSUM) method first introduced to CapProbe [115]. The MDSUM method filters out packet pairs that do not meet a minimum delay sum condition. Another, employed by Pathrate, is based on the PPD distribution with different packet sizes [78] and the assistance of packet trains.

However, the existing cross-traffic filtering techniques for the PPD methods still suffer from slow speed and a large overhead. Applications, such as determining optimal software download rates, forming peer-to-peer networks, and establishing multicast trees, will benefit from a fast estimation of network capacity [197]. Moreover, injecting a large amount of probe traffic unnecessarily not only prolongs the estimation process, but also affects the normal traffic and introduces additional processing burdens to both the measuring and remote nodes.

In this chapter, we propose a new technique called minimum delay difference (MDDIF) [50]. Unlike the MDSUM method that admits a packet pair as the basic unit for capacity measurement, the MDDIF method admits a packet as a basic unit. The MDDIF method obtains minimal possible delays of a first probe packet and a second probe packet, but these two packets do not necessarily belong to the same packet pair. By exploiting useful information in a single packet (which is discarded by the MDSUM method), the MDDIF method requires less time to obtain accurate capacity estimates and has very low computation and storage costs. The

MDDIF method only needs to store and update two minimum packet delay values, but the MDSUM method is required to store two more packet delay values and the procedure of validating the minimum delay sum condition is more complicated.

This chapter is organized as follows. In Section 3.1, we present the model and assumptions used throughout this chapter and review the classic PPD method. We then introduce the MDDIF method in Section 3.2 and compare its performance with the MDSUM method based on their first passage times in Section 3.3. In Section 3.4, we further evaluate the MDDIF method's performance based on Internet and testbed experiment results. We discuss the limitations of MDDIF in Section 3.5 and finally conclude this chapter in Section 3.6.

3.1 Model and Preliminaries

3.1.1 The measurement model

We consider the capacity measurement model given in Figure 3.1. A local endpoint (source node) measures path capacity by dispatching a sequence of packet pairs (two back-to-back packets) to a remote endpoint (destination node). Each probe packet elicits a response packet from the remote endpoint. The round-trip network path under the measurement starts from and ends at the local endpoint, consisting of n (where $n \geq 2$) hops. The first m hops (where $1 \leq m < n$) belong to the forward path and the remaining $n - m$ hops to the reverse path. The probe packets travel

on the forward path and the response packets on the reverse path.

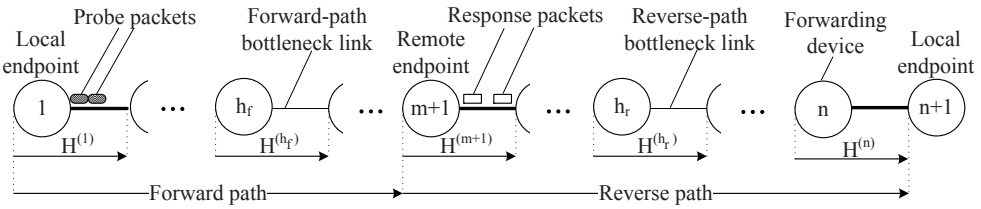


Figure 3.1: The capacity measurement model for a n-hop round-trip network path with a m-hop (where $1 \leq m < n$) forward path and a $(n-m)$-hop reverse path.

Each hop consists of a (local, remote, or forwarding) node and its outgoing link. We use $H^{(h)}$ $(1 \leq h \leq n)$ to denote the h^{th} hop which transmits packets to the outgoing link with a rate of $C^{(h)}$ bits/s. For convenience, we label the hops on the path sequentially. Therefore, the local endpoint belongs to $H^{(1)}$, whereas the remote endpoint belongs to $H^{(m+1)}$. The figure also shows a bottleneck link on the forward path which belongs to a hop denoted by $H^{(h_f)}$, $1 \leq h_f \leq m$. If there are more than one bottleneck hop on the forward path, $H^{(h_f)}$ is referred to the one with the largest h_f. The above applies similarly to the reverse path, where the bottleneck link belongs to a hop denoted by $H^{(h_r)}$, $m+1 \leq h_r \leq n$.

There are three types of path capacity metrics: *forward-path capacity* (denoted by $C_f^{(n)}$), *reverse-path capacity* (denoted by $C_r^{(n)}$), and *round-trip capacity* (denoted by $C_b^{(n)}$), where

$$\begin{aligned} C_f^{(n)} &\equiv C^{(h_f)} = \min_{1 \leq h \leq m} \left\{ C^{(h)} \right\}. \\ C_r^{(n)} &\equiv C^{(h_r)} = \min_{m+1 \leq h \leq n} \left\{ C^{(h)} \right\}. \\ C_b^{(n)} &= \min \left\{ C_f^{(n)}, C_r^{(n)} \right\}. \end{aligned}$$

3.1.2 The assumptions

Unless stated otherwise, we adopt the following assumptions in this chapter:

1. Both the forward and reverse paths are static and unique and do not change during the measurement.

2. The forwarding node in each hop is a store-and-forward device using a FCFS queue.

3. Each probe packet elicits a single response packet from the remote endpoint with negligible delay.

4. All probe and response packets are received successfully. Combining with (1)-(3) also implies that the probe packets arrive at the remote endpoint in the original order, and the response packets arrive at the local endpoint in the original order.

5. The processing delay introduced by the forwarding nodes is small compared with the packet-pair dispersion and therefore negligible.

6. The packet pairs are sufficiently spaced out that a first probe packet is never queued behind the preceding packet pair, and a first response packet is never queued behind the preceding response packets.

Assumptions (1)-(2) are reasonable and have been adopted in previous works [43, 47, 78, 95, 126]. Assumptions (3)-(5) are required to

ensure that the packet-pair dispersion is not biased by the remote endpoint's processing delay, packet loss and reordering events, and forwarding nodes' processing delay. Finally, assumption (6) is valid for adequately spaced packet pairs.

3.1.3 Preliminaries

This section provides preliminary results based on deterministic models for deriving the main results in the next two sections. These preliminary results are not new as they appeared in previous works, such as [58, 127, 164].

In a capacity measurement session, a local endpoint dispatches a sequence of packet pairs $P_i,\ i = 1, 2, \ldots$. Now consider any packet pair $\{p_{j-1}, p_j\}$ in the sequence, where $j = 2i$ indicates the position of the second packet in P_i. We also let S_f and S_r be the sizes of the probe and response packets in bits, respectively. Due to assumption (3), it is convenient to regard the first response packet as the first probe packet "bounced back" from the remote endpoint and similarly for the second response packet. Therefore, we also use p_{j-1} and p_j to refer to the first and second response packets, respectively, but S_f and S_r are generally different.

3.1.3.1 Individual packet delay

Let $d_j^{(h)}$ be the time interval that p_j spends on the first h hops. As illustrated in Figure 3.2, $d_j^{(h)}$ can be defined recursively by

$$d_j^{(h)} = d_j^{(h-1)} + (w_j^{(h)} + X^{(h)} + T^{(h)}) \; for \; h \geq 1, \tag{3.1}$$

and $d_j^{(0)} = 0$ [127]. The delay at $H^{(h)}$ comprises a queueing delay ($w_j^{(h)}$), a constant transmission delay of $X^{(h)}$ ($X^{(h)} = S_f/C^{(h)}$ for $1 \leq h \leq m$ and $X^{(h)} = S_r/C^{(h)}$ for $h > m$), and a constant delay ($T^{(h)}$) for propagating the packet to the next hop. The expression of $d_{j-1}^{(h)}$ for p_{j-1} is the same as Equation (3.1) after updating the subscripts.

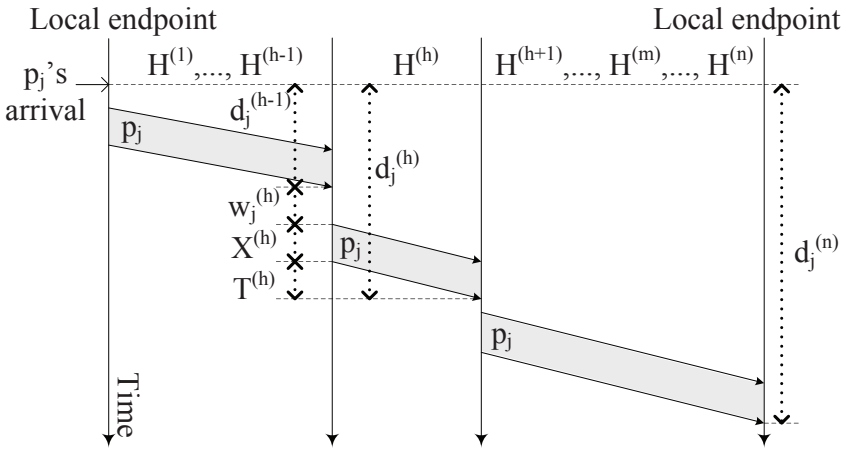


Figure 3.2: The delay components for p_j to traverse the first h hops.

3.1.3.2 Packet-pair dispersion and capacity

The (round-trip) *packet-pair dispersion* (PPD) for $\{p_{j-1}, p_j\}$, denoted by $\delta^{(n)}_{j-1,j}$, is given by [43, 164]

$$\delta^{(n)}_{j-1,j} = d^{(n)}_j - d^{(n)}_{j-1} + \tau^{(1)}_{j-1,j}, \tag{3.2}$$

where $\tau^{(1)}_{j-1,j}$ is the inter-arrival time for p_{j-1} and p_j at $H^{(1)}$. Without loss of generality, we let $\tau^{(1)}_{j-1,j} = 0$. However, $\tau^{(h)}_{j-1,j}$ is generally non-zero for $h > 1$.

If $\{p_{j-1}, p_j\}$ are both unaffected by the cross traffic on the path, their PPD is unbiased and is given by [58]

$$\delta^{(n)}_{j-1,j} \equiv X^{(h_b)} = \max\left\{X^{(h_f)}, X^{(h_r)}\right\}, \tag{3.3}$$

where

$$h_b = \begin{cases} h_f, & \text{if } X^{(h_f)} > X^{(h_r)}, \\ h_r, & \text{otherwise.} \end{cases} \tag{3.4}$$

Since $X^{(h_f)} = S_f/C^{(n)}_f$ and $X^{(h_r)} = S_r/C^{(n)}_r$, the path capacity computed based on $\delta^{(n)}_{j-1,j}$ can give $C^{(n)}_f$ $(= S_f/X^{(h_f)})$ or $C^{(n)}_r$ $(= S_r/X^{(h_r)})$. For the case of $S_f = S_r$, the computation gives $C^{(n)}_b = S_f/X^{(h_b)} = S_r/X^{(h_b)}$.

3.2 Mitigating cross-traffic interference

Previous studies [155, 174] have shown that the PPD could be distorted by two types of cross traffic. Consider Figure 3.3 that depicts a scenario of

measuring path capacity for a three-hop network path with three packet pairs. The first type of cross traffic is the traffic already existing in a forwarding node after the bottleneck link when p_1 arrives. This cross traffic delays p_1 to the extent that the PPD is compressed, causing a capacity overestimation. The second type is the traffic intervening $\{p_3, p_4\}$ in a forwarding node. This cross traffic increases p_4's queueing delay, thus causing a capacity underestimation [78]. On the other hand, $\{p_5, p_6\}$ are both unaffected by cross traffic throughout the path and therefore produce a correct PPD.

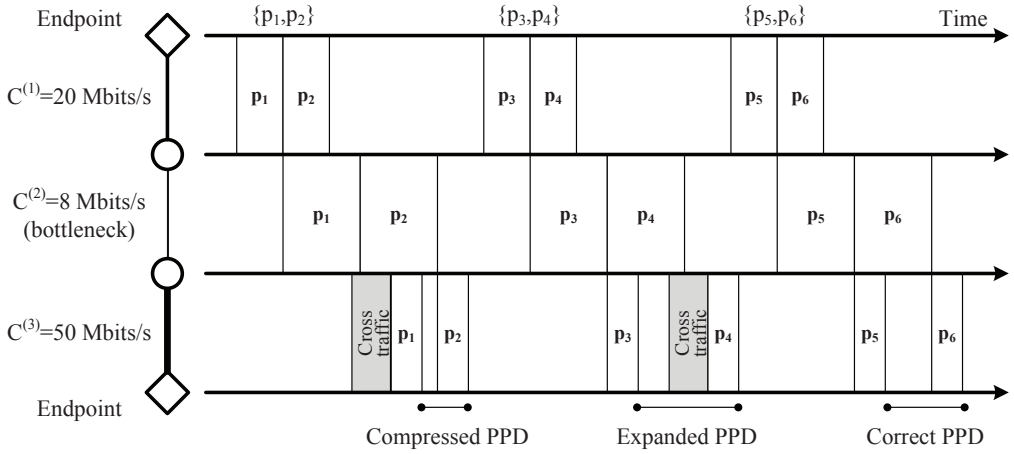


Figure 3.3: Measuring path capacity for a three-hop network path with three packet pairs.

3.2.1 Minimum delay sum

A minimum delay sum (MDSUM) method is proposed to remove distorted PPDs for, such as, CapProbe [115] and AsymProbe [58]. The basic idea is that if any packet in a packet pair is interfered by cross traffic, additional packet delay will be introduced; therefore, a sum of the two pack-

ets' delay (i.e., a delay sum) will also increase. To implement this idea, CapProbe dispatches a sequence of packet pairs until a P_i satisfies the MDSUM conditions: Equation (3.5) holds for P_i, and the left and right hand sides in Equation (3.5) remain unchanged for the next I consecutive packet pairs ($I = 40$ suggested in [115]). CapProbe then uses P_i's PPD to compute the path capacity.

$$\min_i \left\{ d_{2i-1}^{(n)} + d_{2i}^{(n)} \right\} = \min_i \left\{ d_{2i-1}^{(n)} \right\} + \min_i \left\{ d_{2i}^{(n)} \right\}. \tag{3.5}$$

The drawback of the MDSUM method is that it considers only the packet pair whose packets are *both* unaffected by cross traffic. It therefore discards all other packet pairs, including those in which *only* a single packet has been interfered. As a result, useful information in those packet pairs is not fully utilized to speed up the measurement process. According to the scenario in Figure 3.3, the MDSUM method anticipates $\{p_5, p_6\}$ and filters out $\{p_1, p_2\}$ and $\{p_3, p_4\}$ because p_1's and p_4's packet delays are not minimum. In this case, however, the unaffected packets p_2 and p_3 are also discarded. This observation leads us to propose a new filtering technique that is based on the minimum delay of a single packet (instead of a packet pair), to be discussed next.

3.2.2 Minimal possible packet delay

Our new filtering technique is based on the notion of *minimal possible packet delay* (minDelay) defined as:

Definition 1. *A minimal possible packet delay is the delay experienced by a packet in a packet pair for which both the probe packet and the elicited response packet do not encounter any cross-traffic-induced queueing delay on the path, including*

1. *Type-H queueing delay: the queueing delay caused by the cross traffic present at the "head" of the queue upon the first packet's arrival, and*

2. *Type-I queueing delay: the queueing delay caused by the intervening cross traffic between the first and second packets in a packet pair.*

In the following we consider a packet pair $\{p_{j-1}, p_j\}$. It is not difficult to see that p_{j-1}'s minDelay can be obtained iff the probe packet and the elicited response packet are not queued (behind type-H cross traffic) at all hops of the path. Therefore,

Proposition 1. *(The first packet's minDelay) The necessary and sufficient conditions for $d_{j-1}^{(n)}$ being a minDelay are $w_{j-1}^{(h)} = 0,\ h = 1, \ldots, n$.*

However, obtaining the conditions for a second packet's minDelay is more involved. We first derive in Proposition 2 general expressions for $w_j^{(h)}$ and $\delta_{j-1,j}^{(h)}$ which take into account the two types of cross traffic. Figure 3.4 illustrates the two scenarios for which their PPDs are not the same.

Proposition 2. *At $H^{(h)}$, p_j's queueing delay is given by Equation (3.6), and $\{p_{j-1}, p_j\}$'s PPD is given by Equation (3.7).*

$$w_j^{(h)} = \left(w_{j-1}^{(h)} + X^{(h)} - \delta_{j-1,j}^{(h-1)}\right)^+ + q_{j-1,j}^{(h)}, \tag{3.6}$$

where $(x)^+ = \max\{0, x\}$*, and* $q_{j-1,j}^{(h)}$ *is* p_j*'s type-I queueing delay at* $H^{(h)}$*.*

$$\delta_{j-1,j}^{(h)} = \begin{cases} X^{(h)} + q_{j-1,j}^{(h)}, & \textit{if } w_{j-1}^{(h)} + X^{(h)} \geq \delta_{j-1,j}^{(h-1)}, \\ \delta_{j-1,j}^{(h-1)} - w_{j-1}^{(h)} + q_{j-1,j}^{(h)}, & \textit{otherwise.} \end{cases} \tag{3.7}$$

Proof. It is straightforward to obtain $w_j^{(h)}$ and $\delta_{j-1,j}^{(h)}$ directly from Figures 3.4(a)-3.4(b). Alternatively, $w_j^{(h)}$ can be derived from the Lindley's recurrence equation [122]. □

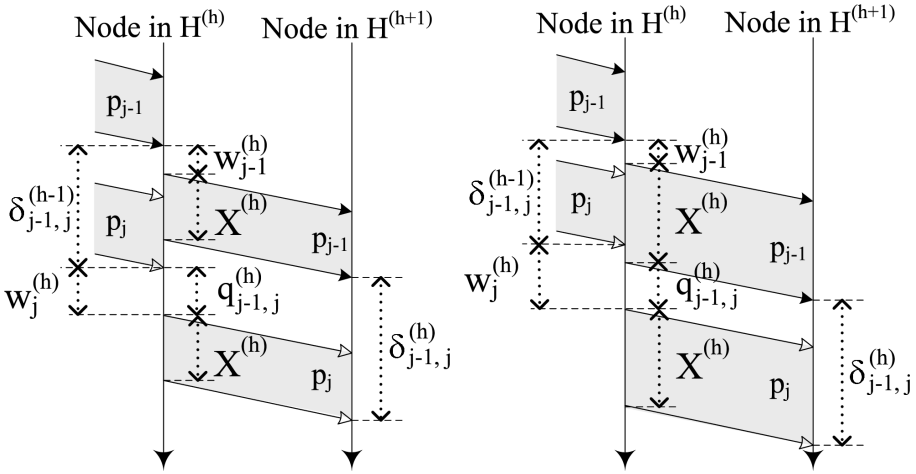


Figure 3.4: Two scenarios for deriving the queueing delay of p_j and PPD of $\{p_{j-1}, p_j\}$ at $H^{(h)}$.

We next consider the following three lemmas which will be used to prove the main results for p_j's minDelay in Proposition 3. Lemma 1 addresses the effect of type-I cross traffic on p_j's delay, whereas Lemmas 2-3 address that of type-H cross traffic. We let $w_{j,H}^{(h)}$ be the type-H queueing delay experienced by p_j at $H^{(h)}$.

Lemma 1. *For $d_j^{(n)}$ to be free from type-I queueing delay, $q_{j-1,j}^{(h)} = 0$ for $h = 1, \ldots, n$.*

Proof. It is clear from Equation (3.6) that $w_j^{(h)}$ does not include type-I queueing delay iff $q_{j-1,j}^{(h)} = 0$. Therefore, type-I cross traffic does not contribute to $d_j^{(n)}$ iff $q_{j-1,j}^{(h)} = 0, \ \forall h$. □

For the next two lemmas, Lemma 1 is assumed true, and we consider two types of hops: $H^{(h)}$ is a *local bottleneck hop* (LBH) if $X^{(h)} \geq \delta_{j-1,j}^{(h-1)}$ and a non-LBH, otherwise. In the absence of type-I cross traffic, p_{j-1} and p_j will be sent back to back on an LBH, but the two packets may be sent with a time gap on a non-LBH.

Lemma 2. *Considering that Lemma 1 holds and $H^{(h)}$ is a non-LBH, p_j does not experience type-H queueing delay at $H^{(h)}$ iff $w_{j-1}^{(h)} \leq \delta_{j-1,j}^{(h-1)} - X^{(h)}$.*

Proof. Note that for a non-LBH,

$$w_{j,H}^{(h)} = \left(w_{j-1}^{(h)} + X^{(h)} - \delta_{j-1,j}^{(h-1)} \right)^+. \tag{3.8}$$

Equation (3.8) shows that $w_{j-1}^{(h)} \leq \delta_{j-1,j}^{(h-1)} - X^{(h)}$ is the only condition for $w_{j,H}^{(h)} = 0$. □

Lemma 3. *Consider that Lemma 1 holds and $H^{(h)}$ is an LBH.*

(i) For $h = 1$, p_j does not experience type-H queueing delay at $H^{(1)}$ iff $w_{j-1}^{(1)} = 0$.

(ii) For $h > 1$, given that $H^{(h)}$ is preceded immediately by s $(0 \leq s \leq h-2)$ adjoining non-LBHs and then an LBH, p_j does not experience type-H queueing delay at $H^{(h)}$ iff $w_{j-1}^{(h-k)} = 0$ for $k = 0, \ldots, s$.

Proof. For case (i), note that $H^{(1)}$ is an LBH, because $X^{(1)} > \delta^{(0)}_{j-1,j} \equiv \tau^{(1)}_{j-1,j} = 0$. From Equation (3.6), $w^{(1)}_j = w^{(1)}_{j-1} + X^{(1)}$; therefore, it is required that $w^{(1)}_{j,H} \equiv w^{(1)}_{j-1} = 0$.

For case (ii), we first consider the case of $s = 0$ (i.e., $H^{(h-1)}$ is an LBH). From Equation (3.7), $\delta^{(h-1)}_{j-1,j} = X^{(h-1)}$, and from Equation (3.6),

$$\begin{aligned} w^{(h)}_j &= w^{(h)}_{j-1} + X^{(h)} - \delta^{(h-1)}_{j-1,j}, \\ &= w^{(h)}_{j-1} + X^{(h)} - X^{(h-1)} \geq 0. \end{aligned}$$

Therefore, it is required that $w^{(h)}_{j,H} \equiv w^{(h)}_{j-1} = 0$.

For case (ii) with $s > 0$, note that $h > 2$. Since $H^{(h-1)}$ to $H^{(h-s)}$ are non-LBHs and the second packet's delay at these hops satisfy Lemma 2, from Equation (3.7),

$$\delta^{(h-k)}_{j-1,j} = \delta^{(h-k-1)}_{j-1,j} - w^{(h-k)}_{j-1} \; for \; k = 1, \ldots, s. \tag{3.9}$$

By repeatedly substituting Equation (3.9) into $w^{(h)}_j$,

$$\begin{aligned} w^{(h)}_j &= w^{(h)}_{j-1} + X^{(h)} - \delta^{(h-1)}_{j-1,j}, \\ &= \sum_{k=0}^{s} w^{(h-k)}_{j-1} + X^{(h)} - X^{(h-s-1)} \geq 0. \end{aligned} \tag{3.10}$$

From Equation (3.10), $w^{(h)}_{j,H} \equiv \sum_{k=0}^{s} w^{(h-k)}_{j-1}$. Therefore, $w^{(h-k)}_{j-1} = 0$ for $k = 0, \ldots, s$ yields $w^{(h)}_{j,H} = 0$. □

Proposition 3. *(The second packet's minDelay) The necessary and suffi-*

cient conditions for $d_j^{(n)}$ being a minDelay are:

$$
\begin{aligned}
&(i) \quad q_{j-1,j}^{(h)} = 0,\ h = 1, \ldots, n,\ and \\
&(ii) \quad w_{j-1}^{(h)} = 0,\ h = 1, \ldots, h_b,\ and \\
&(iii) \quad w_{j-1}^{(h)} \leq \delta_{j-1,j}^{(h-1)} - X^{(h)},\ h = h_b + 1, \ldots, n.
\end{aligned}
$$

When $h_b = n$, condition (iii) is not needed.

Proof. Condition (i) is required because of Lemma 1.

For conditions (ii) and (iii), we first consider $H^{(h_b)}$. With $q_{j-1,j}^{(h)} = 0,\ \forall h$, it is not difficult to see that $X^{(h_b)} \geq \delta_{j-1,j}^{(h_b-1)}$, because $\delta_{j-1,j}^{(h_b-1)} \leq \max_{\forall h}\{X^{(h)}\}$. Therefore, $H^{(h_b)}$ is an LBH. According to Lemma 3, if all the hops between $H^{(h_b)}$, where $h_b > 1$, and $H^{(1)}$ are non-LBHs, condition (ii) must hold. Even if there are one or more LBHs between them, condition (ii) still holds, because the same argument can be applied to each segment of adjoining non-LBHs.

For condition (iii), all the hops after $H^{(h_b)}$, if any, must be non-LBHs for $d_j^{(n)}$ being a minDelay. To see why, assume that $H^{(h_a)}$, where $h_b < h_a \leq n$, is an LBH. Moreover, by setting $h_a - h_b = s + 1$ ($s \geq 0$), we could apply Lemma 3 and Equation (3.10) to $w_j^{(h_a)}$:

$$
\begin{aligned}
w_j^{(h_a)} &= \sum_{k=0}^{s} w_{j-1}^{(h_a-k)} + X^{(h_a)} - X^{(h_b)}, \\
&= X^{(h_a)} - X^{(h_b)}.
\end{aligned}
$$

Since $X^{(h_a)} < X^{(h_b)}$, $w_j^{(h_a)} < 0$ which contradicts that $w_j^{(h_a)} \geq 0$ for $H^{(h_a)}$ being an LBH. Therefore, $H^{(h_a)}$ must be a non-LBH. By applying Lemma 2

to each hop after $H^{(h_b)}$, we obtain condition (iii). □

3.2.3 Minimum delay difference

We propose a new cross-traffic filtering method called *minimum delay difference* (MDDIF) which exploits the minDelay of the packet pairs for capacity estimation. Proposition 4 shows that the unbiased PPD in Equation (3.3) can be obtained by the difference between a second packet's minDelay and a first packet's minDelay, and the two packets do not belong to the same packet pair.

Proposition 4. *A sequence of packet pairs* $\{p_{2i-1}, p_{2i}\}$, $i = 1, 2, \ldots$, *is dispatched by a local endpoint to measure the capacity of an* n*-hop path* ($n \geq 2$). *Moreover,* $d_{2k-1}^{(n)}$ *(for the first packet in* P_k*) and* $d_{2l}^{(n)}$ *(for the second packet in* P_l*) are minDelays, where* $l \neq k$. *Then,*

$$d_{2l}^{(n)} - d_{2k-1}^{(n)} = \max\left\{\frac{S_f}{C_f^{(n)}}, \frac{S_r}{C_r^{(n)}}\right\}. \tag{3.11}$$

Proof. First of all, from Equation (3.1),

$$d_{2l}^{(n)} - d_{2k-1}^{(n)} = d_{2l}^{(n-1)} - d_{2k-1}^{(n-1)} + w_{2l}^{(n)} - w_{2k-1}^{(n)}. \tag{3.12}$$

Using $w_{2k-1}^{(n)} = 0$ (from Proposition 1), Equation (3.6) for $w_{2l}^{(n)}$, and $q_{2l-1,2l}^{(n)} = 0$ (from Proposition 3(i)), Equation (3.12) becomes

$$d_{2l}^{(n)} - d_{2k-1}^{(n)} \;=\; d_{2l}^{(n-1)} - d_{2k-1}^{(n-1)} + \left(w_{2l-1}^{(n)} + X^{(n)} - \delta_{2l-1,2l}^{(n-1)}\right)^{+}. \tag{3.13}$$

We now use mathematical induction on n for the proof.

The base case: $n = 2$ (i.e., $h_f = 1$ and $h_r = 2$) By applying Equation (3.13) recursively for $n = 2$, we obtain

$$d_{2l}^{(2)} - d_{2k-1}^{(2)} = \sum_{h=1}^{2} \left(w_{2l-1}^{(h)} + X^{(h)} - \delta_{2l-1,2l}^{(h-1)} \right)^{+}. \tag{3.14}$$

Note that $\delta_{2l-1,2l}^{(0)} \equiv \tau_{2l-1,2l}^{(1)} = 0$. Moreover, $H^{(h_b)}$ is either $H^{(1)}$ (the forward-path hop) or $H^{(2)}$ (the reverse-path hop):

Case 1 ($h_b = h_f = 1$): Since $d_{2l}^{(2)}$ is a minDelay, $w_{2l-1}^{(1)} = 0$ (from Proposition 3(ii)) and $\left(w_{2l-1}^{(2)} + X^{(2)} - \delta_{2l-1,2l}^{(1)} \right)^{+} = 0$ (from Proposition 3(iii)). Equation (3.14) therefore becomes

$$d_{2l}^{(2)} - d_{2k-1}^{(2)} = X^{(1)} = S_f / C^{(1)},$$

which is the same as Equation (3.11) for $n = 2$.

Case 2 ($h_b = h_r = 2$): Since $d_{2l}^{(2)}$ is a minDelay, $w_{2l-1}^{(1)} = w_{2l-1}^{(2)} = 0$ (from Proposition 3(ii)) and $\delta_{2l-1,2l}^{(1)} = X^{(1)}$. Equation (3.14) therefore becomes

$$d_{2l}^{(2)} - d_{2k-1}^{(2)} = X^{(2)} = S_r / C^{(2)},$$

which is the same as Equation (3.11) for $n = 2$.

The inductive step: Assuming that Equation (3.11) holds for $n \geq 2$, we prove that Equation (3.11) also holds for $n + 1$. By substituting Equation (3.11) (the inductive hypothesis for n) into Equation (3.13) for $n + 1$, we

have

$$d_{2l}^{(n+1)} - d_{2k-1}^{(n+1)} = \left(w_{2l-1}^{(n+1)} + X^{(n+1)} - \delta_{2l-1,2l}^{(n)} \right)^{+} + \max \left\{ \frac{S_f}{C_f^{(n)}}, \frac{S_r}{C_r^{(n)}} \right\}. \quad (3.15)$$

There are two cases to consider: h_b remains the same, and $h_b = n+1$. Note that $H^{(n+1)}$ introduces a new link to the reverse path.

Case 1 ($h_b < n+1$): Since $d_{2l}^{(n+1)}$ is a minDelay, applying $w_{2l-1}^{(n+1)} \leq \delta_{2l-1,2l}^{(n)} - X^{(n+1)}$ (from Proposition 3(iii)) to Equation (3.15) yields

$$\begin{aligned} d_{2l}^{(n+1)} - d_{2k-1}^{(n+1)} &= \max \left\{ \frac{S_f}{C_f^{(n)}}, \frac{S_r}{C_r^{(n)}} \right\}, \\ &= \max \left\{ \frac{S_f}{C_f^{(n+1)}}, \frac{S_r}{C_r^{(n+1)}} \right\}. \end{aligned} \quad (3.16)$$

Case 2 ($h_b = n+1$): Since $d_{2l}^{(n+1)}$ is a minDelay, we have $w_{2l-1}^{(h)} = 0,\ \forall 1 \leq h \leq n+1$ (from Proposition 3(ii)). Accordingly, both $d_{2l}^{(n)}$ and $d_{2l-1}^{(n)}$ are also minDelays; therefore, $d_{2l}^{(n)} - d_{2l-1}^{(n)} = \max \left\{ S_f/C_f^{(n)}, S_r/C_r^{(n)} \right\}$ (the inductive hypothesis). Substituting $w_{2l-1}^{(n+1)} = 0$ and $\delta_{2l-1,2l}^{(n)} = d_{2l}^{(n)} - d_{2l-1}^{(n)}$ (from Equation (3.2)) into Equation (3.15) yields

$$d_{2l}^{(n+1)} - d_{2k-1}^{(n+1)} = X^{(n+1)} = \frac{S_r}{C^{(n+1)}},$$

which is the same as Equation (3.16). □

3.3 A first-passage-time analysis

In this section, we analyze and compare the MDDIF and MDSUM methods based on stochastic models of their *first passage time* (FPT). The MDDIF method's FPT is defined as the first time (in terms of the number of packet pairs sent) to obtain the two minDelays. On the other hand, the MDSUM method's FPT is defined as the first time to obtain the minimum delay sum (which is equal to the sum of the two minDelays). Therefore, a smaller FPT results in a faster measurement. Kapoor *et al.* [115] analyzed the FPT for the MDSUM method under Poisson, deterministic and Pareto cross traffic for a single-queue model. They used a joint probability for both the first and second packets of a packet pair not being queued by cross traffic. Besides deriving the FPT for the MDDIF method, we also derive the FPT for the MDSUM method. Moreover, we consider a multi-hop model, instead of a single-hop model considered in [115].

3.3.1 The first passage times of the MDDIF method and MDSUM method

Let $X_i, i \geq 1$, be a sequence of independent and identically distributed (i.i.d.) Bernoulli random variables with parameter p_X (probability for $X_i = 1$) for the minDelay event of p_{2i-1} (the first packet in P_i). $X_i = 1$ if $d_{2i-1}^{(n)}$ is a minDelay and $X_i = 0$, otherwise. Similarly, $Y_i, i \geq 1$, is a sequence of i.i.d. Bernoulli random variables with parameter p_Y for the minDelay event of p_{2i} (the second packet in P_i). $Y_i = 1$ if $d_{2i}^{(n)}$ is a minDe-

lay and $Y_i = 0$, otherwise. Moreover, the sequence of the joint random variables (X_i, Y_i) are i.i.d. with a joint probability density function (pdf) $p_{XY}(x, y)$. Note that X_i and Y_i are generally not independent.

The MDDIF method's FPT is given by

$$T_{DIF} = \inf \{i : SX_i > 0 \, and \, SY_i > 0\}, \tag{3.17}$$

where $SX_i = \sum_{k=1}^{i} X_k$ and $SY_i = \sum_{k=1}^{i} Y_k$. To obtain the pdf for T_{DIF}, we consider another sequence of random variables $Z_i, i \geq 1$, for which

$$Z_i = \begin{cases} 0, & \text{if } SX_i = 0 \text{ and } SY_i = 0, \\ 1, & \text{if } SX_i = 0 \text{ and } SY_i > 0, \\ 2, & \text{if } SX_i > 0 \text{ and } SY_i = 0, \\ 3, & \text{if } SX_i > 0 \text{ and } SY_i > 0, \end{cases}$$

is a time-homogeneous Markov chain with finite states. Figure 3.5 shows the state transitions diagram for Z_i. As shown, the Markov chain is an *absorbing Markov chain* with an absorbing state 3. The other three states are transient states with a finite number of visits [120]. In other words, the MDDIF method has a finite FPT.

Denote the stationary transition probabilities by $p_{mn} = P[Z_{i+1} = n | Z_i = m]$, $m, n = 0, 1, 2, 3$. The transition probability matrix of the Markov chain

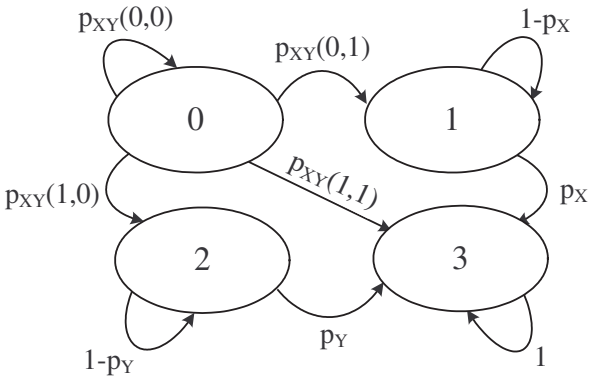


Figure 3.5: State transition diagram for the time-homogenous Markov chain $Z_i, i \geq 1$ with three transient states 0, 1, and 2, and an absorbing state 3.

is given by

$$\begin{aligned}
\mathbf{P} &= [p_{mn}], \\
&= \left[\begin{array}{c|c} \mathbf{Q} & \mathbf{A} \\ \hline 0 & 1 \end{array}\right], \\
&= \left[\begin{array}{ccc|c} p_{XY}(0,0) & p_{XY}(0,1) & p_{XY}(1,0) & p_{XY}(1,1) \\ 0 & 1-p_X & 0 & p_X \\ 0 & 0 & 1-p_Y & p_Y \\ \hline 0 & 0 & 0 & 1 \end{array}\right],
\end{aligned}$$

where **Q** is for the transitions among the three transient states, whereas **A** is for the transitions from the transient states to the absorbing state. Since the MDDIF method starts from state 0, the initial probability vector for the first three (transient) states is given by $\pi_0 = [1\ 0\ 0]$. From [159],

$$P[T_{DIF} = i] = \pi_0 \mathbf{Q}^{i-1} \mathbf{A}. \tag{3.18}$$

To determine the expectation of the FPT, we obtain $\mathbf{t} = [t_0, t_1, t_2]^T$, for which $t_k,\ k = 0, 1, 2$, is the expected number of steps taken prior to reaching the absorbing state, given that the chain begins from state k. From [120], $\mathbf{t} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{c}$, where $\mathbf{I}$ is an identity matrix, $\mathbf{c} = [1\ 1\ 1]^T$, and $(\mathbf{I} - \mathbf{Q})^{-1}$ is the fundamental matrix of the Markov chain. We therefore have

$$\begin{aligned} E[T_{DIF}] &= \pi_0 \mathbf{t}, \\ &= \frac{1}{1 - p_{XY}(0,0)} \left(1 + \frac{p_{XY}(0,1)}{p_X} + \frac{p_{XY}(1,0)}{p_Y}\right). \end{aligned} \tag{3.19}$$

The derivation of $E[T_{DIF}]$ is given in Appendix A.1.

On the other hand, the MDSUM method's FPT is defined as

$$T_{SUM} = \inf\{i : X_i = 1 \; and \; Y_i = 1\}. \tag{3.20}$$

Therefore, T_{SUM} is a geometrically distributed random variable with parameter $p_{XY}(1,1)$.

Besides showing that $E[T_{DIF}] < E[T_{SUM}]$, Proposition 5 states the main idea of the MDDIF method. The necessary and sufficient condition for the MDDIF method to obtain capacity estimates faster than the MDSUM method is when it is possible to find the two minDelays from different packet pairs (i.e., $p_{XY}(0,1) > 0$ and $p_{XY}(1,0) > 0$).

Proposition 5. $E[T_{DIF}] < E[T_{SUM}]$ *iff* $p_{XY}(0,1) > 0$ *and* $p_{XY}(1,0) > 0$.

Proof. By using Equation (3.19) and $E[T_{SUM}] = 1/p_{XY}(1,1)$, we compute

the relative gain of $E[T_{DIF}]$ as

$$\Psi = \frac{E[T_{SUM}] - E[T_{DIF}]}{E[T_{SUM}]} = \frac{\sigma}{\sigma + \xi}, \tag{3.21}$$

where

$$\sigma = p_{XY}(0,1)p_{XY}(1,0)(p_X + p_Y), \tag{3.22}$$

$$\xi = p_{XY}(1,1)[p_{XY}(0,1)(p_X + p_Y) + p_X^2]. \tag{3.23}$$

We leave the derivation of Ψ in Appendix A.2.

Assume that $p_{XY}(0,1) > 0$ and $p_{XY}(1,0) > 0$. Since $p_X = p_{XY}(1,0) + p_{XY}(1,1)$, $p_{XY}(1,0) > 0$ implies $p_X > 0$. Similarly, $p_{XY}(0,1) > 0$ implies $p_Y > 0$. From Equation (3.22), $\sigma > 0$. Moreover, $0 \leq p_{XY}(1,1) < 1$ due to the law of total probability, and it is easy to see that $0 < [p_{XY}(0,1)(p_X + p_Y) + p_X^2] < 1$. Therefore, $0 \leq \xi < 1$ from Equation (3.23), and as a result, $0 < \Psi \leq 1$.

In the other direction, assume that $\Psi > 0$. From Equations (3.21)-(3.22), $\sigma > 0$ which is equivalent to $p_{XY}(0,1) > 0$ and $p_{XY}(1,0) > 0$. □

Proposition 6 shows that the MDDIF method does not have the speed advantage for $h_b = n$. Since $H^{(n)}$ is a reverse-path hop, the MDDIF and MDSUM methods give the same expected FPTs for measuring $C_r^{(n)}$. Nevertheless, the MDDIF method's speed advantage may still be retained for measuring $C_f^{(n)}$ if S_f and S_r are selected such that $h_b = h_f$. This could be done for $C_r^{(n)} \geq C_f^{(n)}$ (e.g., by choosing $S_f = S_r$), and for $C_r^{(n)} < C_f^{(n)}$ if it is feasible to achieve $S_f/S_r > C_f^{(n)}/C_r^{(n)} > 1$ (see the discussion for

Equations (3.3) and (3.4)).

Proposition 6. $E[T_{DIF}] = E[T_{SUM}]$ *for* $h_b = n$.

Proof. The event of $X_i = 0$ and $Y_i = 1$ is not possible (i.e., $p_{XY}(0,1) = 0$) for this scenario. Since p_{2i-1} is not a minDelay, $w_{2i-1}^{(h)} \neq 0$ for some h (from Proposition 1). Thus, it is not possible for p_{2i} being a minDelay, because $w_{2i-1}^{(h)} = 0$ is required for all h (according to Proposition 3). As a result, $\sigma = 0$ in Equation (3.22); thus, $\Psi = 0$. □

3.3.2 A first-passage-time analysis for $h_b = n - 1$

To quantify the MDDIF method's speed advantage for $h_b \neq n$, we analyze the case of $h_b = n - 1$ here. We model the node in $H^{(h)}$ as a single FIFO queue with unlimited buffer. The packet inter-arrival times for the cross traffic to the queue at $H^{(h)}$ (denoted by $A^{(h)}$) are exponentially distributed with rate $\lambda^{(h)}$. This assumption is based on the previous study that the packet inter-arrival time distribution is reasonably represented by the Poisson process on sub-second timescales [116]. The inter-arrival process for the packet pairs to the queue is also exponential; therefore, they take a random look at the state of the queue. Since the packet pairs do not generate a significant load to $H^{(h)}$, the average packet arrival rate $\lambda^{(h)}$ is retained. The packet service time at $H^{(h)}$ (denoted by $B^{(h)}$) is a random variable which depends on the packet size (denoted by $S^{(h)}$) distribution and $C^{(h)}$. To make the analysis simple, we assume that $B^{(h)}$ is an exponential random variable with $\mu^{(h)} = 1/E[B^{(h)}] = C^{(h)}/E[S^{(h)}]$ being the packet service rate at $H^{(h)}$. As a result, each node is modeled as a

classic M/M/1 queue.

3.3.2.1 Computing the probabilities

In this section we derive analytical expressions for the probabilities in Equation (3.19) for the MDDIF method and $p_{XY}(1,1)$ for the MDSUM method. We also note that it is sufficient to obtain expressions for $p_{XY}(1,1)$, p_X, and p_Y, because they can be used to obtain other probabilities: $p_{XY}(0,0) = 1-(p_X+p_Y-p_{XY}(1,1))$, $p_{XY}(0,1) = p_Y - p_{XY}(1,1)$, and $p_{XY}(1,0) = p_X - p_{XY}(1,1)$.

1. Computing p_X We again consider $\{p_{j-1}, p_j\}$. Since p_j will not affect p_{j-1}, p_X is the probability that all nodes on the path are empty upon p_{j-1}'s arrival. The empty probability is given by $1-\rho^{(h)}$ for $H^{(h)}$ [159], where $\rho^{(h)} = \lambda^{(h)}/\mu^{(h)}$. By applying an independence assumption for the nodes,

$$p_X = \prod_{h=1}^{n} \left(1-\rho^{(h)}\right). \tag{3.24}$$

2. Computing $p_{XY}(1,1)$ Same as the last case, p_{j-1} arrives at an empty node in $H^{(h)}$ with probability $1-\rho^{(h)}$. Given that a period of t has been passed since the last cross-traffic packet arrival upon p_{j-1}'s arrival at $H^{(h)}$, the probability that p_j will not be delayed by the intervening cross traffic[1] between p_{j-1} and p_j is given by $P[A^{(h)} > t + \delta_{j-1,j}^{(h-1)} | A^{(h)} > t]$. By the memoryless property of an exponential distribution, this conditional probabil-

[1]The presence of intervening cross traffic may not induce Type-I queueing delay to p_j. Similar to [115], we do not consider this case to simplify the derviation.

ity is given by $P[A^{(h)} > \delta_{j-1,j}^{(h-1)}] = e^{-\lambda^{(h)}\delta_{j-1,j}^{(h-1)}}$. Hence,

$$p_{XY}(1,1) = \prod_{h=1}^{n} \left(1-\rho^{(h)}\right) e^{-\lambda^{(h)}\delta_{j-1,j}^{(h-1)}}. \tag{3.25}$$

3. Computing p_Y Since $h_b = n-1$, we consider two subpaths for the analysis: (I) $\{H^{(1)}, \ldots, H^{(n-1)}\}$ and (II) $\{H^{(n)}\}$. Let p'_Y be the probability that p_j's delay on subpath (I) is a minDelay and p''_Y the probability that p_j's delay on subpath (II) is a minDelay. Therefore, $p_Y = p'_Y p''_Y$.

For subpath (I), due to Proposition 3(i)-(ii), both p_{j-1} and p_j do not experience queueing delay on the subpath. Therefore, p'_Y is the same as Equation (3.25) except for the last hop.

The subpath (II) consists of $H^{(n)}$ which is after $H^{(h_b)}$. Therefore, according to Proposition 3(iii), p_{j-1} must not be delayed by more than $\omega = \delta_{j-1,j}^{(n-1)} - X^{(n)} > 0$. Let $W^{(n)}$ be the random variable for p_{j-1}'s queueing delay at $H^{(n)}$. Based on the Pollaczek-Khinchin equation for an $M/M/1$ queue [159],

$$p_{W^{(n)}}(t) = \left(1-\rho^{(n)}\right)\left(\delta_0(t) + \lambda^{(n)} e^{-\mu^{(n)}\left(1-\rho^{(n)}\right)t}\right), \tag{3.26}$$

where $\delta_0(t)$ is the Dirac delta function. Moreover, p_j does not encounter intervening cross traffic at $H^{(n)}$ (from Proposition 3(i)). The probability for this event, conditioned on the event that p_{j-1} has encountered a queueing delay of t, is given by the probability of the event $A^{(n)} >$

$\delta_{j-1,j}^{(n-1)} - t$. Therefore, we can obtain p''_Y and p_Y:

$$\begin{aligned} p''_Y &= \int_0^{\omega} P[W^{(n)} = t, A^{(n)} > \delta_{j-1,j}^{(n-1)} - t]dt, \\ &= \left(1-\rho^{(n)}\right) e^{-\lambda^{(n)}\delta_{j-1,j}^{(n-1)}} \left[1 + \frac{\rho^{(n)}}{1-2\rho^{(n)}}\left(1 - e^{-\mu^{(n)}(1-2\rho^{(n)})\omega}\right)\right], \end{aligned} \tag{3.27}$$

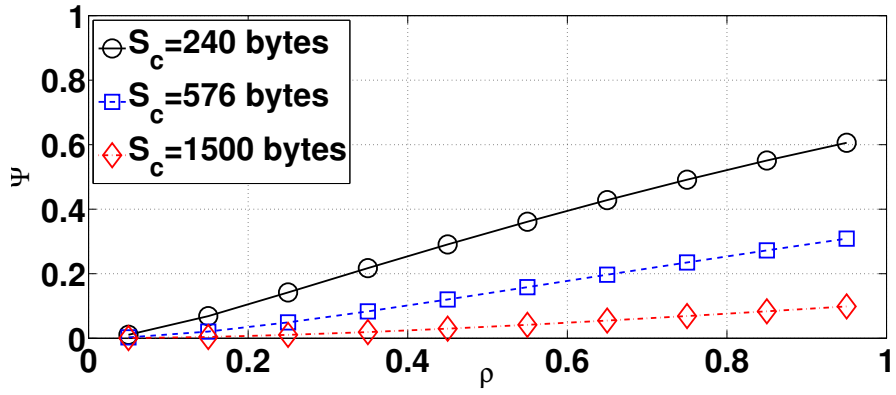
$$\begin{aligned} p_Y &= p'_Y p''_Y, \\ &= p_{XY}(1,1)\left[1 + \frac{\rho^{(n)}}{1-2\rho^{(n)}}\left(1 - e^{-\mu^{(n)}(1-2\rho^{(n)})\omega}\right)\right]. \end{aligned} \tag{3.28}$$

3.3.2.2 Analytical results

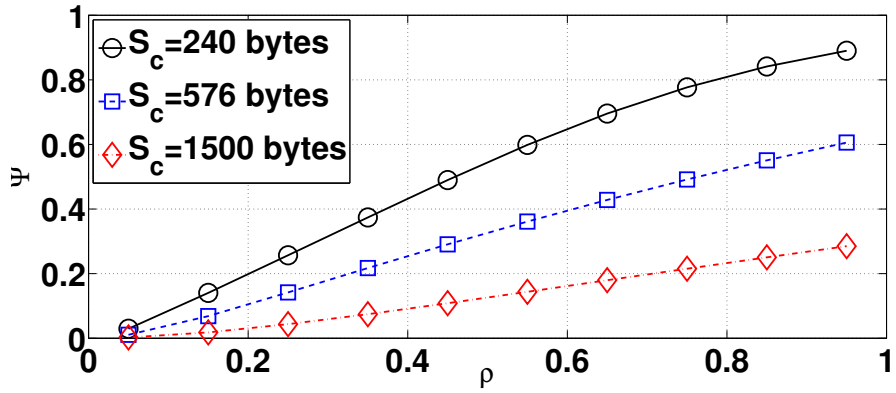
Using the analytical results from the last section for $h_b = n-1$, Figure 3.6 reports Ψ for $n = 5$ with link capacities of $\{100, 75, 55, 40, 80\}$ Mbits/s and $S_f = \{240, 576, 1500\}$ bytes. Each sub-figure plots Ψ against a mean utilization ρ ($\rho^{(h)} = \rho,\ \forall h$) with a given mean (cross-traffic) packet size S_c ($E[S^{(h)}] = S_c,\ \forall h$). The results are in agreement with Proposition 5.

Figure 3.6 shows that the benefit of the MDDIF method increases with S_f and ρ, but decreases with S_c. As ρ increases, p_X, $p_{XY}(1,1)$, p_Y, $p_{XY}(0,1)$ all decrease (Equations (3.24), (3.25), (3.28), and (3.29)). That is, it is harder for both MDDIF and MDSUM methods to find valid samples as the intensity of cross traffic increases. However, the impact on the MDSUM method is much more serious, because it is required to obtain the minDelay for both packets from the same packet pair.

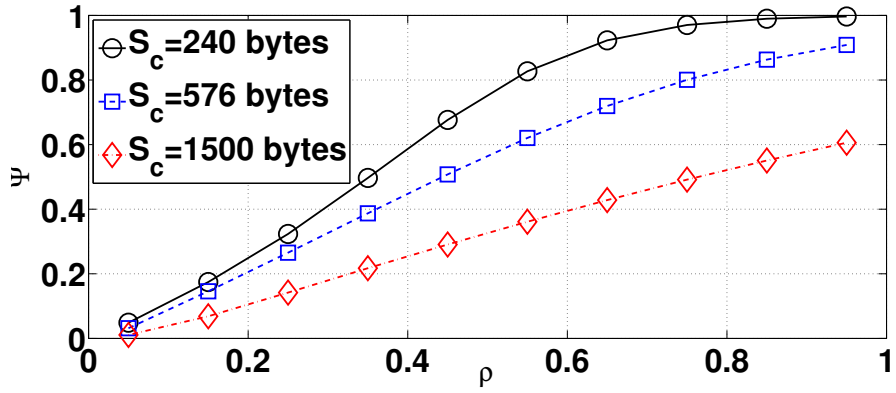
As for the impact of S_f and S_c, Figure 3.7(a) shows the distribution of $p_{XY}(1,1)$ with $S_c = [240, 1500]$ bytes, $S_f = [240, 1500]$ bytes, and $\rho = 20\%$. Notice that $p_{XY}(1,1)$ drops drastically as S_f increases and S_c decreases,



(a) $S_f = 240$ bytes.



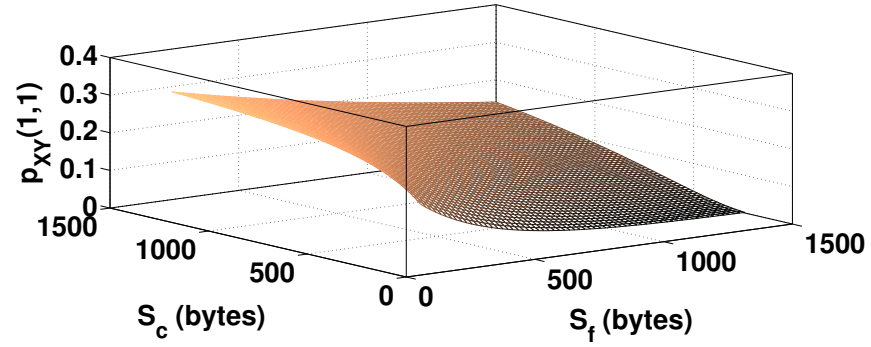
(b) $S_f = 576$ bytes.



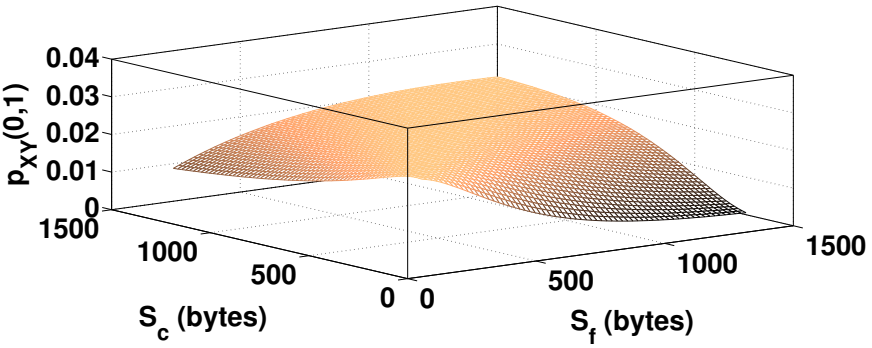
(c) $S_f = 1500$ bytes.

Figure 3.6: The relative gain of the expected first passage times for the MDDIF and MDSUM methods.

causing the MDSUM method a longer time to find a valid capacity sample. This is because the PPD ($\delta_{j-1,j}^{(h_b)}$) increases with S_f and the probability for the cross-traffic packets intervening between the two packets increases for a small S_c. Although S_f and S_c also affect the MDDIF method in a similar fashion, the impact is less severe, because it can obtain the two minDelays from different packet pairs.



(a) $p_{XY}(1,1)$.



(b) $p_{XY}(0,1)$.

Figure 3.7: The values of $p_{XY}(1,1)$ and $p_{XY}(0,1)$ for $\rho = 20\%$, and different probe and cross-traffic packet sizes.

There is also a subtle relationship between S_f and S_c concerning $p_{XY}(0,1)$,

which is illustrated in Figure 3.7(b). By inspecting Equation (3.28),

$$p_{XY}(0,1) = \frac{p_{XY}(1,1)\rho^{(n)}}{1-2\rho^{(n)}}\left(1-e^{-\mu^{(n)}(1-2\rho^{(n)})\omega}\right). \quad (3.29)$$

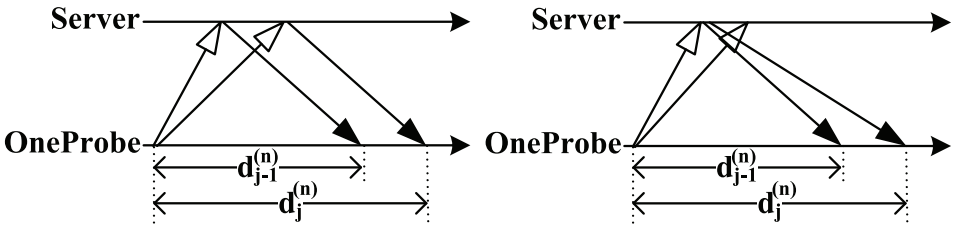
Clearly, the likelihood of fulfilling Proposition 3(iii) increases with S_f, because the probability of p_j's queueing due to p_{j-1}'s decreases. Increasing S_f, however, can decrease the probability of fulfilling Proposition 3(i), because the increased dispersion can accommodate more cross-traffic packet arrivals between the two packets [78, 115]. Therefore, Figure 3.7(b) shows that $p_{XY}(0,1)$ peaks near $S_f = S_c$ but drops for $S_c > S_f$ and $S_f > S_c$.

Besides the speed advantage, the MDDIF method is also simpler than the MDSUM method. According to Equation (3.5), the MDSUM method is required to keep track of the minimum delay sum and the minDelay for the first and second packets, and performs a validation test for each measurement. Clearly, the MDSUM also needs to store the PPD sample responsible for the minimum delay sum. The MDDIF method, on the other hand, only needs to store two minDelays.

3.4 Measurement results

We have incorporated the MDDIF method into HTTP/OneProbe to measure forward-path capacity and reverse-path capacity with a remote web server. Figure 3.8 shows the OneProbe's packet transmissions for measuring asymmetric capacity. To measure the forward-path capacity and round-trip capacity, OneProbe dispatches a pair of probe TCP data pack-

ets, each of which elicits a response TCP data packet from the server. To measure the reverse-path capacity, OneProbe dispatches a specially crafted probe packet (Section 2.4.2.6) to elicit two back-to-back response TCP data packets from the server (which deviates from assumption (3)). To compute the capacity estimates, the MDDIF method requires the RTT samples that measure the time between sending a probe packet and receiving the elicited response packet.



(a) Measuring $C_f^{(n)}$ and $C_b^{(n)}$. (b) Measuring $C_r^{(n)}$.

Figure 3.8: OneProbe's packet transmissions for forward-path, reverse-path, and round-trip capacity measurements.

For the purpose of the evaluation, we have also incorporated the MDSUM method into HTTP/OneProbe. Besides the RTT samples, the MDSUM method also requires the PPD samples that measure the inter-arrival time between a pair of response packets.

There are several important advantages of implementing the MDDIF and MDSUM methods in HTTP/OneProbe. First, HTTP/OneProbe facilitates the *real* data-path capacity measurement by using only TCP data packets as probe and response packets, whereas many existing tools [47, 70, 79, 104, 127, 144, 163, 197] for the non-cooperative capacity measurement do not. Second, it has been shown that HTTP/OneProbe can avoid the processing latency at the remote web server [137]. This is, however,

not true for other HTTP-based RTT measurement tools, such as, httping [98]. Moreover, HTTP/OneProbe can unambiguously detect packet loss and reordering events for each probe packet and each response packet. This capability ensures that all the RTTs come from lossless and order-preserved probe and response packets.

In the following, we present three sets of capacity measurement results using the HTTP/OneProbe implementation. For the first set, we used a controlled testbed environment to evaluate the impact of cross traffic on the measurement accuracy and the FPTs for both the MDDIF and MDSUM methods. For the second and third sets, we used ADSL links (real and emulated) as the bottleneck links.

3.4.1 Testbed evaluation of the MDDIF method and the MDSUM method

The testbed, shown in Figure 3.9, was configured with a 16-hop round-trip path ($n = 16$), consisting of a probe sender, a web server running Apache v2.2.3 as the remote node, four cross-traffic clients $\mathbb{X}_1 - \mathbb{X}_4$, and seven forwarding devices—three Linux 2.6.26 routers $\mathbb{R}_1 - \mathbb{R}_3$ and four store-and-forward Ethernet switches $\mathbb{S}_1 - \mathbb{S}_4$. $\mathbb{S}_1$ is a Gigabit switch, $\mathbb{S}_4$ is a 10 Mbits/s switch, and the others are 100 Mbits/s switches. Since we used $S_f = S_r$, $h_b = 10$ and $C_b^{(n)} = 10$ Mbits/s. We ran TC/Netem [97] in each router to emulate a fixed RTT of 300 milliseconds between the probe sender and web server. We found that the delay emulated by TC/Netem in each router was stable and similar to the results reported in [162].

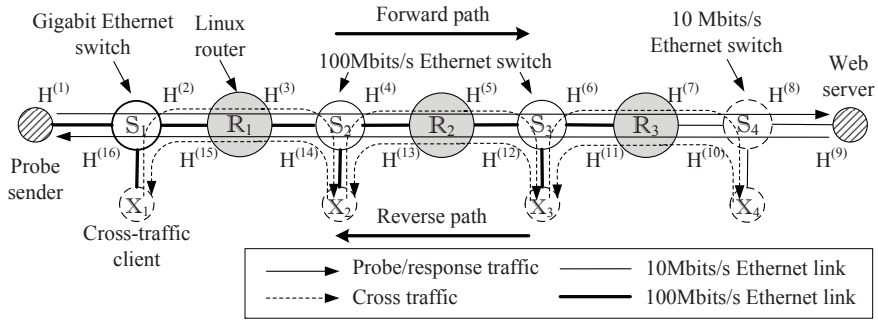


Figure 3.9: The testbed topology for the evaluation of the MDDIF and MDSUM methods.

Each cross-traffic client generated forward-path (reverse-path) cross traffic to another cross-traffic client to the right (left) to emulate a loading rate of ρ on the corresponding path segment. Similar to [78], the cross-traffic packets had uniformly distributed sizes in the $[40, 1500]$ bytes range and Pareto inter-arrivals with a shape parameter $\alpha = 1.9$. We ran HTTP/OneProbe from the probe sender to dispatch a sequence of packet pairs according to a Poisson distribution with a mean rate of 2Hz. The probe sender was equipped with a DAG 4.5 passive network monitoring card [4] to obtain the PPD and RTT samples in microsecond resolution which was limited by the `pcap` header structure [18].

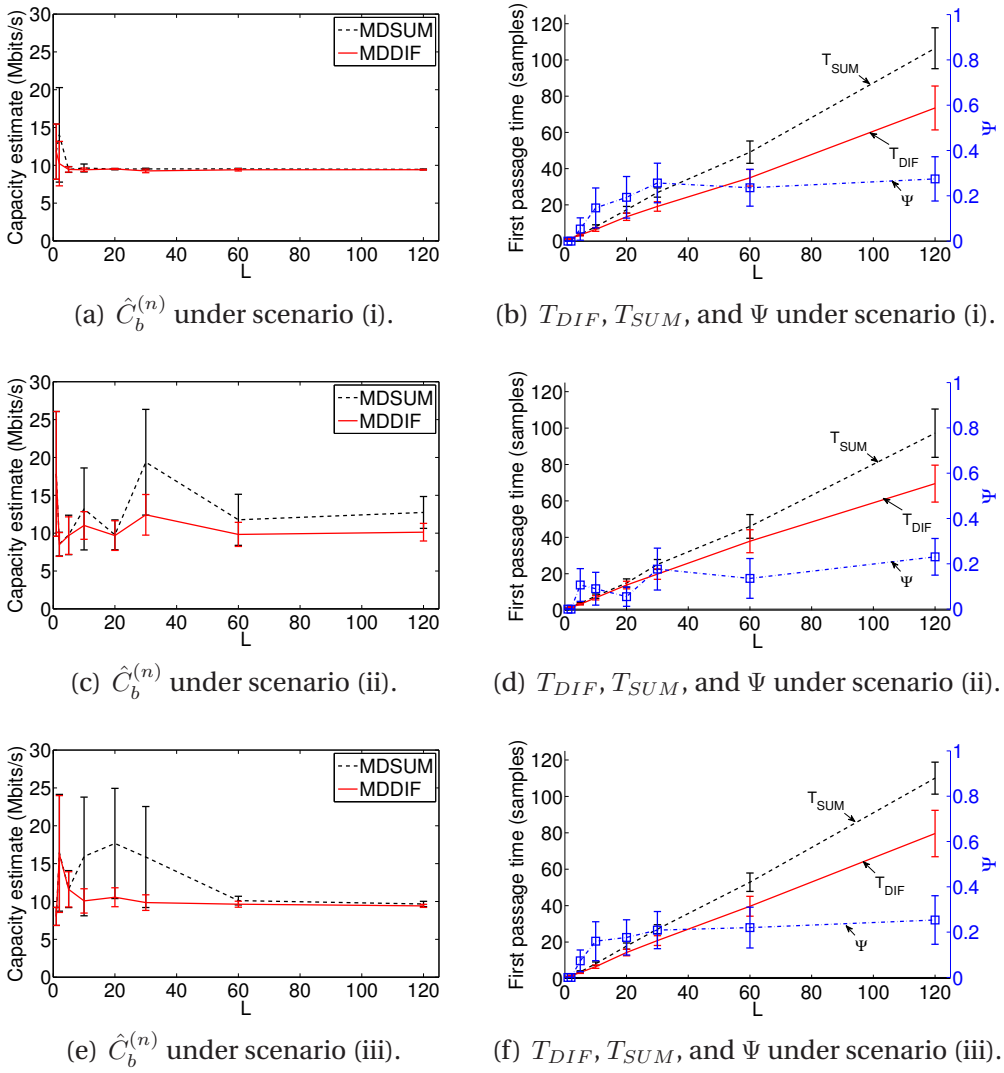
We conducted three sets of experiments with three cross-traffic loading scenarios using the MDDIF and MDSUM methods with $S_f = S_r = 240$ bytes, and the results were plotted in Figures 3.10(a)-3.10(f). For each set, we conducted experiments with different numbers of packet-pair samples L, ranging from 1 to 120. Moreover, for each L value, we repeated the experiments 30 times to obtain the mean and confidence intervals for $\hat{C}_b^{(n)}$ (an estimate of $C_b^{(n)}$), FPTs (T_{DIF} and T_{SUM}, in number of sam-

ples), and Ψ. When T_{SUM} was undefined after sending L packet pairs, we let $T_{SUM} = L$ and computed $\hat{C}_b^{(n)}$ using the PPD of the packet pair with the smallest RTT sum.

For low cross-traffic loads, Figures 3.10(a)-3.10(b) show that the capacity estimates obtained by the two methods coincide for $L \geq 20$, and both were very accurate (MDDIF: 9.42 Mbits/s, MDSUM: 9.46 Mbits/s). Moreover, the figure shows that the MDDIF method has a clear speed advantage with $\Psi \approx 27\%$.

For higher cross-traffic load, Figures 3.10(c)-3.10(d) show that the measurement accuracy deteriorated for both methods. For $L = 120$, both methods overestimated $C_b^{(n)}$ and produced higher variations in their estimates. However, the impact of the cross traffic on the MDSUM method was more significant. The MDSUM method obtained 12.73 Mbits/s (27.3% error), whereas the MDDIF method 10.12 Mbits/s (1.2% error). Moreover, the MDDIF method enjoyed a relative gain of about 23% on the measurement speed.

In the third set of experiments, we emulated a typical high-load downlink condition (e.g., downloading from a server) by deploying asymmetric cross-traffic loads of $\rho = 0.5$ for $\mathbb{X}_3 \rightarrow \mathbb{X}_2$ and $\mathbb{X}_2 \rightarrow \mathbb{X}_1$, and $\rho = 0.1$ for others. Figures 3.10(e)-3.10(f) show that both methods were sufficiently accurate for $L = 120$: the MDDIF method obtained 9.42 Mbits/s (5.8% error) and the MDSUM method 9.67 Mbits/s (3.3% error). Although the MDSUM method is slightly more accurate, its capacity estimates saw a higher variation when L is not large enough. Similar to the last two cases, the MDDIF method had a clear speed advantage of about 25%.



(a) $\hat{C}_b^{(n)}$ under scenario (i).

(b) T_{DIF}, T_{SUM}, and Ψ under scenario (i).

(c) $\hat{C}_b^{(n)}$ under scenario (ii).

(d) T_{DIF}, T_{SUM}, and Ψ under scenario (ii).

(e) $\hat{C}_b^{(n)}$ under scenario (iii).

(f) T_{DIF}, T_{SUM}, and Ψ under scenario (iii).

Figure 3.10: Round-trip capacity estimates and FPTs for the MDDIF and MDSUM methods and corresponding relative gains with $S_f = S_r = 240$ bytes under three cross-traffic scenarios: (i) $\rho = 0.1$ for all path segments, (ii) $\rho = 0.5$ for all path segments, and (iii) $\rho = 0.5$ for $\mathbb{X}_3 \to \mathbb{X}_2$ and $\mathbb{X}_2 \to \mathbb{X}_1$, and $\rho = 0.1$ for the rest.

3.4.2 Measuring remote ADSL links

We deployed HTTP/OneProbe to conduct both forward-path and reverse-path capacity measurement from a local measuring node connected to

an 1 Gbit/s Ethernet link. A sequence of probes with a fixed sampling interval of 500 milliseconds was dispatched to a remote ADSL endpoint with a downlink speed of 8 Mbits/s and an uplink speed of 800 Kbits/s. Therefore, the forward path (from the measuring node to the ADSL node) contained the ADSL's downlink, whereas the reverse path (from the ADSL node to the measuring node) contained the ADSL's uplink. The ADSL downlink and uplink were also the bottleneck links on the forward path and reverse path, respectively. Both nodes were located in Hong Kong, and the forward path consisted of 11 hops.

Same as the last section, the measuring node was equipped with a DAG 4.5 card to measure the RTT and PPD samples. The RTT (PPD) measurement was used for capacity estimation based on the MDDIF (MDSUM) method. Both the ADSL links and the cross traffic on the path could introduce interference to the RTT and PPD measurement which were obtained at the same time. We used $S_f = 1440$ bytes and $S_r = 90$ bytes (all packet sizes include the IP headers) for the forward-path measurement. According to Equation (3.3), this packet size setting ensures that the largest dispersion was introduced by the ADSL downlink. We used $S_f = S_r = 1440$ bytes for the reverse-path measurement.

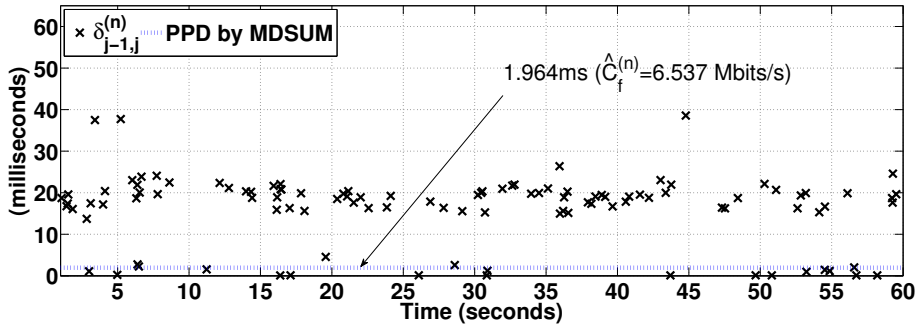
Figures 3.11(a) and 3.11(b) report the PPD samples for the forward-path and reverse-path capacity measurement, respectively. The ranges of the PPD measurement are $[0.01, 5.7]$ milliseconds for the forward path and $[14.7, 20.3]$ milliseconds for the reverse path. The corresponding ranges of the forward-path and reverse-path capacity estimates (denoted by $\hat{C}_f^{(n)}$ and $\hat{C}_r^{(n)}$) are $[2.215, 1272]$ Mbits/s and $[0.627, 0.865]$ Mbits/s, respectively.

We also applied the approach in [70] to account for the layer-two overhead. Since each 1440-byte probe packet was carried by 30 ATM cells, each of which had 53 bytes, we scaled up the capacity estimates by a factor of 1.1 $(1590/1440)$. By using the MDSUM method, we obtained $\hat{C}_f^{(n)} = 6.537$ Mbits/s after processing 140 packet pairs (for which the MDSUM conditions were fulfilled[2]) and $\hat{C}_r^{(n)} = 0.750$ Mbits/s after processing 63 packet pairs.

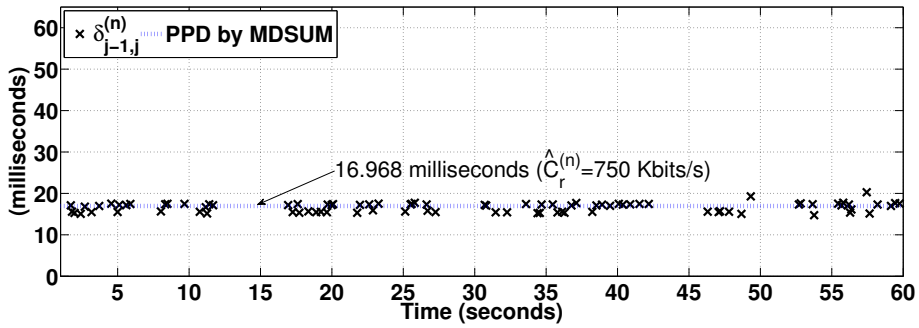
Figures 3.11(c) and 3.11(d), on the other hand, report the first and second probe packets' RTT samples for the MDDIF method. The difference between the first and second probe packet's RTTs is much harder to observe for the forward-path measurement, because the forward-path capacity is an order of magnitude higher than the reverse-path capacity. The MDDIF method obtained fairly accurate results: $\hat{C}_f^{(n)} = 8.01$ Mbits/s and $\hat{C}_r^{(n)} = 0.776$ MBits/s after processing 106 and 22 packet pairs, respectively. Therefore, the MDDIF method improves the MDSUM method in both measurement accuracy and speed. By inspecting the trace, we found that, even though the MDSUM condition had been satisfied for both the forward-path and reverse-path measurements, the minimum delay sums were greater than the corresponding sums of the packet pairs' minDelay, implying that the corresponding PPD samples were still distorted by cross traffic.

The above shows that the MDDIF method can resolve the PPD variability problem observed in an ADSL environment [70]. In particular,

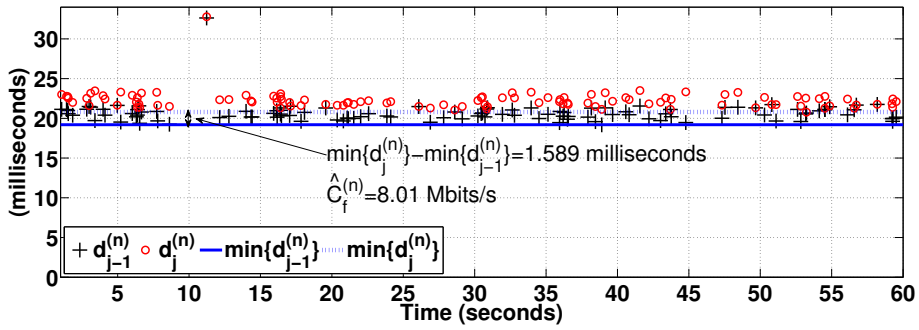
[2]The MDSUM conditions are considered fulfilled when the difference between the left and right hand sides of Equation (3.5) is less than 1%. [115].



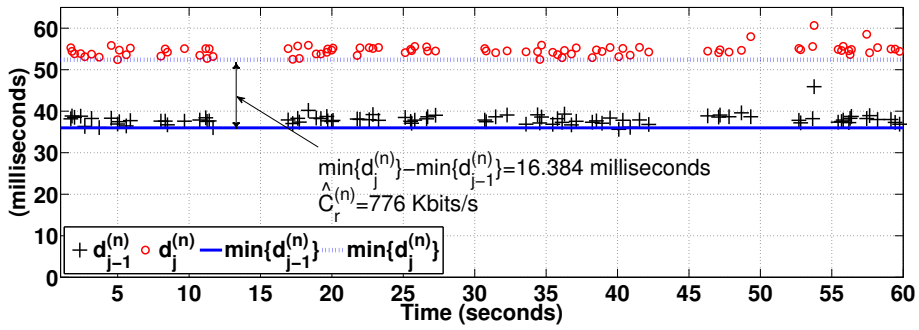
(a) PPD for forward-path capacity measurement.



(b) PPD for reverse-path capacity measurement.



(c) RTT for forward-path capacity measurement.



(d) RTT for reverse-path capacity measurement.

Figure 3.11: Time series of the PPD and RTT samples for the ADSL-at-the-remote-node experiments.

it was reported that the inter-arrival time of adjacent packets under an ADSL environment can vary significantly even in the absence of cross traffic, and such variation could render the PPD techniques ineffective. The MDDIF method, however, does not suffer from this problem, because it neither obtains the PPD directly from adjacent packets nor requires achieving the minimal possible delays from the same packet pair.

3.4.3 Measuring local ADSL links

We conducted another set of capacity measurement experiments by "setting" the forward-path and reverse-path bottleneck links to the local endpoint's links. We achieved this by emulating two types of asymmetric link capacity—ADSL2 (upstream: 1 Mbits/s, downstream: 18 Mbits/s) and ADSL (upstream: 0.8 Mbits/s, downstream: 8 Mbits/s)—using a Click v1.6 router. We deployed HTTP/OneProbe in three separate machines on our campus and used the MDDIF method to measure the capacity of the paths to three PlanetLab nodes: KAIST (in Korea), UMASS (in US), and UNIBO (in Italy). Each machine targeted one of the PlanetLab nodes. Based on our knowledge, the forward-path and reverse-path capacity were limited by the emulated ADSL2/ADSL links.

Besides HTTP/OneProbe, we also attempted to deploy AsymProbe [58], a cooperative measurement tool that implements the MDSUM method, for comparison purposes. Since our campus network blocked all incoming UDP packets used by AsymProbe, we implemented AsymProbe using HTTP/OneProbe's two-packet probe (Figure 3.8(a)) and refer this imple-

mentation to as AProbe. Each machine conducted the HTTP/OneProbe and AProbe measurement for every 15 minutes. HTTP/OneProbe used $S_f = S_r = 1500$ bytes for reverse-path measurement, and $S_f = 1500$ bytes and $S_r = 260$ bytes for forward-path measurement. On the other hand, AProbe used maximum and minimum packet sizes of 1500 bytes and 260 bytes, respectively. Each tool obtained a capacity estimate by processing at most 200 packet pair samples with a fixed probing rate of 2Hz.

Table 3.1 shows the median capacity estimated by HTTP/OneProbe and AProbe based on 24-hour measurement. Measurement results presented in the first two rows show that the capacity measurement obtained by HTTP/OneProbe using the MDDIF method was very accurate. In particular, HTTP/OneProbe could obtain accurate reverse-path estimates even when bottleneck link was in the last hop of the path. On the other hand, AProbe's reverse-path measurement was not accurate, because it could obtain only the forward-path dispersion and therefore the estimates represent the lower bounds for the reverse-path capacity. Nonetheless, AProbe still obtained lower bound values for the two ADSL cases: $1500/260 \times 0.8 = 4.615$ Mbits/s and $1500/260 \times 1$ = 5.769 Mbits/s.

We repeated the experiments with a symmetric network link of 10 Mbits/s which, according to the reasons stated earlier, should be the bottleneck capacity. All other settings were unchanged. As shown in the third row of Table 3.1, HTTP/OneProbe's and AProbe's results were close to 10 Mbits/s. We did not try a higher bandwidth, because we were no longer able to ensure that the bottleneck link was still located in our cam-

pus network.

Table 3.1: Median capacity (in Mbits/s) measured by HTTP/OneProbe and AProbe.

Link types	Tools	KAIST		UMASS		UNIBO	
		$\hat{C}_f^{(n)}$	$\hat{C}_r^{(n)}$	$\hat{C}_f^{(n)}$	$\hat{C}_r^{(n)}$	$\hat{C}_f^{(n)}$	$\hat{C}_r^{(n)}$
ADSL (Up = 0.8, Down = 8)	HTTP/OneProbe	0.799	7.921	0.771	7.926	0.798	7.900
	AProbe	0.786	4.392	0.758	4.544	0.758	4.310
ADSL2 (Up = 1, Down = 18)	HTTP/OneProbe	1.018	17.817	0.962	17.870	0.991	17.804
	AProbe	0.988	5.472	0.989	5.262	1.025	5.300
10 Mbits/s Ethernet link	HTTP/OneProbe	10.025	9.748	10.568	9.748	10.353	9.744
	AProbe	10.592	9.740	9.423	9.748	9.630	9.748

3.5 Discussion

The MDDIF method could take a considerably long time to obtain minDelays of both the first and second probe packets when the cross traffic is intense and non-reactive (e.g., highly congested or UDP-predominant network paths [115]). We also observed from our analytical results that the expected FPT of the MDDIF method (and also the MDSUM method) drastically increases with the number of congested hops in the path. However, such observation is not only true for the MDDIF method, but also for other measurement methods (e.g., Pathchar [104], Nettimer [127], and PingPair [180]) based on the minDelay estimation technique. As a consequence, a careful analysis of the delay samples is required while using those methods. For example, based on our Internet measurement results presented in Section 5.3.2.1, it is clear that the capacity measurement with the MDDIF method should be avoided during the period of

RTT inflation. Moreover, various techniques (e.g., the convergence test [79] and the bootstrap method [127]) have been proposed to detect the convergence of the minDelays.

Another limitation of the MDDIF method is that the method will produce incorrect capacity estimates in the presence of multichannel bottleneck link. Such problem exists in all path capacity measurement methods based on packet pairs and has been discussed in [78, 175]. In particular, a link with $k > 1$ channels will transmit a pair of probe packets in parallel. As a result, the pair will pass through different channels and they will not queue one after another (which violates our assumption). Moreover, following the discussion in [78], the MDDIF method can only measure the peak rate, but not the sustainable rate (which happens after a certain burst size), of a traffic shaper based on the leaky bucket algorithm [216].

By improving the measurement speed, the MDDIF method reduces the measurement interference on the network path. Besides the proof given in Proposition 4 showing the correctness of the MDDIF method, we have also proved the speed advantage of the MDDIF method in Proposition 5, where the MDDIF method can obtain a correct estimate with a fewer number of packet-pair samples comparing with the MDSUM method. The testbed evaluation in Section 3.4.1 also shows that the MDDIF method obtains accurate capacity estimates with significant speed gain under three different cross-traffic scenarios.

Besides the speed advantage, the MDDIF method is also simpler than the MDSUM method in terms of the storage requirement. The MDSUM

method is required to keep track of the minimum delay sum and the minDelay for the first and second packets, and performs a validation test (according to Equation (3.5)) for each measurement. The MDSUM method should also retain the PPD sample having the minimum delay sum. The MDDIF method, however, only needs to store two minDelays. Since the method only searches for minDelays, it promises lower computation costs than the other histogram-based or kernel density-based methods (e.g., Bprobe [47] and nettimer [128]) that require post processing of the measurement data.

3.6 Summary

This chapter introduced the minimum delay difference (MDDIF) method, a new cross-traffic filtering approach for capacity measurement. Unlike the existing packet-pair dispersion methods, the MDDIF method obtains the packet-pair dispersion from the minimal possible delay (minDelay) for a first probe packet and a second probe packet both of which generally belong to different packet pairs. We have proved that a difference of these two minDelays gives the forward-path and reverse-path PPD required for measuring round-trip capacity and asymmetric capacity and that the MDDIF method is faster than the minimum delay sum (MDSUM) method. We also conducted testbed and Internet measurement experiments to compare the MDDIF and MDSUM methods.

4

Measuring Capacity Asymmetry with Three Round-Trip Times

Measuring asymmetric (instead of round-trip) capacity is useful for many existing applications (e.g., file sharing and video streaming applications) whose performance is dominated mainly by a one-way path [196, 220]. Although capacity asymmetry can be measured by deploying a tool at both endpoints of a path (e.g., [43, 59, 60, 78, 174]), this cooperative approach lacks the flexibility to measure arbitrary paths. Therefore, we aim at designing and implementing a new non-cooperative approach that does not require the remote endpoint's cooperation (in terms of setting up additional software).

Among the existing tools given in Table 2.1, only SProbe [197], AsymProbe [58], and DSLprobe [70] measure capacity asymmetry. In particular, the three methods use the well-known packet dispersion method initially proposed for measuring round-trip capacity [121]. The packet dispersion method relies on setting the probe packet much larger (or smaller) than the response packet. This approach, however, introduces two serious limitations to their measurement capability. First, they cannot measure any degree of capacity asymmetry because of the packet size restriction: the maximum packet size is limited by the path MTU to avoid packet fragmentation. Second, they generally cannot support all measurement scenarios, because they may not be able to elicit the required packet size from the remote endpoint. For instance, DSLprobe elicits only small TCP RSTs (but not large response packets) from ADSL endpoints. However, AsymProbe [57] requires installing a UDP server at the cooperative remote endpoint.

In this chapter, we propose TRIO, a new non-cooperative method for measuring capacity asymmetry. TRIO estimates both forward-path capacity and reverse-path capacity by measuring the minimum round-trip times (minRTTs) for three specially crafted probe packets. The three minRTTs are sufficient for computing the packet dispersions on both paths. Since TRIO does not measure the dispersion directly, it eliminates the packet size restriction. As a result, TRIO can measure any degree of capacity asymmetry. By using minRTTs for the computation, TRIO also mitigates cross-traffic impact on the capacity estimates. We have implemented TRIO in HTTP/OneProbe. To further improve the measurement

accuracy, TRIO also provides three self-diagnosis tests to filter out artifacts due to packet reordering and loss, and invalid minRTTs and capacity estimates.

The remainder of this chapter is structured as follows. In Section 4.1, we first present a measurement model and two types of probes for measuring asymmetric capacity. In Section 4.2, we then analyze the properties of using the two types of probes to measure capacity asymmetry. We introduce TRIO and an implementation based on HTTP/OneProbe in Section 4.3. In Section 4.4, we evaluate TRIO's accuracy based on testbed and Internet measurement, and compare TRIO with the existing methods for asymmetric-capacity measurement. We discuss limitations and possible extensions of TRIO in Section 4.5 and summarize this chapter in Section 4.6.

4.1 Model and generalized methods

4.1.1 Measurement model

We consider the same measurement model given in Figure 3.1. A local endpoint measures asymmetric capacity by injecting a sequence of probes, each of which comprises a group of one or more probe packets, to a remote endpoint (the $(m+1)^{th}$ node). Each probe packet elicits one or more response packets from the remote endpoint. Therefore, the measurement packets traverse a *round-trip path* consisting of n hops which starts from and ends at the local endpoint. The first m hops (where

$1 \leq m < n$) of the round-trip path belong to the *forward path*, and the remaining $n - m$ hops to the *reverse path*. Besides the measurement traffic, the path also admits cross traffic that enters and exits from arbitrary hops of the path.

For convenience, we label the hops on the path sequentially and refer $H^{(h)}$ to as the h^{th} hop that comprises a (local, remote, forwarding) node and its outgoing link. Therefore, the local endpoint belongs to both $H^{(1)}$ and $H^{(n+1)}$, and the remote endpoint to $H^{(m+1)}$. At $H^{(h)}$ (for $h = 1, \ldots, n$), packets are transmitted (i.e., serialized) to the outgoing link with a transmission rate of $C^{(h)}$ in bits/s. We define three path capacity metrics:

1. Forward-path capacity $C_f^{(n)} = \min_{1 \leq h \leq m} C^{(h)}$,
2. Reverse-path capacity $C_r^{(n)} = \min_{m+1 \leq h \leq n} C^{(h)}$, and
3. Round-trip capacity $C_b^{(n)} = \min\left\{C_f^{(n)}, C_r^{(n)}\right\}$.

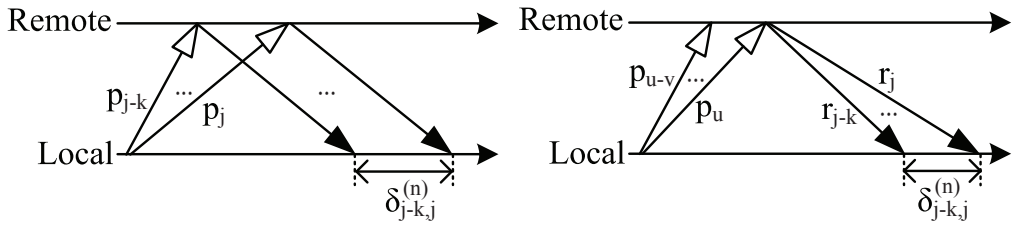
If $C_f^{(n)} \neq C_r^{(n)}$, then the round-trip path is referred to as a *capacity-asymmetric path*. Moreover, Figure 3.1 shows a bottleneck link in the forward path that belongs to a hop denoted by $H^{(h_f)}$, where $h_f = \max\{1 \leq h \leq m : C^{(h)} = C_f^{(n)}\}$. Therefore, if more than one hop in the forward path have the link capacity $C_f^{(n)}$, $H^{(h_f)}$ is referred to the one closest to the remote endpoint. The above is similarly applied to $H^{(h_r)}$ in the reverse path, where $h_r = \max\{m + 1 \leq h \leq n : C^{(h)} = C_r^{(n)}\}$.

The problem tackled in this chapter is for the local endpoint to measure both $C_f^{(n)}$ and $C_r^{(n)}$ of a capacity-asymmetric path. We further classify a capacity-asymmetric path into a *fast-reverse* (FR) path if $C_f^{(n)} < C_r^{(n)}$ or

a *fast-forward* (FF) path if $C_f^{(n)} > C_r^{(n)}$. Let $C_{f/r}^{(n)} = C_f^{(n)}/C_r^{(n)}$. The degree of capacity asymmetry decreases with $C_{f/r}^{(n)}$ for an FR path but increases with $C_{f/r}^{(n)}$ for an FF path. As Table 1.1 suggests, an FR-path example is that the local endpoint is an xDSL/cable user (i.e., $C_{f/r}^{(n)} = C_{up}/C_{dn}$), whereas an FF-path example is that the remote endpoint is using an xDSL/cable connection (i.e., $C_{f/r}^{(n)} = C_{dn}/C_{up}$).

4.1.2 Generalized measurement methods

Before introducing TRIO, it is useful to first generalize the measurement methods used by SProbe [197], AsymProbe [58], and DSLprobe [70] for capacity asymmetry. As illustrated in Figure 4.1, there are two such methods: *round-trip probes* (k-RTP) and *two-way probes* ((v,k)-TWP), where k and v are configurable parameters.



(a) k-RTP. (b) (v,k)-TWP.

Figure 4.1: Two generalized methods for measuring asymmetric capacity.

A k-RTP (where $k \geq 0$) comprises a group of $k+1$ back-to-back probe packets $\{p_{j-k}, \ldots, p_j\}$, each of which elicits a single response packet from the remote endpoint. A 0-RTP therefore comprises a probe packet and the elicited response packet. As shown in Table 4.1, SProbe, AsymProbe, and DSLprobe all use the RTPs to measure the forward-path capacity by

setting $S_f > S_r$, where S_f and S_r are the probe packet size and response packet size in bits, respectively, and receiving the packet dispersion $\delta^{(n)}_{j-k,j}$ from the outgoing link of $H^{(n)}$. The forward-path capacity is then estimated by $kS_f/\delta^{(n)}_{j-k,j}$. AsymProbe and DSLprobe also use the RTPs with $S_f \leq S_r$ for measuring the reverse-path capacity which is estimated by $kS_r/\delta^{(n)}_{j-k,j}$.

On the other hand, a (v,k)-TWP (where $v,k \geq 0$) comprises a sequence of $v+1$ back-to-back probe packets $\{p_{u-v},\ldots,p_u\}$. The probe packets are customized to let the remote endpoint return a sequence of $k+1$ back-to-back response packets in response to p_u's arrival but ignore other preceding packets. If $k > 0$, the local endpoint can measure the packet dispersion $\delta^{(n)}_{j-k,j}$ from the response packets and estimate the reverse-path capacity by $kS_r/\delta^{(n)}_{j-k,j}$. As shown in Table 4.1, SProbe uses the TWPs for the reverse-path measurement.

Table 4.1 also shows in rows 3-5 the capability and limitations of the existing tools. In row 3 (4), we indicate whether a tool can measure $C^{(n)}_f$ and $C^{(n)}_r$ of an FF (FR) path. Note that the three existing tools cannot measure all four cases. In the case of ✓*, the method concerned cannot measure beyond a certain degree of capacity asymmetry. On the other hand, TRIO incorporates both RTPs and TWPs for forward-path measurement and TWPs for reverse-path measurement, and can measure all measurement scenarios. We will explain these results in the next two sections using analytical models and empirical evaluation.

Table 4.1: A summary of the methods (1–2) and capabilities (3–5) used by the existing tools and TRIO for measuring capacity asymmetry. The symbol '✓*' means that the method works for only some measurement scenarios.

	SProbe [197]	AsymProbe [58]	DSLprobe [70]	TRIO
1. Methods for measuring $C_f^{(n)}$	1-RTP ($S_f/S_r = 1500/40$)	1-RTP ($S_f > S_r$)	k-RTP ($S_f/S_r = 1500/40$)	0-RTP & $(1,0)$-TWP ($S_f = S_r$)
2. Methods for Measuring $C_r^{(n)}$	$(0,1)$-TWP ($S_r =$ MTU)	1-RTP ($S_f < S_r$)	k-RTP ($S_f = S_r = 40$)	$(1,1)$-TWP ($S_f = S_r$)
3. FF path ($C_f^{(n)}$,$C_r^{(n)}$)	(✓*,✓)	(✓*,✓)	(✓*,✓)	(✓,✓)
4. FR path ($C_f^{(n)}$,$C_r^{(n)}$)	(✓,✓)	(✓,✓*)	(×,×)	(✓,✓)
5. Non-cooperativeness	Yes	No	Yes	Yes

4.2 Analysis of the round-trip and two-way probes

In this section, we analyze the properties of using the round-trip and two-way probes to measure capacity asymmetry. We adopt the following assumptions in the ensuing analysis, which are commonly found in previous works (e.g., [50, 95, 126]), unless stated otherwise:

1. Both the forward and reverse paths are static and unique during the measurement.

2. Each node is a store-and-forward device using a FCFS queueing discipline.

3. The processing delay is negligible when compared with the transmission delay and propagation delay at each hop.

4. All the probe and response packets are received successfully. Combining with (1)-(3) also implies that the probe packets arrive at the remote endpoint in the original order and similarly for the response

packets.

5. The probes are sufficiently spaced out that a first packet in a probe is never delayed by the preceding probe, and similarly for the response packets.

4.2.1 Preliminaries

We first provide preliminary results based on deterministic models which will be used for deriving the main results in later sections. To derive the preliminary results, we consider a sequence of $k+1$ (where $k > 0$) back-to-back probe packets $\{p_{j-k}, \ldots, p_j\}$ dispatched from $H^{(1)}$. Moreover, whenever necessary, we use p_j to also refer to the elicited response packet.

Packet delay and queueing delay Let $d_j^{(h)},\ h = 0, 1, \ldots, n$, be the *packet delay* for p_j to traverse the first h hops, and $t_j^{(h)},\ h = 1, \ldots, n+1$, be the time for p_j to fully arrive (including the last bit of the packet) at $H^{(h)}$. Therefore, for $h > 0$,

$$\begin{aligned} d_j^{(h)} &= t_j^{(h+1)} - t_j^{(1)}, \\ &= d_j^{(h-1)} + (w_j^{(h)} + X^{(h)} + T^{(h)}), \end{aligned} \tag{4.1}$$

and $d_j^{(0)} = 0$. The delay at $H^{(h)}$ comprises a queueing delay of $w_j^{(h)}$, a constant transmission (serialization) delay of $X^{(h)}$ ($X^{(h)} = S_f/C^{(h)}$ for $1 \leq h \leq m$ and $X^{(h)} = S_r/C^{(h)}$, otherwise), and a constant delay of $T^{(h)}$ for propagating the packet to the next hop.

The *queueing delay* $w_j^{(h)}$ is given by the Lindley's recurrence equation

[122]:

$$w_j^{(h)} = \left(w_{j-1}^{(h)} + X^{(h)} - \delta_{j-1,j}^{(h-1)}\right)^+ + q_{j-1,j}^{(h)}, \tag{4.2}$$

where $(x)^+ = \max\{0, x\}$, $q_{j-1,j}^{(h)}$ is the queueing delay of p_j caused by intervening cross traffic between $\{p_{j-1}, p_j\}$ at $H^{(h)}$, and $\delta_{j-1,j}^{(h)}$ is the *packet dispersion* between p_j and p_{j-1} at the outgoing link of $H^{(h)}$.

Packet dispersion Let $\delta_{j-k,j}^{(h)}$ be the packet dispersion for $\{p_{j-k}, \ldots, p_j\}$ at the outgoing link of $H^{(h)}$. For $k = 1$, the packet dispersion is a *packet-pair dispersion* (PPD); for $k > 1$, it is a *packet-train dispersion* (PTD). Without loss of generality, we assume that the $k + 1$ packets arrive at $H^{(1)}$ at the same time instance (i.e., $t_j^{(1)} = t_{j-1}^{(1)} = \ldots = t_{j-k}^{(1)}$). The packet dispersion is thus given by

$$\begin{aligned} \delta_{j-k,j}^{(h)} &= t_j^{(h+1)} - t_{j-k}^{(h+1)}, \\ &= d_j^{(h)} - d_{j-k}^{(h)} + \delta_{j-k,j}^{(0)}, \end{aligned} \tag{4.3}$$

where $\delta_{j-k,j}^{(0)} \equiv t_j^{(1)} - t_{j-k}^{(1)} = 0$.

Lemma 4 below gives two expressions for $\delta_{j-k,j}^{(h)}$—one based on p_j's queueing delay and a recursive relation—when the probe and response packets do not experience queueing delay induced by cross traffic.

Lemma 4. *Considering that* $\{p_{j-k}, \ldots, p_j\}$ *do not experience queueing delay caused by cross traffic on a path segment* $\{H^{(1)}, \ldots, H^{(h)}\}$*, then* $\{p_{j-k}, \ldots, p_j\}$*'s*

dispersion at the outgoing link of $H^{(h)}$ is given by

$$\begin{aligned} \delta^{(h)}_{j-k,j} &= \sum_{l=1}^{h} w^{(l)}_{j}, && (4.4) \\ &= \delta^{(h-1)}_{j-k,j} + \left(kX^{(h)} - \delta^{(h-1)}_{j-k,j}\right)^{+}, && (4.5) \end{aligned}$$

where $\delta^{(0)}_{j-k,j} = 0$.

Proof. If the probe and response packets do not experience cross-traffic induced queueing delay, then $w^{(l)}_{j-k} = q^{(l)}_{j-1,j} = 0,\ \forall 1 \le l \le h$. For Equation (4.4), we substitute Equation (4.1) for p_j and p_{j-k} in Equation (4.3) and then set $w^{(l)}_{j-k} = 0,\ l = 1, \ldots, h$, to obtain

$$\delta^{(h)}_{j-k,j} = d^{(h)}_{j} - d^{(h)}_{j-k} = \sum_{l=1}^{h} \left(w^{(l)}_{j} - w^{(l)}_{j-k}\right) = \sum_{l=1}^{h} w^{(l)}_{j}.$$

For Equation (4.5), we first replace $w^{(l)}_{j}$ in Equation (4.4) by Equation (4.2). We next set $q^{(l)}_{j-i,j-i+1} = 0,\ \forall 1 \le l \le h,\ \forall 1 \le i \le k$, and use Equation (4.2) to expand $w^{(l)}_{j-1}$ until obtaining $w^{(l)}_{j-k}$. Since $w^{(l)}_{j-k} = 0,\ \forall 1 \le l \le h$, and $\delta^{(l-1)}_{j-k,j} = \sum_{i=1}^{k} \delta^{(l-1)}_{j-i,j-i+1}$, we obtain

$$\begin{aligned} \delta^{(h)}_{j-k,j} &= \sum_{l=1}^{h} \left[\left(w^{(l)}_{j-1} + X^{(l)} - \delta^{(l-1)}_{j-1,j}\right)^{+} + q^{(l)}_{j-1,j}\right], \\ &= \sum_{l=1}^{h} \left(w^{(l)}_{j-k} + kX^{(l)} - \sum_{i=1}^{k} \delta^{(l-1)}_{j-i,j-i+1}\right)^{+}, \\ &= \sum_{l=1}^{h} \left(kX^{(l)} - \delta^{(l-1)}_{j-k,j}\right)^{+}. && (4.6) \end{aligned}$$

which is the same as Equation (4.5). □

4.2.2 k-round-trip probe

The existing tools rely on an unbiased packet dispersion obtained from a k-RTP (where $k > 0$) to estimate the path capacity. The packet dispersion is unbiased if all the probe and response packets do not suffer from queueing delay induced by cross traffic throughout the round-trip path. We first present in Proposition 7 the unbiased packet dispersion obtained by a k-RTP.

Proposition 7. *Considering that $\{p_{j-k}, \ldots, p_j\}$ of a k-RTP (where $k > 0$) do not experience queueing delay caused by cross traffic on an n-hop round-trip path with $C_f^{(n)}$ and $C_r^{(n)}$, the unbiased packet dispersion of $\{p_{j-k}, \ldots, p_j\}$ at the outgoing link of $H^{(n)}$ is given by*

$$\delta_{j-k,j}^{(n)} = \begin{cases} kX^{(h_f)}, & \text{if } S_{f/r} > C_{f/r}^{(n)}, \\ kX^{(h_r)}, & \text{if } S_{f/r} < C_{f/r}^{(n)}, \\ kX^{(h_f)} = kX^{(h_r)}, & \text{otherwise,} \end{cases} \tag{4.7}$$

where $X^{(h_f)} = S_f/C_f^{(n)}$ and $X^{(h_r)} = S_r/C_r^{(n)}$ are the forward-path PPD and reverse-path PPD, respectively, and $S_{f/r} = S_f/S_r$ is the degree of packet-size asymmetry.

Proof. By applying Equation (4.5) to $\delta_{j-k,j}^{(n)}$ recursively,

$$\delta_{j-k,j}^{(n)} = \max_{h=1,\ldots,n} \left\{ kX^{(h)} \right\} = k \left(\max \left\{ \frac{S_f}{C^{(h_f)}}, \frac{S_r}{C^{(h_r)}} \right\} \right), \tag{4.8}$$

which is equivalent to Equation (4.7). □

According to Proposition 7, the local endpoint must choose a suit-

able $S_{f/r}$ to obtain an appropriate packet dispersion for capacity measurement. Let S_{max} and S_{min} be the maximally and minimally permitted packet sizes, where $S_{max} \geq S_{min}$. A sound measurement strategy is then setting $S_{f/r} = S_{max}/S_{min} \geq 1$ to receive $\delta^{(n)}_{j-k,j} = kX^{(h_f)}$ (for measuring $C^{(n)}_f$) and $S_{f/r} = S_{min}/S_{max} \leq 1$ to receive $\delta^{(n)}_{j-k,j} = kX^{(h_r)}$ (for measuring $C^{(n)}_r$). As a result, both $C^{(n)}_f$ and $C^{(n)}_r$ can be estimated by $kS_{max}/\delta^{(n)}_{j-k,j}$.

Table 4.2 enumerates all possible scenarios of using k-RTP to measure $C^{(n)}_f$ and $C^{(n)}_r$ for the FF path ($C^{(n)}_{f/r} > 1$) and FR path ($C^{(n)}_{f/r} < 1$). The two correct cases for the $C^{(n)}_f$ measurement, labeled by (1) and (3), meet the condition $S_{f/r} \geq C^{(n)}_{f/r}$ to obtain $\delta^{(n)}_{j-k,j} = kX^{(h_f)}$, whereas the two correct cases for the $C^{(n)}_r$ measurement, labeled by (2) and (4), meet the condition $S_{f/r} \leq C^{(n)}_{f/r}$ to obtain $\delta^{(n)}_{j-k,j} = kX^{(h_r)}$. On the other hand, the two incorrect cases (I) and (II) do not obtain an appropriate dispersion due to the insufficient degree of packet-size asymmetry.

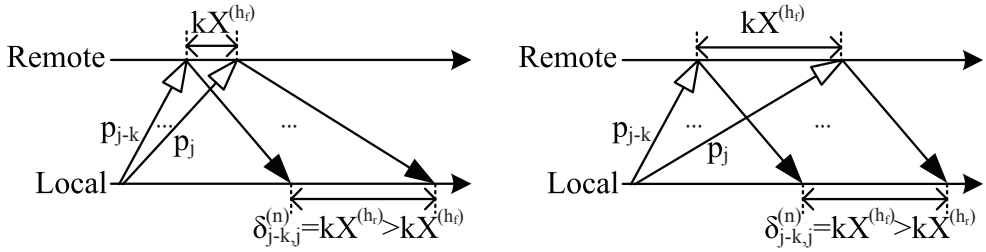
Table 4.2: The six scenarios of using k-RTP to measure $C^{(n)}_f$ and $C^{(n)}_r$ for the FF and FR paths.

	FF path ($C^{(n)}_{f/r} > 1$)	FR path ($C^{(n)}_{f/r} < 1$)
Using $S_{f/r} \geq 1$ to measure $C^{(n)}_f$	(1) $S_{f/r} \geq C^{(n)}_{f/r} > 1$ or (I) $C^{(n)}_{f/r} > S_{f/r} \geq 1$	(3) $S_{f/r} \geq 1 > C^{(n)}_{f/r}$
Using $S_{f/r} \leq 1$ to measure $C^{(n)}_r$	(2) $S_{f/r} \leq 1 < C^{(n)}_{f/r}$	(4) $S_{f/r} \leq C^{(n)}_{f/r} < 1$ or (II) $C^{(n)}_{f/r} < S_{f/r} \leq 1$

Unfortunately, the packet size restriction limits the degree of packet-size asymmetry and therefore the usefulness of k-RTP. In particular, the probe (response) packets' size should not exceed the path MTU, which defines the maximum size of an IP packet allowed to transmit on the

path without fragmentation. However, if the packets are fragmented before the bottleneck, $\delta^{(n)}_{j-k,j}$ measured from the reassembled packets will include additional transmission delay for the fragments' IP header at the bottleneck, thus introducing capacity underestimation.

Figure 4.2 shows the two types of interferences encountered by k-RTP due to the limited degree of packet-size asymmetry. Using $S_{max} = 1500$ bytes (a typical MTU value) and $S_{min} = 240$ bytes as an example, we have $S_{f/r} = 1500/240 = 6.25$ for measuring $C^{(n)}_f$ and $S_{f/r} = 240/1500 = 0.16$ for measuring $C^{(n)}_r$. According to Table 1.1, the $C^{(n)}_f$ measurement with remote xDSL users will suffer *response interference* because $C^{(n)}_{f/r} = C_{dn}/C_{up} > 6.25$ (case (I) in Table 4.2). As shown in Figure 4.2(a), the response interference causes the incorrect dispersion $\delta^{(n)}_{j-k,j} = kX^{(h_r)} > kX^{(h_f)}$, thus introducing $C^{(n)}_f$ underestimation. This also explains why SProbe, AsymProbe, and DSLprobe cannot measure $C^{(n)}_f$ of FF paths accurately for all measurement scenarios, as indicated by ✓* for $C^{(n)}_f$ in row 3 of Table 4.1. On the other hand, the $C^{(n)}_r$ measurement performed by local xDSL users will suffer *probe interference* because $C^{(n)}_{f/r} = C_{up}/C_{dn} < 0.16$ (case (II) in Table 4.2). Figure 4.2(b) shows that the probe interference causes the incorrect dispersion $\delta^{(n)}_{j-k,j} = kX^{(h_f)} > kX^{(h_r)}$, thus introducing $C^{(n)}_r$ underestimation. This also explains ✓* for AsymProbe in row 4 of Table 4.1. Moreover, based on the S_f and S_r values, we can see that DSLprobe is designed only for FF paths.



(a) Response interference on the $C_f^{(n)}$ measurement when $C_{f/r}^{(n)} > S_{f/r} \geq 1$.

(b) Probe interference on the $C_r^{(n)}$ measurement when $C_{f/r}^{(n)} < S_{f/r} \leq 1$.

Figure 4.2: The two types of interferences encountered by k-RTP due to the limited degree of packet-size asymmetry.

4.2.3 (v,k)-two-way probe

We present in Proposition 8 the unbiased packet dispersion of $\{r_{j-k},\ldots,r_j\}$ obtained by a (v,k)-TWP.

Proposition 8. *Considering that $\{r_{j-k},\ldots,r_j\}$ elicited by a (v,k)-TWP (where $k>0$) do not experience queueing delay caused by cross traffic on the reverse path with $C_r^{(n)}$, the unbiased packet dispersion of $\{r_{j-k},\ldots,r_j\}$ at the outgoing link of $H^{(n)}$ is given by*

$$\delta_{j-k,j}^{(n)} = \frac{kS_r}{C_r^{(n)}}. \tag{4.9}$$

Proof. Since $\{r_{j-k},\ldots,r_j\}$ are all elicited by p_u, we can assume that they have the same arrival times $t_j^{(m+1)} = t_{j-1}^{(m+1)} = \ldots = t_{j-k}^{(m+1)}$ at $H^{(m+1)}$ and $\delta_{j-k,j}^{(m)} = 0$. Therefore, we can regard $\{r_{j-k},\ldots,r_j\}$ as a k-RTP dispatched by the remote endpoint to the local endpoint (i.e., from $H^{(m+1)}$ to $H^{(n+1)}$). Similar to the proof for Proposition 7, by recursively applying Equation (4.5) to $\delta_{j-k,j}^{(n)}$, we obtain Equation (4.9). □

We notice from Proposition 8 two nice properties about the (v,k)-TWP measurement. First, the reverse-path capacity, given by $kS_r/\delta^{(n)}_{j-k,j}$, is independent of v (and therefore the probe packets preceding p_u). As will be shown in the next section, this property is exploited by TRIO to use the same TWP to measure both $C^{(n)}_f$ and $C^{(n)}_r$. Second, Equation (4.9) shows that the packet dispersion of $\{r_{j-k},\ldots,r_j\}$ is independent of the forward-path dispersion. Thus, the TWP measurement is immune from the probe interference.

Table 4.1 shows that SProbe is the only existing tool that exploits the (v,k)-TWP ($v=0$ and $k=1$) to measure $C^{(n)}_r$. Specifically, SProbe dispatches an HTTP `GET` request to induce a pair of S_r-byte TCP data packets from a web server for the measurement, where S_r depends on the negotiated maximum segment size (MSS) which is upper bounded by the path MTU. However, our empirical evaluation of SProbe (to be presented in Section 4.4) shows that its reverse-path measurement is often inaccurate, because the response packets are not dispatched consecutively.

4.2.4 Testbed experiments

In addition to the analysis, we also conducted testbed experiments to evaluate the impact of probe and response interferences on the RTP measurement and the properties of the TWP measurement. The testbed, shown in Figure 4.3, was configured with a 12-hop round-trip path ($n=12$), consisting of a probe sender, a web server running Apache v2.2.3 as the destination node, three cross-traffic clients ($\mathbb{X}_1-\mathbb{X}_3$), and five forwarding

devices: two Linux 2.6.26 routers ($\mathbb{R}_1 - \mathbb{R}_2$) and three 100 Mbits/s Ethernet switches ($\mathbb{S}_1 - \mathbb{S}_3$).

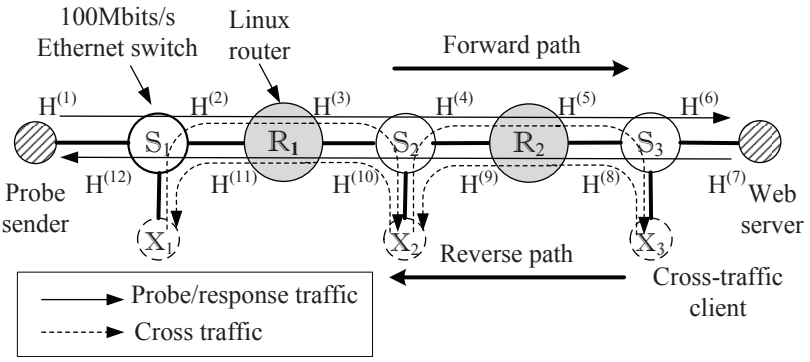


Figure 4.3: The testbed topology.

Moreover, each cross-traffic client generated forward-path (reverse-path) cross traffic to other cross-traffic client to the right (left) to emulate a loading rate ρ of 20% on the corresponding path segment. Similar to [78], the cross-traffic packets' inter-arrival time follows the Pareto distribution with a shape parameter of $\alpha = 1.9$ (i.e., the inter-arrival time has infinite variance), and the IP packet size is uniformly distributed over $[40, 1500]$ bytes. The probe sender dispatched a sequence of Poisson-modulated 1-RTPs with a mean probing rate of 2 Hz. It was also equipped with a DAG card [4] to obtain RTT samples in microsecond resolution which is limited by the `pcap` header structure [18].

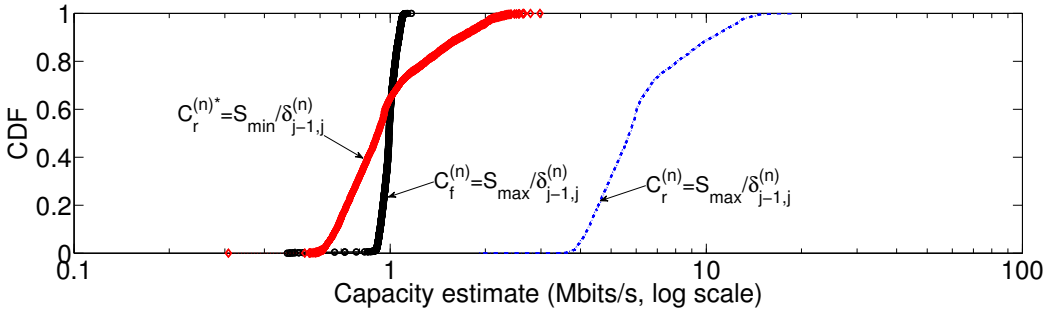
4.2.4.1 The RTP measurement

FR path We emulated an FR path with both forward-path and reverse-path bottlenecks situated close to the probe sender by running Click v1.8 [123] in $\mathbb{R}_1$ to emulate $C_f^{(n)} = C^{(3)} = 1$ Mbit/s at $H^{(3)}$ and $C_r^{(n)} = C^{(11)} = 24$

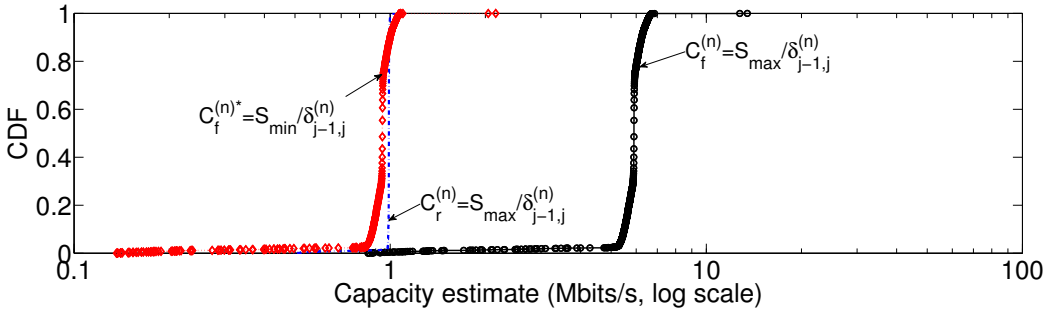
Mbits/s at $H^{(11)}$. The Click router was also configured to set the RTT between the probe sender and web server to 300 ms. We used IP packet sizes $S_{max}/S_{min} = 1500/240$ (in bytes) to obtain 3500 estimates for $C_f^{(n)}$ and $C_r^{(n)}$ without using any cross-traffic filtering technique, and their CDFs are plotted in Figure 4.4(a).

Figure 4.4(a) shows that the 1-RTP measurement for $C_f^{(n)}$ (case (3) in Table 4.2) is very accurate: 99% of the estimates fall in the range of [0.87,1.10] Mbits/s. However, due to the probe interference (case (II) in Table 4.2), the 1-RTP measurement significantly underestimates $C_r^{(n)}$ with 99% of the estimates falling in the range of [3.63,14.18] Mbits/s. Moreover, we notice a large variation in the $C_r^{(n)}$ estimates, which could be reduced by a filtering technique. Besides, we have also included $C_r^{(n)*} = S_{min}/\delta_{j-1,j}^{(n)}$ which estimates $C_r^{(n)}$ using S_{min} (instead of S_{max}). Therefore, $C_r^{(n)*}$'s CDF is just a left shift of the $C_r^{(n)}$'s CDF, and $C_r^{(n)*}$ actually measures $C_f^{(n)}$. This explains why $C_f^{(n)}$'s and $C_r^{(n)*}$'s CDFs are close to each other.

FF path We repeated the experiments by designating $\mathbb{R}_2$ to emulate an FF path with the bottlenecks close to the web server. We restored $C^{(3)} = C^{(11)} = 100$ Mbits/s and ran Click in $\mathbb{R}_2$ to emulate $C_f^{(n)} = C^{(5)} = 24$ Mbits/s at $H^{(5)}$ and $C_r^{(n)} = C^{(9)} = 1$ Mbit/s at $H^{(9)}$. Other parameters remained unchanged. Figure 4.4(b) shows that the 1-RTP measurement for $C_r^{(n)}$ (case (2) in Table 4.2) is very accurate: 99% of the estimates fall in the range of [0.73,1.01] Mbits/s. However, due to the response interference (case (I) in Table 4.2), $C_f^{(n)}$ is significantly underestimated with 99% of the estimates falling in the range of [1.18,6.75] Mbits/s). We have also included $C_f^{(n)*} = S_{min}/\delta_{j-1,j}^{(n)}$ which estimates $C_f^{(n)}$ using S_{min}. Similar to



(a) An FR path with $C_{f/r}^{(n)} = 1/24$ (in Mbits/s).



(b) An FF path with $C_{f/r}^{(n)} = 24/1$ (in Mbits/s).

Figure 4.4: CDF of the capacity estimates obtained from 1-RTPs with $S_{max}/S_{min} = 1500/240$ (in bytes) and $\rho = 20\%$.

the FR path, $C_r^{(n)}$'s and $C_f^{(n)*}$'s CDFs are close to each other, because $C_f^{(n)*}$ actually measures $C_r^{(n)}$.

4.2.4.2 The TWP measurement

We configured the testbed with $C_{f/r}^{(n)} = 1/24$ (in Mbits/s) using $\mathbb{R}_1$ and measured $C_r^{(n)}$ using $(0,1)$-TWPs and $(1,1)$-TWPs with $S_f \in \{240, 1500\}$ bytes (to induce different forward-path PPDs) and $S_r = 1500$ bytes. As recalled, the previous 1-RTP measurement fails to obtain a correct $C_r^{(n)}$ for this path due to the probe interference. Similar to before, we did not apply any cross-traffic filtering technique to the analysis. Other configu-

ration settings were unchanged. Figure 4.5 shows that more than 60% of the TWPs obtain $C_r^{(n)}$ with less than 2% of error, regardless of the number of probe packets and probe packet size. Moreover, the TWP measurement does not overestimate $C_r^{(n)}$, because the cross traffic did not exist after the reverse-path bottleneck that would otherwise compress the PPD. The underestimation, on the other hand, was the result of expanding the PPD by the intervening cross traffic present at or before the reverse-path bottleneck.

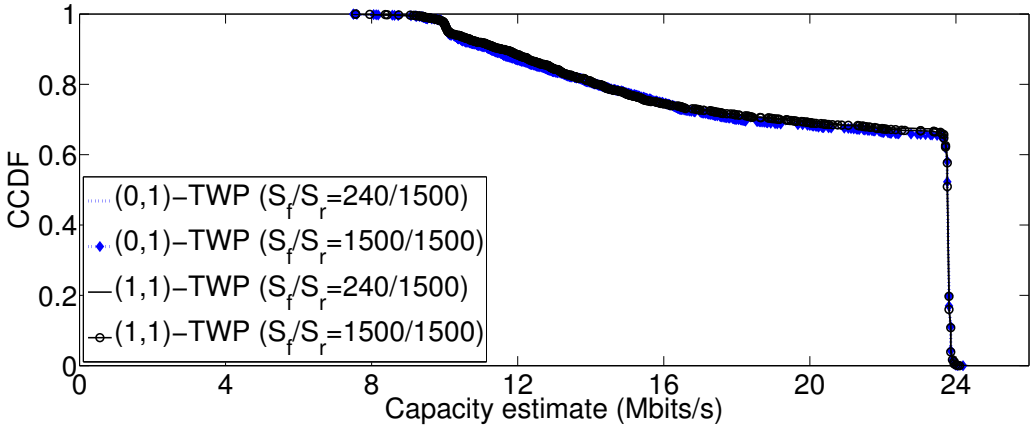


Figure 4.5: CCDF of the capacity estimates obtained by $(0,1)$-TWPs and $(1,1)$-TWPs with $S_f \in \{240, 1500\}$ bytes, $S_r = 1500$ bytes, and $\rho = 20\%$.

4.3 TRIO

This section introduces TRIO that exploits both k-RTP and (v,k)-TWP to measure capacity asymmetry. Figure 4.6 illustrates an i^{th} TRIO's probe consisting of a 1-RTP (denoted by P_i^R) and a $(1,1)$-TWP (denoted by P_i^T). In P_i^R, two back-to-back probe packets (p_{j-1}^R and p_j^R, where $j = 2i$) are

dispatched, and each elicits a response packet. In P_i^T, two back-to-back probe packets are also dispatched (p_{j-1}^T and p_j^T, where $j = 2i$), but only p_j^T elicits two response packets (r_{j-1}^T and r_j^T, where $j = 2i$). Moreover, the only requirement on the packet size is that the probe packets for 1-RTP and $(1,1)$-TWP share the same size, and similarly for their elicited response packets. To make it simple, we just assume that all probe and response packets have the same size $S_f = S_r = S$.

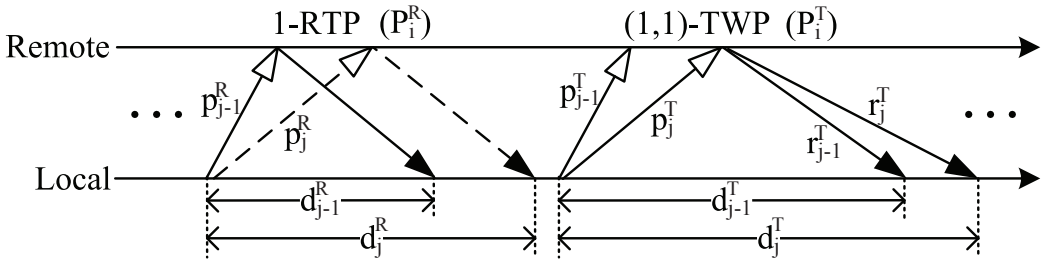


Figure 4.6: A TRIO's probe consisting of a 1-RTP and a $(1,1)$-TWP.

Unlike the existing methods, TRIO does not measure the packet dispersion directly. Instead, it measures three RTTs—d_{j-1}^R, d_{j-1}^T, and d_j^T—from the RTP and TWP, as shown in Figure 4.6. It then uses d_{j-1}^R and d_{j-1}^T to estimate the forward-path capacity and d_{j-1}^T and d_j^T the reverse-path capacity. Note that d_j^R is not required for the capacity measurement; therefore, we use dotted line for the corresponding probe and response packets. An important advantage of admitting RTT as the basic unit for capacity measurement is removing the probe and response interferences. As a result, TRIO can measure both FF and FR paths with any degree of capacity asymmetry. Another advantage is the ability of filtering RTT samples that are biased by cross traffic.

There are two important points worth noting about TRIO's measure-

ment methods. First, for the sole purpose of measuring the forward-path capacity, using 0-RTP and $(1,0)$-TWP actually suffices, because they can provide both d^R_{j-1} and d^T_{j-1} required for the measurement. However, TRIO uses $(1,1)$-TWP instead to conduct the reverse-path measurement at the same time. Second, TRIO uses 1-RTP (instead of 0-RTP) to additionally obtain d^R_j which, as we will see shortly, is used for measurement validation.

4.3.1 Measurement methods

TRIO obtains the minimums of d^R_{j-1}, d^R_j, d^T_{j-1}, and d^T_j (i.e., minRTTs) from a sequence of probes. A probe packet's minRTT is the RTT experienced by the probe packet and the elicited response packet, but the packets do not encounter any cross-traffic-induced queueing delay on the path (Section 3.2.2). Accordingly, a minRTT could still include the queueing delay induced by the preceding packets belonging to the same probe. By sending a sufficiently long sequence of $\{P^R_i, P^T_i\}$, we assume (similarly as [79, 127]) that the minimum observable values of d^R_{j-1}, d^R_j, d^T_{j-1}, and d^T_j from the sequence converge to their corresponding minRTTs.

4.3.1.1 Measuring the forward-path capacity

With the minimums of d^R_{j-1} and d^T_{j-1}, TRIO can estimate the forward-path PPD for arbitrary packet size S and avoid the response interference. In particular, Proposition 9 shows that the forward-path PPD can be obtained by subtracting the minimum of d^R_{j-1} from the minimum of d^T_{j-1}.

Proposition 9. *Consider that* $\{P_i^R, P_i^T\},\ i = 1, 2, \ldots,$ *with* $S_f = S_r = S$ *are dispatched on an* n*-hop round-trip path with* $C_f^{(n)}$ *and* $C_r^{(n)}$*. If both* d_{2x-1}^R *(obtained from* P_x^R*) and* d_{2y-1}^T *(obtained from* P_y^T*) are minRTTs, then their difference gives the forward-path PPD:*

$$d_{2y-1}^T - d_{2x-1}^R = \frac{S}{C_f^{(n)}}. \tag{4.10}$$

Proof. First of all, based on Figure 4.6 and using Equation (4.1),

$$d_{2x-1}^R = \sum_{h=1}^{n} \left(w_{2x-1}^{(h)} + X^{(h)} + T^{(h)} \right), \tag{4.11}$$

$$d_{2y-1}^T = \sum_{h=1}^{m} w_{2y}^{(h)} + \sum_{h=m+1}^{n} w_{2y-1}^{(h)} + \sum_{h=1}^{n} \left(X^{(h)} + T^{(h)} \right). \tag{4.12}$$

Since d_{2x-1}^R and d_{2y-1}^T are minRTTs, $\sum_{h=1}^{n} w_{2x-1}^{(h)} = 0$ and $\sum_{h=m+1}^{n} w_{2y-1}^{(h)} = 0$. By subtracting Equation (4.11) from Equation (4.12) and then recursively applying Equation (4.5) to $\delta_{2y-1,2y}^{(m)}$, we obtain $d_{2y-1}^T - d_{2x-1}^R = \sum_{h=1}^{m} w_{2y}^{(h)} = \max_{h=1,\ldots,m} \left\{ X^{(h)} \right\} = S/C^{(h_f)} \equiv S/C_f^{(n)}$. □

4.3.1.2 Measuring the reverse-path capacity

TRIO can estimate the reverse-path PPD for arbitrary packet size S and avoid the probe interference by subtracting the minimum of d_{j-1}^T from the minimum of d_j^T. Using Proposition 8 and Equation (4.3), we have $S_r/C_r^{(n)} = \delta_{j-1,j}^{(n)} \equiv d_j^T - d_{j-1}^T = d_{2y}^T - d_{2x-1}^T$, where d_{2x-1}^T (obtained from P_x^T) and d_{2y}^T (obtained from P_y^T) are minRTTs. Therefore, we report the result as a Corollary 1 that directly follows from Proposition 8.

Corollary 1. *Consider that* $\{P_i^R, P_i^T\},\ i = 1, 2, \ldots,$ *with* $S_f = S_r = S$ *are dispatched on an* n*-hop round-trip path with* $C_f^{(n)}$ *and* $C_r^{(n)}$*. If both* d_{2x-1}^T *(obtained from* P_x^T*) and* d_{2y}^T *(obtained from* P_y^T*) are minRTTs, then their difference gives the reverse-path PPD:*

$$d_{2y}^T - d_{2x-1}^T = \frac{S}{C_r^{(n)}}. \tag{4.13}$$

Similar to SProbe, TRIO also exploits solely the TWP to eliminate the probe interference by eliciting two back-to-back response packets. However, TRIO also makes use of the TWP for the forward-path measurement. Moreover, TRIO does not measure the PPD directly from a single TWP but obtains the minRTT from a sequence of TWPs.

4.3.2 Implementation

We implemented TRIO using HTTP/OneProbe's probing technique. Each probe packet in the 1-RTP and $(1,1)$-TWP is a TCP data packet that carries a legitimate HTTP `GET` request of specific length; each response packet is also a TCP data packet that contains the requested HTTP data. The size of the response packet can be manipulated using the MSS advertisement option. Therefore, each 1-RTP is basically the same as the HTTP/OneProbe's probe. To implement $(1,1)$-TWP, TRIO inserts a zero-byte RWND in the first probe packet and a $(2\times\text{MSS})$-byte RWND in the second probe packet that elicits a pair of back-to-back response packets immediately (Section 2.4.2.6). Moreover, it has been shown that the HTTP/OneProbe's probe can mitigate the substantial processing overhead at the remote

web server [137]. This is, however, not true for other HTTP-based RTT measurement tools, such as, httping [98]. To improve the measurement accuracy, TRIO performs three types of self-diagnoses to be described next.

4.3.2.1 Self diagnosis for packet loss and reordering

Using the HTTP/OneProbe's probing technique, TRIO can detect loss and reordering of individual probe and response packets that could significantly affect the measurement accuracy [174]. The detection is performed based on the expected TCP response packet patterns. When a TCP probe packet is lost, for instance, the TCP response packets will be different from the ones that would be normally elicited by a pair of TCP probe packets. As a result, TRIO removes all packet pairs that do not elicit the expected TCP response packets to ensure that all RTT samples used for the capacity measurement come from lossless and order-preserved probe and response packets.

4.3.2.2 Self diagnosis for the minRTT estimates

Incorrect estimates of the minimums of d^R_{j-1}, d^R_j, d^T_{j-1}, and d^T_j will significantly affect the forward-path and reverse-path measurements. TRIO validates the minRTT estimates based on the following inequality:

$$\min_i \left\{d^R_{2i-1}\right\} < \min_i \left\{d^T_{2i-1}\right\} \leq \min_i \left\{d^R_{2i}\right\} < \min_i \left\{d^T_{2i}\right\}. \tag{4.14}$$

Following Proposition 9, it is obvious that $\min_i \left\{d^R_{2i-1}\right\} < \min_i \left\{d^T_{2i-1}\right\}$.

Moreover, by inspecting Figure 4.6, it is not difficult to see that

$$\min_i \left\{d_{2i}^R\right\} - \min_i \left\{d_{2i-1}^T\right\} = \min_i \left\{ \sum_{h=m+1}^{n} w_{2i}^{(h)} \right\} \geq 0, \tag{4.15}$$

$$\begin{aligned} \min_i \left\{d_{2i}^T\right\} - \min_i \left\{d_{2i}^R\right\} &= \frac{S_f}{C^{(h_f)}} + \frac{S_r}{C^{(h_r)}} - \max\left\{\frac{S_f}{C^{(h_f)}}, \frac{S_r}{C^{(h_r)}}\right\}, \\ &= \min\left\{\frac{S_f}{C^{(h_f)}}, \frac{S_r}{C^{(h_r)}}\right\} > 0. \end{aligned} \tag{4.16}$$

Therefore, $\min_i \left\{d_{2i-1}^T\right\} \leq \min_i \left\{d_{2i}^R\right\} < \min_i \left\{d_{2i}^T\right\}$.

4.3.2.3 Self diagnosis for the capacity estimates

By obtaining the minimum of d_j^R from the 1-RTPs, TRIO can additionally estimate the round-trip capacity using the MDDIF method:

$$\hat{C}_b^{(n)} = \frac{S}{\left(\min_i \left\{d_{2i}^R\right\} - \min_i \left\{d_{2i-1}^R\right\}\right)}. \tag{4.17}$$

Moreover, Equation (4.16) suggests that TRIO can estimate the capacity for the faster uni-directional path denoted by $\hat{C}_B^{(n)} = \max\left\{C_f^{(n)}, C_r^{(n)}\right\}$ using the minimums of d_j^R and d_j^T:

$$\hat{C}_B^{(n)} = \frac{S}{\left(\min_i \left\{d_{2i}^T\right\} - \min_i \left\{d_{2i}^R\right\}\right)}. \tag{4.18}$$

Given a forward-path capacity estimate ($\hat{C}_f^{(n)}$) and a reverse-path capacity estimate ($\hat{C}_r^{(n)}$) obtained by TRIO, if either Equation (4.19) or Equation (4.20) cannot be fulfilled after sending a predefined number of probes, it is likely that the minRTT estimates for d_{j-1}^R, d_j^R, d_{j-1}^T, and d_j^T have yet to

converge[1].

$$\hat{C}_b^{(n)} = \min\left\{\hat{C}_f^{(n)}, \hat{C}_r^{(n)}\right\}, \tag{4.19}$$

$$\hat{C}_B^{(n)} = \max\left\{\hat{C}_f^{(n)}, \hat{C}_r^{(n)}\right\}. \tag{4.20}$$

To sum up, if either Equation (4.14), Equation (4.19), or Equation (4.20) do not hold, TRIO invalidates the current forward-path and reverse-path capacity estimates and keeps sending more probes until the three equations are fulfilled.

4.4 Evaluation

In this section, we evaluate TRIO empirically and compare it with SProbe, AsymProbe, and DSLprobe, whenever possible, based on a testbed and a set of Internet paths.

4.4.1 Testbed evaluation

The testbed is the same as Figure 4.3, except that we inserted a Linux router ($\mathbb{R}_3$) and a 100 Mbits/s Ethernet switch ($\mathbb{S}_4$) between $\mathbb{S}_3$ and the web server, and attached a cross-traffic client ($\mathbb{X}_4$) to $\mathbb{S}_4$. We designated $\mathbb{R}_1$ to emulate ten different cases of capacity asymmetry and a fixed RTT of 300 ms, and emulated a loading rate ρ of 20% on the corresponding path segments. Other configuration settings remained unchanged. As

[1]For Equations (4.19) and (4.20), we assume that the left and right hand sides are equal when their difference is less than 1% to deal with the measurement system's imprecision.

shown in Table 4.3, the first five cases of capacity asymmetry correspond to FR paths, whereas the next five cases FF paths.

The probe sender ran SProbe, AsymProbe, and DSLprobe with most of their default configuration settings unchanged for 30 times. To obtain a fair comparison, we set $S_{max}/S_{min} = 1500/40$ (in bytes) for all of them. We also repeated the experiments with TRIO that dispatched an interleaved sequence of Poisson-modulated 1-RTPs and $(1,1)$-TWPs with $S = 1500$ bytes and a mean probing rate of 2 Hz for 300 seconds. Moreover, we discounted DSLprobe's reverse-path estimate by a factor of 2.65 [70] for adjusting the layer-2 overhead due to the ADSL link. To compute the layer-two (Ethernet) capacity, we then applied a factor of $1518/1500$ (1518 bytes is the maximum Ethernet frame size) to its forward-path estimate and a factor of $64/40$ (64 bytes is the minimum Ethernet frame size) to its reverse-path estimate. Similarly, we applied the same factor of $1518/1500$ to the forward-path and reverse-path capacity estimates obtained by SProbe, AsymProbe, and TRIO.

Table 4.3 shows the evaluation results in terms of the means and 95% confidence intervals of $\hat{C}_f^{(n)}$ and $\hat{C}_r^{(n)}$. We highlight those results with an absolute difference (computed by $|E[\hat{C}_f^{(n)}] - C_f^{(n)}|/C_f^{(n)}$ or $|E[\hat{C}_r^{(n)}] - C_r^{(n)}|/C_r^{(n)}$) greater than 0.1. The overall results are consistent with the analytical results discussed in the last two sections: (1) TRIO's capacity estimates are accurate for all scenarios, (2) SProbe's capacity estimates are accurate only for forward-path capacity under FR paths, (3) AsymProbe's capacity estimates are accurate, except for high degrees of capacity asymmetry, and (4) DSLprobe's capacity estimates are accurate for some cases

under FF paths.

Table 4.3: Capacity estimates (in Mbits/s) obtained by TRIO, SProbe, AsymProbe, and DSLprobe. The symbol '–' means that the corresponding tool could not output the result.

$C_{f/r}^{(n)}$	TRIO	SProbe
0.512/20	$0.51 \pm 0.00/20.08 \pm 0.02$	$0.51 \pm 0.01/\mathbf{2.41 \pm 0.10}$
0.64/6	$0.64 \pm 0.00/6.01 \pm 0.02$	$0.63 \pm 0.01/\mathbf{0.74 \pm 0.00}$
0.8/8	$0.80 \pm 0.00/8.02 \pm 0.00$	$0.86 \pm 0.08/\mathbf{0.94 \pm 0.06}$
1/18	$0.99 \pm 0.00/17.76 \pm 0.04$	$1.00 \pm 0.04/\mathbf{1.00 \pm 0.04}$
10/30	$10.09 \pm 0.01/30.14 \pm 0.05$	$\mathbf{11.35 \pm 2.50/3.66 \pm 0.10}$
6/0.64	$6.07 \pm 0.01/0.64 \pm 0.00$	$6.50 \pm 1.29/\mathbf{0.08 \pm 0.00}$
8/0.8	$8.08 \pm 0.02/0.80 \pm 0.00$	$\mathbf{11.72 \pm 2.24/0.10 \pm 0.00}$
18/1	$18.38 \pm 0.09/1.00 \pm 0.00$	$\mathbf{20.92 \pm 1.03/0.12 \pm 0.00}$
20/0.512	$20.97 \pm 0.21/0.51 \pm 0.00$	$\mathbf{12.84 \pm 0.02/0.06 \pm 0.00}$
30/10	$30.63 \pm 0.09/10.04 \pm 0.01$	$\mathbf{109.70 \pm 64.54/1.22 \pm 0.03}$
	AsymProbe	DSLprobe
0.512/20	$0.51 \pm 0.01/\mathbf{12.10 \pm 0.17}$	$-/\mathbf{0.65 \pm 0.01}$
0.64/6	$0.63 \pm 0.04/5.95 \pm 0.20$	$-/\mathbf{0.87 \pm 0.01}$
0.8/8	$0.79 \pm 0.01/8.04 \pm 0.06$	$-/\mathbf{1.08 \pm 0.02}$
1/18	$1.01 \pm 0.02/18.32 \pm 0.04$	$-/\mathbf{1.37 \pm 0.02}$
10/30	$10.51 \pm 0.40/31.10 \pm 0.22$	$-/31.91 \pm 6.96$
6/0.64	$5.58 \pm 0.10/0.64 \pm 0.00$	$5.87 \pm 0.07/0.68 \pm 0.00$
8/0.8	$8.24 \pm 0.15/0.80 \pm 0.00$	$7.89 \pm 0.05/0.85 \pm 0.00$
18/1	$18.76 \pm 0.16/1.00 \pm 0.00$	$17.14 \pm 0.23/1.07 \pm 0.00$
20/0.512	$\mathbf{12.56 \pm 0.20}/0.51 \pm 0.00$	$-/0.55 \pm 0.00$
30/10	$31.07 \pm 0.17/10.07 \pm 0.04$	$\mathbf{36.80 \pm 7.37}/10.81 \pm 0.09$

SProbe, AsymProbe, and DSLprobe Based on Table 4.2, SProbe is expected to underestimate the forward-path capacity for the FF path with $C_{f/r}^{(n)} = 20/0.512$ because of the response interference. Furthermore, it is surprising to see that forward-path estimates' accuracy also decreases with $C_f^{(n)}$. This observation indicates that its forward-path measurement is more sensitive to the cross-traffic interference when the forward-path PPD is small. On the other hand, SProbe unexpectedly underestimates the reverse-path capacity for all scenarios. Inspecting the server-side

raw packet traces reveals that the HTTP `GET` request failed to elicit two back-to-back TCP data response packets, which was probably due to the server's limited congestion window.

Since AsymProbe uses the 1-RTP for both forward-path and reverse-path measurement, it is expected to encounter the response interference for the FF path with $C_{f/r}^{(n)} = 20/0.512$ and the probe interference for the FR path with $C_{f/r}^{(n)} = 0.512/20$. For a similar reason, DSLprobe should encounter the response interference for the same FF path. However, notice that DSLprobe does not report the results. A study of the source code [20] reveals that it does not compute the forward-path estimate when the measured forward-path PPD is not 30% greater than the observed reverse-path PPD (i.e., $1500/20 < 40/0.512 \times 1.3$ in this case) with an apparent attempt of avoiding a high degree of capacity asymmetry. Moreover, the reported results for other cases under FR paths are inaccurate, because, as discussed before, DSLprobe is designed for FF-path measurement.

Other configurations for TRIO We repeated the evaluation of TRIO by changing (i) S to 240 bytes, (ii) ρ to 40%, and (iii) the forward-path and reverse-path bottlenecks next to the web server (i.e., emulated by $\mathbb{R}_3$), whereas the other parameter settings remained unchanged. Table 4.4 shows that TRIO still obtains fairly accurate capacity estimates for all scenarios with less than 8% and 3% errors for the forward-path and reverse-path measurements, respectively.

Moreover, Figure 4.7 shows the means and 95% confidence intervals of the capacity estimates obtained by TRIO under two adverse path conditions—

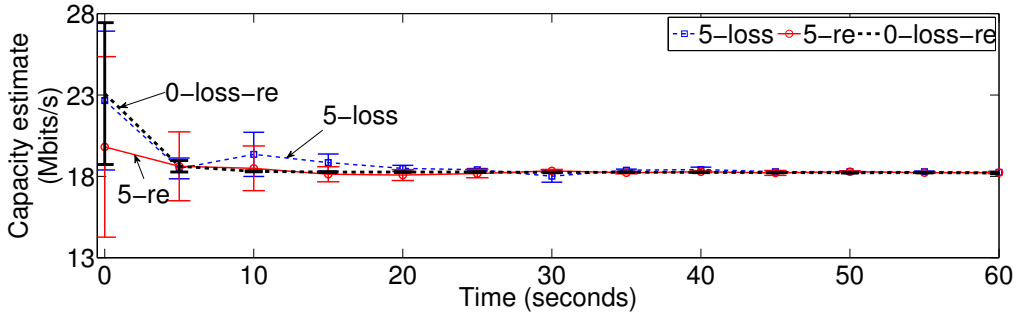
Table 4.4: Capacity estimates (in Mbits/s) obtained by TRIO for three path scenarios.

$C_{f/r}^{(n)}$	TRIO	$C_{f/r}^{(n)}$	TRIO
(i) $S = 240$ bytes, $\rho = 20\%$, $h_f = 3$, and $h_r = 11$			
0.512/20	$0.52 \pm 0.00/20.57 \pm 0.06$	6/0.64	$6.19 \pm 0.03/0.65 \pm 0.00$
0.64/6	$0.65 \pm 0.00/6.11 \pm 0.01$	8/0.8	$8.27 \pm 0.05/0.81 \pm 0.00$
0.8/8	$0.81 \pm 0.00/8.12 \pm 0.01$	18/1	$19.21 \pm 0.22/1.02 \pm 0.00$
1/18	$1.02 \pm 0.00/18.51 \pm 0.06$	20/0.512	$21.59 \pm 0.27/0.52 \pm 0.00$
10/30	$10.34 \pm 0.08/31.00 \pm 0.22$	30/10	$32.44 \pm 0.74/10.19 \pm 0.02$
(ii) $S = 1500$ bytes, $\rho = 40\%$, $h_f = 3$, and $h_r = 11$			
0.512/20	$0.51 \pm 0.00/20.08 \pm 0.01$	6/0.64	$6.07 \pm 0.01/0.64 \pm 0.00$
0.64/6	$0.64 \pm 0.00/6.02 \pm 0.00$	8/0.8	$8.12 \pm 0.02/0.80 \pm 0.00$
0.8/8	$0.80 \pm 0.00/8.02 \pm 0.01$	18/1	$18.55 \pm 0.14/1.00 \pm 0.00$
1/18	$1.00 \pm 0.00/17.99 \pm 0.03$	20/0.512	$21.03 \pm 0.14/0.51 \pm 0.00$
10/30	$10.09 \pm 0.02/30.18 \pm 0.03$	30/10	$30.99 \pm 0.20/10.04 \pm 0.01$
(iii) $S = 1500$ bytes, $\rho = 20\%$, $h_f = 5$, and $h_r = 9$			
0.512/20	$0.51 \pm 0.00/20.18 \pm 0.04$	6/0.64	$6.05 \pm 0.01/0.64 \pm 0.00$
0.64/6	$0.64 \pm 0.00/6.01 \pm 0.02$	8/0.8	$8.06 \pm 0.01/0.80 \pm 0.00$
0.8/8	$0.80 \pm 0.00/8.03 \pm 0.01$	18/1	$18.17 \pm 0.06/1.00 \pm 0.00$
1/18	$1.00 \pm 0.00/18.04 \pm 0.04$	20/0.512	$20.65 \pm 0.13/0.51 \pm 0.00$
10/30	$10.06 \pm 0.01/30.11 \pm 0.19$	30/10	$30.86 \pm 0.17/10.04 \pm 0.01$

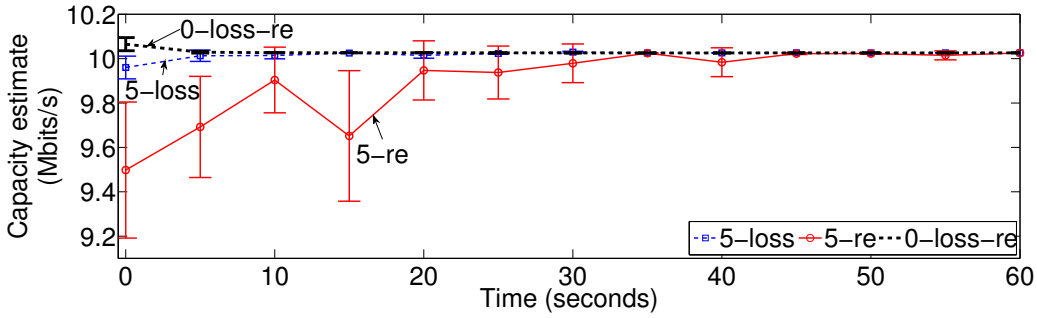
a packet-dropping probability of 5% (labeled by 5-loss) and reordering every the $5^{th}, 10^{th}, 15^{th}, \cdots$ packets (labeled by 5-re)—which were emulated by $\mathbb{R}_1$ for both forward and reverse paths. For comparison purpose, we also emulated a perfect path condition without loss and reordering (labeled by 0-loss-re). For each path condition, we set the degree of capacity asymmetry to $C_{f/r}^{(n)} = 18/1$ (in Mbits/s, emulated by $\mathbb{R}_1$) and ρ to 20%. We ran TRIO to send an interleaved sequence of Poisson-modulated 1-RTPs and $(1,1)$-TWPs with $S = 1500$ bytes and a mean probing rate of 2 Hz for 60 seconds, and repeated the experiment for 50 times.

Figure 4.7 shows that TRIO's estimates converge to the true values with less than 20 seconds under the perfect path condition. Although the

emulated packet loss and reordering affect many probes and RTT samples, TRIO can remove them from the capacity estimation and, as a result, obtain accurate estimates with a slightly longer time (less than 50 seconds).



(a) Forward-path measurement.



(b) Reverse-path measurement.

Figure 4.7: Time series of TRIO's capacity estimates under three path conditions with $C_{f/r}^{(n)} = 18/1$ (in Mbits/s).

4.4.2 Evaluation in the Internet

4.4.2.1 Measurement setup

We also evaluated TRIO and three other tools in Internet paths. The measurement was conducted at an actual ADSL user's home (upstream: 0.6

Mbits/s, downstream: 6 Mbits/s) and at our campus using a Click v1.8 router to emulate three kinds of ADSL links: link 1 (upstream: 0.8 Mbits/s, downstream: 8 Mbits/s), link 2 (upstream: 1 Mbit/s, downstream: 18 Mbits/s), and link 3 (upstream: 0.512 Mbits/s, downstream: 20 Mbits/s), where the home and campus are located in different geographical regions. A measuring system (a laptop) hosting the tools was connected to the actual ADSL link via a 100 Mbits/s Ethernet link. Another measuring system (a workstation) hosting the tools was connected to the Click router via a 100 Mbits/s Ethernet link and equipped with a DAG card to obtain the PPD and RTT samples. Note that the measurement scenarios correspond to FR paths, and we assume that both forward-path and reverse-path bottlenecks are located at the ADSL links.

For each (actual/emulated) ADSL link, we deployed the tools to measure the capacity of the paths to 50 primary and secondary Debian mirror sites reported in [3] on 1 June 2010. These sites were located in 50 different countries; therefore, the paths' characteristics are expected to be very diverse. Since the current implementation of AsymProbe requires the remote endpoint's cooperation, we implemented AsymProbe in HTTP/OneProbe, referred to as AProbe, and used it to evaluate AsymProbe's performance. However, we failed to deploy both SProbe and DSLprobe, because the tools could not trigger valid responses from the mirror sites. Therefore, we could evaluate and compare only TRIO and AsymProbe in this set of evaluation.

Since both TRIO and AProbe are implemented in HTTP/OneProbe, they were configured to send legitimate HTTP requests to fetch the same

web object from each mirror site. Moreover, TRIO sent an interleaved sequence of Poisson-modulated 1-RTPs and (1,1)-TWPs with $S = 1024$ bytes and an average rate of 2 Hz for 180 seconds for each path measurement. For AProbe, we set $S_{max}/S_{min} = 1380/300$ (in bytes) and obtained a sequence of 180 Poisson-modulated 1-RTPs at an average rate of 2 Hz for measuring the capacity in each direction. Moreover, we applied the approach in [70] to account for the layer-two overhead for the real ADSL link's measurement. For TRIO, since each 1024-byte IP packet was carried by 22 ATM cells, each of which had 53 bytes (including a 5-byte header), we scaled up the capacity estimates by a factor of 1.14 (1166/1024). Similarly, we applied the factor of 1.11 (1537/1380) to the capacity estimates obtained by AProbe. By using this packet size configuration, we expect that AProbe will underestimate the reverse-path capacity for all the paths (case (II) in Table 4.2).

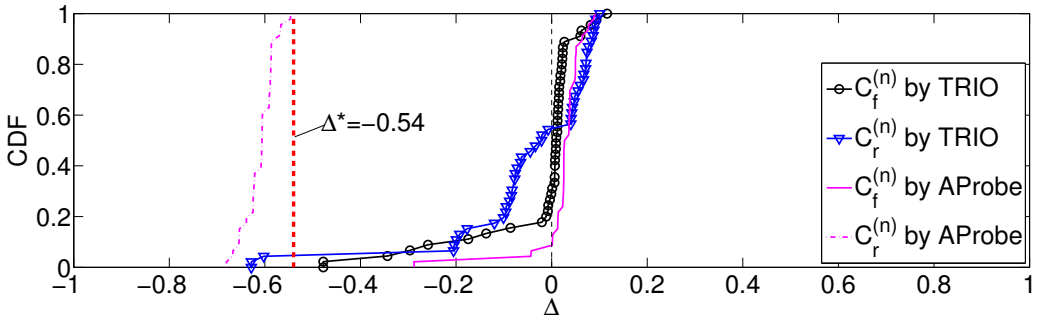
4.4.2.2 Measurement results

Figure 4.8 reports for each ADSL link a CDF for the relative difference $\Delta(\hat{c}, C) = (\hat{c} - C)/C$ (Δ in short) between forward-path (reverse-path) capacity estimates $\hat{c}$ and the actual forward-path (reverse-path) capacity C which is assumed to be the ADSL link's upstream (downstream) bandwidth. The figure shows that TRIO obtains fairly accurate forward-path and reverse-path estimates for all four ADSL links: more than 80% of the capacity estimates obtained by TRIO deviate less than 10% from the actual capacity. On the other hand, AProbe obtains accurate forward-path estimates for all the links, but, as discussed earlier on, it significantly un-

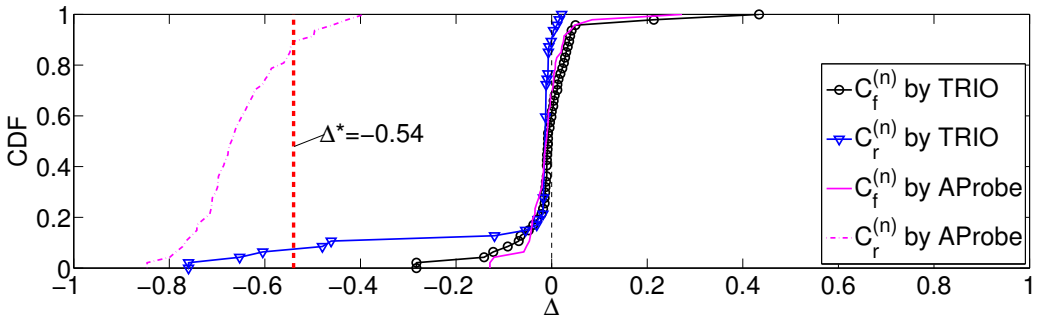
derestimates the reverse-path capacity because of the probe interference. We have also plotted in each figure $\Delta^* = \Delta(\frac{S_{max}}{S_{min}/C_f^{(n)}}, C_r^{(n)})$ which measures the expected difference between the inaccurate measurement of $C_r^{(n)}$ due to the probe interference (i.e., $\delta_{j-1,j}^{(n)} = X^{(h_f)} = S_{min}/C_f^{(n)}$) and $C_r^{(n)}$. As a result, the CDF for its reverse-path estimates is close to Δ^* in each case.

Moreover, we notice some capacity underestimations (i.e., $\Delta(\hat{c}, C) < 0$) from the TRIO and AProbe measurements that could be caused by the actual reverse-path capacity being less than our configured values. Let C_r be the actual reverse-path capacity. If $C_r^{(n)} > C_r > \frac{S_{max}}{S_{min}/C_f^{(n)}}$, both TRIO's and AProbe's estimates will result in a negative Δ. For AProbe, since the PPD is still constrained by the forward path (cases (II) in Table 4.2), Δ for its estimates should be close to Δ^*. However, unlike AProbe, we could observe some intermediate values between zero and Δ^* for the TRIO's estimates, implying that TRIO could still obtain C_r accurately (although we do not have the ground truth to validate our claim). Further, if $C_r^{(n)} > \frac{S_{max}}{S_{min}/C_f^{(n)}} \geq C_r$, the PPD will be constrained by the reverse path (cases (2) or (4) in Table 4.2) and the probe interference will no longer exist. Therefore, AProbe should be able to estimate the reverse-path capacity accurately. This explains why the TRIO's and AProbe's underestimations with $\Delta(\hat{c}, C_r^{(n)}) < \Delta^*$ (i.e., $\hat{c} < \frac{S_{max}}{S_{min}/C_f^{(n)}}$) are clustered together.

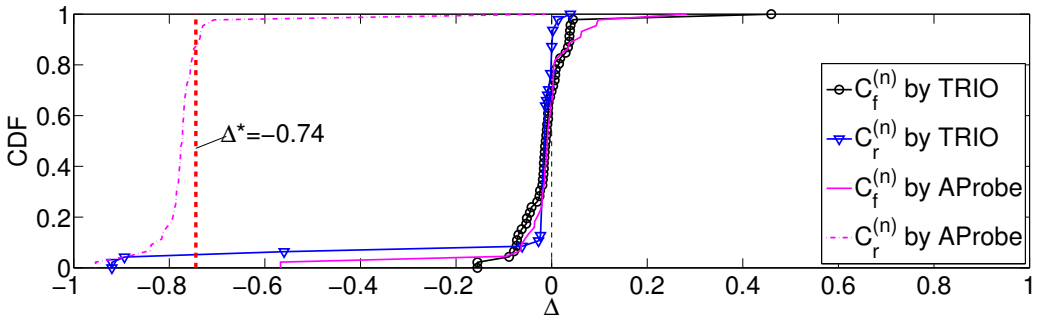
We also repeated the experiments with the mirror web servers for Fedora [5], Gentoo [7], and openSUSE [12]. In particular, we selected 20 servers from each Linux distribution network and repeated the TRIO experiment with an emulated ADSL link (upstream: 0.6 Mbits/s, downstream:



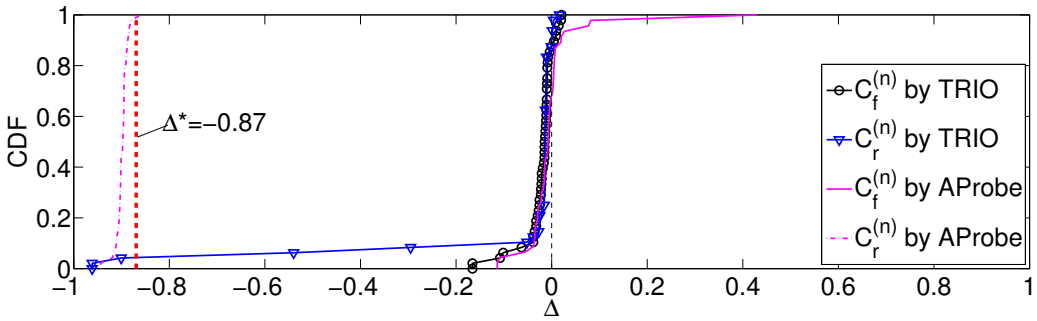
(a) ADSL ($C_{f/r}^{(n)} = 0.6/6$).



(b) Link 1 ($C_{f/r}^{(n)} = 0.8/8$).



(c) Link 2 ($C_{f/r}^{(n)} = 1/18$).



(d) Link 3 ($C_{f/r}^{(n)} = 0.512/20$).

Figure 4.8: CDF of the relative differences between the configured capacity and the capacity estimates.

6 Mbits/s). Our results show that TRIO can still obtain fairly accurate capacity estimates for each direction: more than 79% of the reverse-path capacity estimates are within 80% of the configured capacity, whereas 95% of the forward-path estimates are within 90% of the configured capacity.

4.5 Discussion

Both TRIO and AsymProbe (which also uses minRTTs to filter out invalid PPDs [115]) could take a long time to obtain correct minRTTs from significantly congested network paths. Nonetheless, only TRIO can flexibly manipulate the probe and response packet sizes to improve the measurement accuracy. In particular, TRIO can reduce the packet size to mitigate the cross-traffic impact on the second packet in the packet pair [50, 78, 115]. However, the minimum packet size should be limited according to the time resolution supported by the measuring node. For instance, the node should support microsecond resolution to obtain the PPD (i.e., 3.2 μs) of 40-byte packets introduced by a 100 Mbits/s bottleneck. Increasing the packet size is necessary when significant variance in the capacity estimates is observed [115]. On the other hand, TRIO can also apply the convergence test [79] and the bootstrap method [127] to detect the convergence of minRTTs.

Another issue is that TRIO will produce incorrect capacity estimates in the presence of multichannel bottleneck link. Such problem exists in all path capacity measurement methods (e.g., AsymProbe and SProbe)

based on packet pairs and has been discussed in [78, 175]. In particular, a link with $c > 1$ channels will transmit a pair of probe packets in parallel. As a result, the pair will pass through different channels and they will not queue one after another (which violates our assumption). Moreover, following the discussion in [78], TRIO can only measure the peak rate, but not the sustainable rate (which happens after a certain burst size), of a traffic shaper based on the leaky bucket algorithm [216]. Nonetheless, exploring k-RTP and (v, k)-TWP with $v, k > 1$ (i.e., packet trains) is a viable solution for TRIO to tackle these issues and is a subject for future work.

On the other hand, the overhead introduced by TRIO is small compared to AsymProbe, DSLprobe, and SProbe. To achieve the forward-path and reverse-path measurements, TRIO only dispatches an interleaved sequence of 1-RTPs and $(1, 1)$-TWPs with $S_f = S_r = S$ to measure the four minRTTs, as compared with the three methods that leverage multiple (e.g., three for AsymProbe [58]) probing phases with different S_f and S_r to obtain the required packet dispersions. As shown in Chapter 3, admitting minRTT as the basic unit for capacity measurement also has the speed advantage over the methods based on the packet dispersion technique. Moreover, TRIO can further reduce its overhead by using 0-RTP and $(1, 1)$-TWP (i.e., three probe packets and three response packets), but at the cost of reduced accuracy.

4.6 Summary

We presented TRIO for measuring capacity asymmetry of a network path. The key design choice responsible for its versatility and accuracy is to use three minRTTs for the capacity estimation. Using minRTTs, instead of packet dispersion, eliminates the probe and response interferences suffered by the existing methods. As a result, TRIO is the first method that can measure any degree of capacity asymmetry and under all measurement scenarios. Using the RTTs also has the important advantage of mitigating the cross-traffic interference by filtering biased RTT samples and performing self-validation tests.

To obtain the three minRTTs (and another minRTT for self-validation), we carefully crafted two types of probes: RTP and TWP. We showed that integrating the RTP and TWP enables a simultaneous measurement of the forward-path and reverse-path capacity (and validating the estimates). Previous probing methods either use only one of them or use both separately. We also proved that the three minRTTs are sufficient for deriving the capacity and implemented TRIO using HTTP/OneProbe. Both analytical results and empirical evaluations confirmed TRIO's capability and measurement accuracy.

5

Measurement of Loss Pairs in Network Paths

Packet loss behavior in network paths has been extensively studied for the last twenty years. Besides characterizing the packet loss behavior based on various loss metrics (e.g., loss stationarity [234], loss episodes [209, 234], and loss correlation [161, 231]), it is also useful to study the correlation between packet loss and other important performance metrics. For example, the loss-pair measurement [133] was proposed a decade ago for correlating a packet loss event and the delay that would have been experienced by the lost packet. A packet pair is referred to as a loss pair [132, 133] if exactly one packet (the first or second) in the pair is lost. If

the two packets traverse the path close to each other, then the residual packet's delay can be used to infer the lost packet's delay. The loss-pair measurement has been used to characterize packet dropping behavior [133], classify the type of packet loss [134], and detect dominant congestion links [227] and common congestion points [93, 194].

Despite the unique advantage shared by no other methods, no actual loss-pair measurement from Internet paths has ever been reported. Only ns-2 simulation and emulated testbed experiments were used to evaluate the effectiveness of using loss pairs to discover additional path properties [132, 133, 134]. As a result, the behavior of loss pairs in Internet paths is largely unknown. Moreover, some important delay components, such as the impact of the first packet on the second, have not been taken into consideration. In this chapter, we revisit the loss-pair measurement method [51] and make three main contributions:

1. Delay characterization We conducted a more detailed analysis for the residual packets' delays by including the impacts of cross traffic and the first packet. The new analysis invalidates the previous claim that the first and second residual packets give the same result [132, 133]. We instead show that using the first packet's delay is generally more accurate than the second packet's delay on inferring the congested router's queueing delay upon packet loss. Moreover, we show that the delay variation of the first and second residual packets can be used to estimate the link capacity of a hop preceding the congested router.

2. Method for measuring loss pairs We exploited OneProbe's capability [137] of detecting path events from a single endpoint to measure all four

possible loss pairs on a round-trip path: two for the forward path and the other two for the reverse path. To the best of our knowledge, OneProbe is the first non-cooperative method capable of performing comprehensive loss-pair measurement. Previous loss-pair measurement considered only two possible loss pairs on a round-trip path [132, 133]. We also utilized OneProbe's facility of packet size configuration to validate that a smaller packet size generally increases the accuracy of delay inference.

3. Loss-pair measurement in the Internet We conducted loss-pair measurement using HTTP/OneProbe (an OneProbe implementation based on HTTP/1.1 [87]) for 88 round-trip paths between eight universities in Hong Kong and 11 PlanetLab nodes located at eight countries. Our measurement shows that loss pairs were prevalent in the packet pairs that suffered packet loss, and a loss-pair analysis can help infer additional properties about the lossy paths. Besides, we show that loss pairs' delays provide path signatures for correlating multiple path measurements.

The remainder of this chapter is structured as follows. In Section 5.1, we review the loss-pair measurement method and describe how OneProbe detects the loss-pair events. In Section 5.2, we analyze the residual packets' delays and relate the results to the problem of estimating the queueing delay at the congested router upon packet drop. In Section 5.3, we report our findings of measuring 88 paths continuously for almost three weeks. We discuss the limitations of this work in Section 5.4 and conclude this chapter in Section 5.5.

5.1 Active loss-pair measurement

In loss-pair measurement, a source node sends a sequence of *probe pairs*, each pair consisting of two back-to-back probe packets, to a destination node. The possible delivery statuses of a probe pair are `00` (both received), `01` (only the first is received), `10` (only the second is received), or `11` (none is received). The cases of `01` and `10` are referred to as *loss pairs* in [133]. Moreover, the destination node may be induced to send a sequence of *response pairs*, each pair consisting of two back-to-back response packets, to the source node. There are four similar delivery statuses for each response pair. As a result, there are generally four possible loss pairs for a round-trip path: `P10` and `P01` for a probe pair, and `R10` and `R01` for a response pair.

Both passive and active methods could be used for measuring loss pairs. An active loss-pair measurement of a path can be performed on both endpoints of the path or from only a single endpoint. In this section, we use OneProbe to illustrate how the four types of loss pairs can be measured from only one endpoint. We also deployed HTTP/OneProbe to measure loss pairs on Internet paths, and the results will be presented in Section 5.3.

OneProbe sends a sequence of probe pairs, each consisting of two TCP data packets, to a remote server. If both packets are received in the same order, each packet elicits a response TCP packet, thus returning a response pair. Even if one or more probe packets is lost, at least one response TCP packet will be elicited immediately. Moreover, by prede-

termining the number, types, and order of the response packets elicited under each delivery status (00, 01, 10, or 11) of the probe pair, OneProbe can distinguish the delivery statuses for both probe and response pairs just based on the elicited response packets. Table 5.1 shows two cases. For those marked by '✓', OneProbe can simultaneously detect the probe pair's and response pair's delivery statuses. For those marked by '–', OneProbe can only detect the probe pair's status, because at most one response packet can be elicited for those cases.

Table 5.1: The delivery statuses of probe and response pairs measured by OneProbe.

	R00	R10	R01	R11
P00	✓	✓	✓	✓
P10	✓	✓	✓	✓
P01	–	–	–	–
P11	–	–	–	–

Six cases in Table 5.1 involve at least one loss pair, and they are illustrated in Figure 5.1. For P00 and P10, two response packets can be elicited from the server. As a result, OneProbe can detect the forward-path and reverse-path loss pairs at the same time. However, in the absence of a response pair, OneProbe can detect only the forward-path loss pair for P01. Furthermore, packet reordering does not affect the loss-pair measurement, because OneProbe can also identify from the response packets end-to-end packet ordering events for the probe and response pairs.

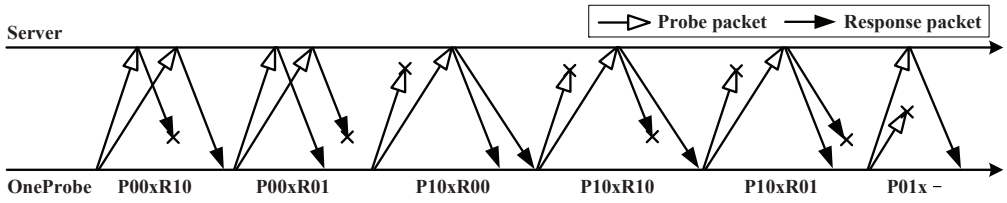


Figure 5.1: The six loss-pair events measured by OneProbe.

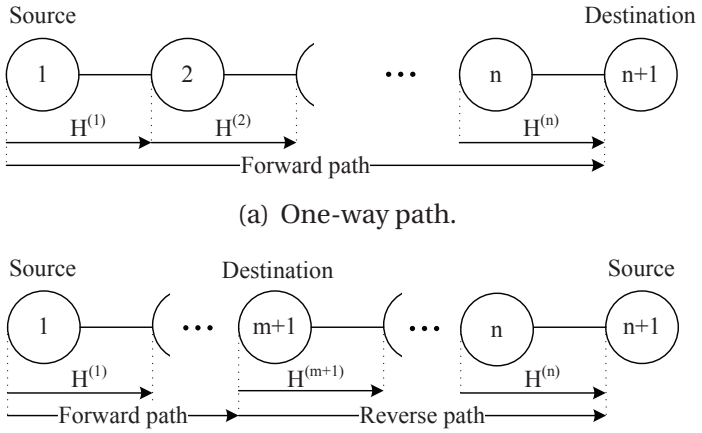
5.2 Analysis of loss pairs' delays

Since a packet pair's delay is used for inferring path properties, in this section we analyze the first and second packets' delay, and their difference. In the following analysis, we consider the four loss-pair events (P10xR00, P01x–, P00xR10, and P00xR01) for which a loss pair exists in only one unidirectional path, and a similar analysis can be performed for the other events. To simplify the notations, we also use LP_{10} to denote a loss pair with the delivery status 10 (i.e., P10xR00 and P00xR10), and LP_{01} to denote that with the status 01 (i.e., P01x– and P00xR01).

After presenting the network models in Section 5.2.1, we first derive in Section 5.2.2 the residual packets' delays in the LP_{10} and LP_{01}, taking into consideration the queueing delay at all hops. In Section 5.2.3, we then extend the analysis to the problem of using the delay to characterize the congested node's queueing delay upon packet drops. Finally in Section 5.2.4, we show that the LP_{10}'s and LP_{01}'s delays can be utilized to estimate the capacity of a link preceding the congested node.

5.2.1 Network models

Consider a sequence of probe pairs dispatched on a network path of n hops (where $n \geq 1$) which also admits other cross traffic. The network path is assumed unchanged throughout the measurement. Each hop in the path consists of a store-and-forward node and its outgoing link connecting to the next hop. We use $H^{(h)}$ to denote the h^{th} hop that transmits (i.e., serializes) packets to the outgoing link with capacity of $C^{(h)}$ bits/s. Each node is configured with a droptail queue which is modeled as a single-server queue with a buffer size of $B^{(h)}$ bits for $H^{(h)}$ and a First-Come-First-Serve (FCFS) queueing discipline. For convenience, we label the hops on the path sequentially, starting from 1 at the source node. The n-hop network path can be either a one-way path (forward path) depicted in Figure 5.2(a) or a round-trip path (forward path and reverse path) in Figure 5.2(b) in which the destination node is located at $H^{(m+1)}$, $1 \leq m < n$.



(a) One-way path.

(b) Round-trip path.

Figure 5.2: Two models for the loss-pair analysis.

We use $\{p_{j-1}, p_j\},\ j = 2i, i = 1, 2, \ldots$, to denote the i^{th} probe pair with p_{j-1} being the first packet in the pair. Each probe packet is of S bits long, including the IP header. Therefore, sending a probe packet on $H^{(h)}$ incurs at least a packet transmission delay of $X^{(h)}$ ($= S/C^{(h)}$) and a constant propagation delay denoted by $T^{(h)}$. Besides, adjacent probe pairs are assumed to be sufficiently spaced out, so that a packet is never queued behind the preceding packet pair, and the probe packets are not out-of-ordered due to the FCFS queueing discipline. In the case of round-trip path, we also make similar assumptions for the response packet pairs. Moreover, we use the same notations and packet size for the response pairs to simplify our ensuing discussion. However, the analysis can be easily adapted to different probe and response packet sizes.

We start the analysis by considering the total delay for p_j to traverse the first h hops of the path, denoted by $d_j^{(h)},\ h = 0, 1, \ldots, n$, and $d_j^{(0)} = 0$. We also let $t_j^{(h)},\ h = 1, \ldots, n+1$, be the time for p_j's to fully arrive (including the last bit of the packet) at $H^{(h)}$. Therefore,

$$\begin{aligned} d_j^{(h)} &= t_j^{(h+1)} - t_j^{(1)}, \\ &= d_j^{(h-1)} + \left(w_j^{(h)} + X^{(h)} + T^{(h)} \right), \end{aligned} \tag{5.1}$$

where $w_j^{(h)}$ is the queueing delay experienced at $H^{(h)}$. The recursive expression in Equation (5.1) also applies to p_{j-1} after updating the subscripts.

Moreover, we can relate $d_j^{(h)}$ and $d_{j-1}^{(h)}$ as

$$\begin{aligned} d_j^{(h)} &= \left(t_j^{(h+1)} - t_{j-1}^{(h+1)}\right) + \left(t_{j-1}^{(h+1)} - t_{j-1}^{(1)}\right), \\ &= \tau_{j-1,j}^{(h+1)} + d_{j-1}^{(h)}. \end{aligned} \tag{5.2}$$

where $\tau_{j-1,j}^{(h+1)}$ is the $\{p_{j-1}, p_j\}$'s inter-arrival time at $H^{(h+1)}$.

5.2.2 Analyzing the residual packets' delays

In the following, we consider a packet in $\{p_{j-1}, p_j\}$ being dropped at $H^{(h')}$ and the other packet delivered successfully. Thus, the loss pair is either an LP_{10} or LP_{01}. We also assume that the packet losses are due to node congestion. We obtain their residual packets' delays by including the queueing delay incurred from each hop. For the LP_{10}, it is also important to include p_{j-1}'s delay on the first $h' - 1$ hops.

5.2.2.1 LP_{10}

To obtain p_j's delay for the LP_{10}, we first apply Equation (5.1) recursively until reaching the $(h'-1)^{th}$ node (since p_{j-1} is discarded at the h'^{th} node):

$$d_j^{(n)} = d_j^{(h'-1)} + \sum_{h=h'}^{n} \left(w_j^{(h)} + X^{(h)} + T^{(h)}\right). \tag{5.3}$$

By using Equation (5.2) for $d_j^{(h'-1)}$ and then applying Equation (5.1) recursively for $d_{j-1}^{(h'-1)}$, we obtain

$$\begin{aligned} d_j^{(n)} &= d_{j-1}^{(h'-1)} + \tau_{j-1,j}^{(h')} + \sum_{h=h'}^{n} \left(w_j^{(h)} + X^{(h)} + T^{(h)} \right), \\ &= \sum_{h=1}^{h'-1} w_{j-1}^{(h)} + \sum_{h=h'}^{n} w_j^{(h)} + \tau_{j-1,j}^{(h')} + \sum_{h=1}^{n} \left(X^{(h)} + T^{(h)} \right). \end{aligned} \tag{5.4}$$

In addition to the queueing delay at all the nodes [133], Equation (5.4) also shows that the residual packet's delay contains $\tau_{j-1,j}^{(h')}$ which, as will be seen shortly, depends on a number of delay components in the preceding hops.

5.2.2.2 LP_{01}

To obtain p_{j-1}'s delay for the LP_{01}, we apply Equation (5.1) recursively for $d_{j-1}^{(h)}$ to obtain

$$d_{j-1}^{(n)} = \sum_{h=1}^{n} w_{j-1}^{(h)} + \sum_{h=1}^{n} \left(X^{(h)} + T^{(h)} \right). \tag{5.5}$$

Since the first packet is the residual packet, its delay is not affected by the second packet and does not contain $\tau_{j-1,j}^{(h')}$.

5.2.2.3 Testbed experiments

We conducted testbed experiments to evaluate the impact of $\tau_{j-1,j}^{(h')}$ on the residual packet's delay. The testbed, shown in Figure 5.3, was configured with a 12-hop round-trip path ($n = 12$), consisting of a probe sender, a web server running Apache v2.2.3 as the destination node, three

cross-traffic clients $\mathbb{X}_1 - \mathbb{X}_3$, and five forwarding devices: two Linux 2.6.26 routers $\mathbb{R}_1 - \mathbb{R}_2$ and three 100 Mbits/s Ethernet switches $\mathbb{S}_1 - \mathbb{S}_3$. We designated $H^{(5)}$ ($\mathbb{R}_2$ and its link to $\mathbb{S}_3$) to be the only congested node on the path (i.e., $h' = 5$). We achieved this by running TC/Netem [97] in $\mathbb{R}_2$ to emulate $C^{(5)} = 50$ Mbits/s and a FCFS queue to accommodate approximately 100 ms of packets, and generating forward-path cross traffic (from $\mathbb{X}_2$ to $\mathbb{X}_3$) to congest $H^{(5)}$. Moreover, we designated $H^{(3)}$ ($\mathbb{R}_1$ and its link to $\mathbb{S}_2$) to be a bottleneck link by configuring Click v1.8 [123] in $\mathbb{R}_1$ to emulate $C^{(3)} = 1$ Mbit/s. The Click router was also configured to set the RTT between the probe sender and web server to 200 ms.

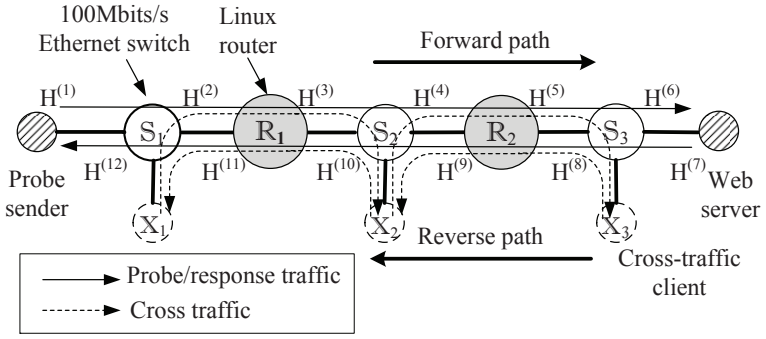
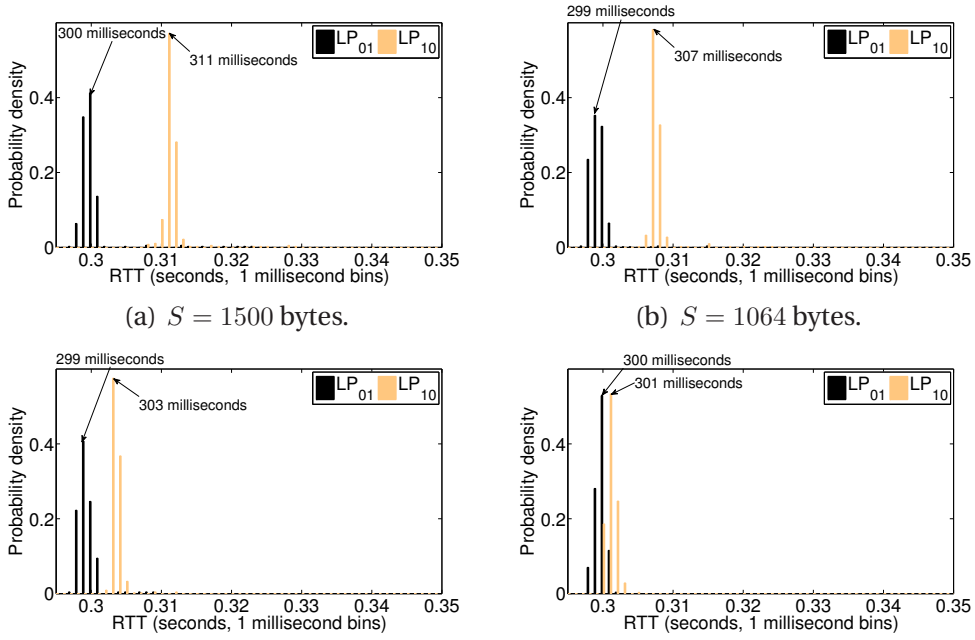


Figure 5.3: The testbed for the loss-pair experiments.

For this set of experiments, except for $H^{(5)}$, we did not generate cross traffic for other hops (i.e., $w_j^{(h)} = 0,\ \forall h \neq h'$, in Equations (5.4) and (5.5)). We ran HTTP/OneProbe from the probe sender to dispatch a sequence of 5000 Poisson-modulated probe pairs with a mean probing rate of 5 Hz. The probe sender was equipped with a DAG 4.5 passive network monitoring card [4] to obtain the RTT samples in microsecond resolution which is limited by the `pcap` header structure [18]. Similar to [115], the cross-traffic sources entered Pareto-distributed `ON` and `OFF` states with a shape

$\alpha = 1.9$ and had a fixed packet size of 1500 bytes.

Figure 5.4(a) shows the distributions of the residual packets' delays for the LP_{10} (i.e., P10xR00) and LP_{01} (i.e., P01x–) with $S = 1500$ bytes. Similar to [133], we applied a small bin size of 1 ms to mitigate the noise introduced by the non-congested hops. The figure shows that the residual packets' delays are dominated by the congested node's queueing delay of 100 ms, because most of them center around 300 ms and 311 ms for the LP_{01} and LP_{10}, respectively. We also note that many delay samples for the LP_{10} include an additional quantity of 11 ms which, according to Equation (5.4), came from $\tau_{j-1,j}^{(h')}$. Unlike other noises, this quantity cannot be filtered out by choosing a small bin size.



(a) $S = 1500$ bytes. (b) $S = 1064$ bytes.

(c) $S = 576$ bytes. (d) $S = 240$ bytes.

Figure 5.4: Residual packets' delays for LP_{10} and LP_{01} with $S = \{1500, 1064, 576, 240\}$ bytes on the testbed for which $C^{(3)} = 1$ Mbit/s, $C^{(h')} = 50$ Mbits/s, and $h' = 5$.

Moreover, we repeated the experiments with $S = \{1064, 576, 240\}$ bytes and their results are shown in Figures 5.4(b)-5.4(d). As the figures reveal, when the probe packet size decreases, the additional quantity observed from the LP_{10}'s delay samples also decreases.

5.2.3 Characterizing the congested node's state

5.2.3.1 LP_{10}

A packet is dropped at $H^{(h')}$ when the node's buffer is full after the instantaneous input traffic rate exceeds $C^{(h')}$ for some time. We let $\{Q^{(h)}(t),\ t \geq 0\}$ be the continuous-time process of its queue length in terms of bits, and $\mathbb{Q}_j^{(h)} = Q^{(h)}(t_j^{(h)-})$ (i.e., the queue length just prior to the arrival of p_j). When $\{p_{j-1}, p_j\}$ is an LP_{10}, we have

$$\mathbb{Q}_{j-1}^{(h')} + S \ > \ B^{(h')}, \text{ and} \tag{5.6}$$

$$\mathbb{Q}_j^{(h')} + S \ \leq \ B^{(h')}, \tag{5.7}$$

where

$$\mathbb{Q}_j^{(h')} = \left(\mathbb{Q}_{j-1}^{(h')} + A_{j-1,j}^{(h')} - D_{j-1,j}^{(h')}\right)^+, \tag{5.8}$$

and $(x)^+ = \max\{0, x\}$. $A_{j-1,j}^{(h')}$ is the amount of packets (in bits) arriving to and buffered at the queue during $(t_{j-1}^{(h')}, t_j^{(h')})$, and $D_{j-1,j}^{(h')}$ is the total amount of packets (in bits) departed from the node during $[t_{j-1}^{(h')}, t_j^{(h')})$. Therefore, the queueing delay of p_j at $H^{(h')}$ can be expressed as

$$w_j^{(h')} \ = \ \frac{\mathbb{Q}_j^{(h')}}{C^{(h')}} + R_j^{(h')}, \tag{5.9}$$

where $R_j^{(h')}$ is the residual service time upon p_j's arrival.

Moreover, $\tau_{j-1,j}^{(h')},\ h' > 1$, can be expressed as [164]:

$$\tau_{j-1,j}^{(h')} = X^{(h^*)} + q_{j-1,j}^{(h^*)} + \sum_{h=h^*+1}^{h'-1} \left(w_j^{(h)} - w_{j-1}^{(h)} \right), \tag{5.10}$$

where $H^{(h^*)},\ 1 \leq h^* \leq h' - 1$, is the last hop preceding $H^{(h')}$ for which p_j arrives before p_{j-1}'s full departure from the node. That is, both belong to the same busy period of the queue at $H^{(h^*)}$ [164]. Moreover, $q_{j-1,j}^{(h^*)}$ is p_j's queueing delay at $H^{(h^*)}$ due to intervening cross traffic arriving between p_{j-1} and p_j, and $X^{(h^*)}$ is the time for transmitting p_j at $H^{(h^*)}$.

For the purpose of estimating $\mathbb{Q}_j^{(h')}/C^{(h')}$, it is useful to consider p_j's *path queueing delay* defined by $\Theta_j = d_j^{(n)} - \min_{\forall i,\ j=2i}\{d_{j-1}^{(n)}\}$. Assuming that the minimum observable delay of $p_{j-1},\ j = 2i,\ i = 1, 2, \ldots,$ precludes the cross-traffic-induced queueing delay and using Equations (5.4), (5.9), and (5.10), we have

$$\begin{aligned} \Theta_j &= d_j^{(n)} - \sum_{h=1}^{n} \left(X^{(h)} + T^{(h)} \right), \\ &= \frac{\mathbb{Q}_j^{(h')}}{C^{(h')}} + R_j^{(h')} + X^{(h^*)} + \zeta_j, \end{aligned} \tag{5.11}$$

where $\zeta_j (= \sum_{h=1}^{h^*} w_{j-1}^{(h)} + q_{j-1,j}^{(h^*)} + \sum_{h=h^*+1}^{h'-1} w_j^{(h)} + \sum_{h=h'+1}^{n} w_j^{(h)})$ is the queueing delay contributed by the cross traffic present at $H^{(h')}$'s upstream and downstream hops.

From Equation (5.11), Θ_j can be used to estimate $\mathbb{Q}_j^{(h')}/C^{(h')}$, and the estimation is biased by the residual service time, $X^{(h^*)}$, and cross traf-

fic. Furthermore, $\mathbb{Q}_j^{(h')}/C^{(h')}$ is a good approximation for $\mathbb{Q}_{j-1}^{(h')}/C^{(h')}$ under certain conditions. For instance, when $\tau_{j-1,j}^{(h')}$ is small enough and Equation (5.7) still holds, $\mathbb{Q}_j^{(h')}$ is expected to be very close to $\mathbb{Q}_{j-1}^{(h')}$, thus making $\mathbb{Q}_j^{(h')}$ a tight lower bound for $B^{(h')} - S$. As a result, if $\mathbb{Q}_j^{(h')}/C^{(h')} \gg R_j^{(h')} + X^{(h^*)} + \zeta_j$ and $B^{(h')} \gg S$, then $\Theta_j \approx B^{(h')}/C^{(h')}$ which was first given in [133]. On the other hand, according to Equation (5.8), $\mathbb{Q}_j^{(h')}$ could be dampened when $A_{j-1,j}^{(h')} \ll D_{j-1,j}^{(h')}$ and $\tau_{j-1,j}^{(h')}$ becomes large, because the congestion may be relieved by the time p_j arrives.

5.2.3.2 $\mathbf{LP}_{01}$

The analysis for the LP_{01} is similar to the above. When $\{p_{j-1}, p_j\}$ is an LP_{01}, we have

$$\mathbb{Q}_{j-1}^{(h')} + S \;\leq\; B^{(h')}, \text{ and} \tag{5.12}$$

$$\mathbb{Q}_j^{(h')} + S \;>\; B^{(h')}, \tag{5.13}$$

where

$$\mathbb{Q}_j^{(h')} = \left(\mathbb{Q}_{j-1}^{(h')} + S + A_{j-1,j}^{(h')} - D_{j-1,j}^{(h')}\right)^+. \tag{5.14}$$

By replacing $w_{j-1}^{(h')}$ with a similar expression as Equation (5.9), we obtain p_{j-1}'s path queueing delay, defined by $\Theta_{j-1} = d_{j-1}^{(n)} - \min_{\forall i,\, j=2i}\{d_{j-1}^{(n)}\}$:

$$\Theta_{j-1} = \frac{\mathbb{Q}_{j-1}^{(h')}}{C^{(h')}} + R_{j-1}^{(h')} + \zeta_{j-1}, \tag{5.15}$$

where $\zeta_{j-1} = \sum_{h=1}^{h'-1} w_{j-1}^{(h)} + \sum_{h=h'+1}^{n} w_{j-1}^{(h)}$ and $R_{j-1}^{(h')}$ is the residual service time upon p_{j-1}'s arrival. Unlike the LP_{10}, the LP_{01}'s path queueing delay

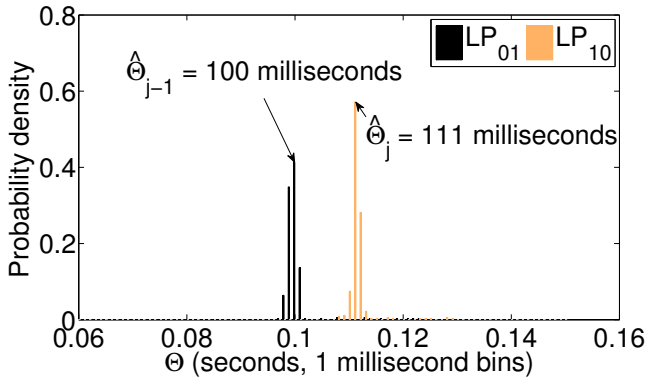
does not contain $X^{(h^*)}$, and ζ_{j-1} contains fewer components.

To estimate $B^{(h')}/C^{(h')}$ by the LP_{01}, note that $\mathbb{Q}_{j-1}^{(h')}$ serves as a tight lower bound for $\mathbb{Q}_j^{(h')}$ if $A_{j-1,j}^{(h')}$ is close to $D_{j-1,j}^{(h')}$ or $\tau_{j-1,j}^{(h')}$ is small enough. If p_{j-1} arrives at $H^{(h')}$ with its queue length not close to $B^{(h')}$ and $\tau_{j-1,j}^{(h')}$ is small, then Equation (5.13) will not hold with a high probability. However, if p_{j-1} arrives at an almost full queue and $\tau_{j-1,j}^{(h')}$ is small, it is more likely that Equation (5.13) will hold. As a result, if $\mathbb{Q}_{j-1}^{(h')}/C^{(h')} \gg R_{j-1}^{(h')}+\zeta_{j-1}$, $\tau_{j-1,j}^{(h')}$ is small enough, and $B^{(h')} \gg S$, then $\Theta_{j-1} \approx B^{(h')}/C^{(h')}$.

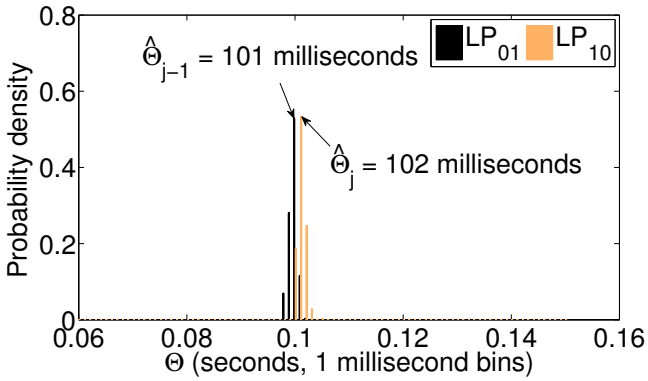
5.2.3.3 Testbed results

Figure 5.5(a) plots the path queueing delays for the LP_{10} and LP_{01} with $S = 1500$ bytes which are obtained from the previous set of testbed experiments. Notice that $B^{(h')}/C^{(h')}$ ($= 100$ ms) is much greater than $S/C^{(h')}$ ($= 240$ μs). We denote the bin with the highest count for the LP_{10} as $\hat{\Theta}_j$ and that for the LP_{01} as $\hat{\Theta}_{j-1}$. As shown, $\hat{\Theta}_{j-1}$ is the same as $B^{(h')}/C^{(h')}$. However, $\hat{\Theta}_j$ deviates from $B^{(h')}/C^{(h')}$ by about 11 ms, which is close to $X^{(h^*)}$ ($= 1500$ bytes$/1$ Mbit/s $= 12$ ms). Therefore, the results validate the contribution of $X^{(h^*)}$ to Θ_j, as modeled in Equation (5.11).

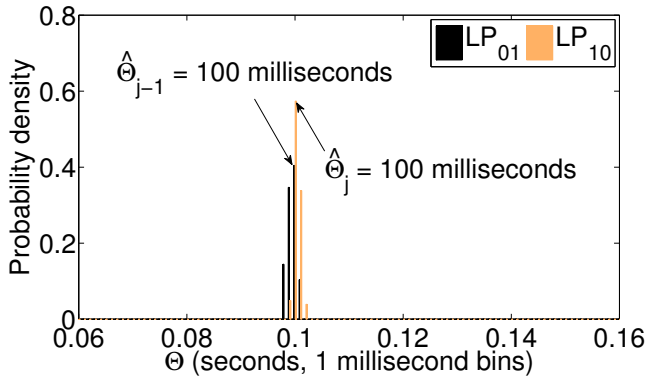
We also repeated the experiments by using a small probe packet size $S = 240$ bytes (with the same bottleneck link capacity $C^{(h^*)} = 1$ Mbit/s) and a larger bottleneck link capacity $C^{(h^*)} = 10$ Mbits/s (with the same probe packet size $S = 1500$ bytes), where $B^{(h')}/C^{(h')} \gg S/C^{(h')}$ for both cases. As shown in Figures 5.5(b) and 5.5(c), $\hat{\Theta}_{j-1}$ remains very close to $B^{(h')}/C^{(h')}$. Although $\hat{\Theta}_j$ may still deviate from $B^{(h')}/C^{(h')}$, the degree of the deviation becomes smaller, because of the decrease in $X^{(h^*)}$.



(a) $S = 1500$ bytes, $C^{(h^*)} = 1$ Mbit/s.



(b) $S = 240$ bytes, $C^{(h^*)} = 1$ Mbit/s.



(c) $S = 1500$ bytes, $C^{(h^*)} = 10$ Mbits/s.

Figure 5.5: Path queueing delays for the LP_{10} and LP_{01} on the testbed for which $C^{(h^*)} = \{1, 10\}$ Mbits/s, $C^{(h')} = 50$ Mbits/s, $h^* = 3$, and $h' = 5$.

The estimates of $B^{(h')}/C^{(h')}$ made by the LP_{10} and LP_{01} are both prone to queueing delay at the non-congested nodes. However, the effect on the LP_{10}'s estimate is generally more significant than the LP_{01}'s, because the LP_{10}'s delay always contains $X^{(h^*)}$ which cannot be eliminated. Note that $X^{(h^*)}$ could be significant if the measurement is conducted using a low-bandwidth residential link. Though the impact of $X^{(h^*)}$ can be mitigated by choosing a smaller packet size for active loss-pair measurement, this is not feasible for passive loss-pair measurement. Whenever the packet size is not configurable, the LP_{01} should be used to avoid the bias.

5.2.4 Estimating $H^{(h^*)}$'s link capacity

In this section, we show that another benefit of the loss-pair analysis is estimating $H^{(h^*)}$'s link capacity from both LP_{10}'s delay and LP_{01}'s delay, assuming that both LP_{10} and LP_{01} observe the same congested hop $H^{(h')}$. Subtracting Equation (5.15) from Equation (5.11) gives

$$\begin{aligned} \Delta_{j-1,j} &= \Theta_j - \Theta_{j-1}, \\ &= X^{(h^*)} + \epsilon, \end{aligned} \tag{5.16}$$

where $\epsilon = \frac{\mathbb{Q}_j^{(h')}}{C^{(h')}} - \frac{\mathbb{Q}_{j-1}^{(h')}}{C^{(h')}} + \zeta_j - \zeta_{j-1} + R_j^{(h')} - R_{j-1}^{(h')}$. Equation (5.16) shows that $\Delta_{j-1,j}$ includes a signature for $X^{(h^*)}$ and a noise term ϵ. Since the queueing delay and residual service times of p_{j-1} and p_j are contributed from different busy periods of the nodes, ϵ can be reasonably regarded as a random noise.

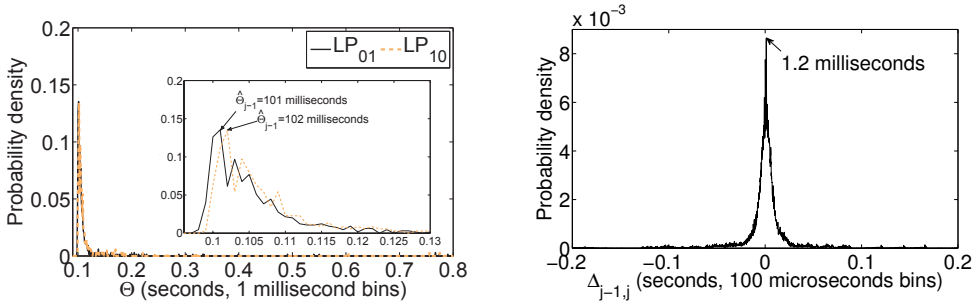
5.2.4.1 Testbed results

We conducted a new set of testbed experiments to evaluate this capability by configuring $\mathbb{R}_1$ to emulate $C^{(3)} = C^{(11)} = 10$ Mbits/s, and keeping $C^{(5)} = 50$ Mbits/s and $B^{(5)}/C^{(5)} = 100$ ms unchanged. Besides $H^{(5)}$, we also introduced the Pareto `On/Off` cross traffic between $\mathbb{X}_1$ and $\mathbb{X}_2$ in the forward and reverse paths. Other configuration settings were unchanged. As a result, $h^* = 3$ and $h' = 5$. We obtain the distribution of $\Delta_{j-1,j}$ by a mutual subtraction between Θ_{j-1} and Θ_j measured from `P01x`– and `P10xR00`, respectively, with $S = 1500$ bytes.

As shown in Figure 5.6(a), although $\hat{\Theta}_{j-1}$ and $\hat{\Theta}_j$ are relatively close to $B^{(h')}/C^{(h')} = 100$ ms, they also experience a higher variation due to the more significant cross traffic throughout the round-trip path. On the other hand, the probability density distribution of $\Delta_{j-1,j}$, shown in Figure 5.6(b), is symmetric about the peak at around 1.2 ms, which corresponds to the transmission delay of $H^{(h^*)}$ (i.e., $X^{(h^*)} = 1500$ bytes$/10$ Mbits/s). Thus, the peak of the distribution, together with the packet size, gives an accurate estimation of $H^{(h^*)}$'s link capacity. We also note from other testbed results (which are not shown in the paper) that $\Delta_{j-1,j}$ diminishes with the $H^{(h^*)}$'s link capacity and increases with the probe packet size. For example, for $S = 40$ bytes, we expect to use a microsecond bin size to make the transmission delay stand out in the distribution of $\Delta_{j-1,j}$.

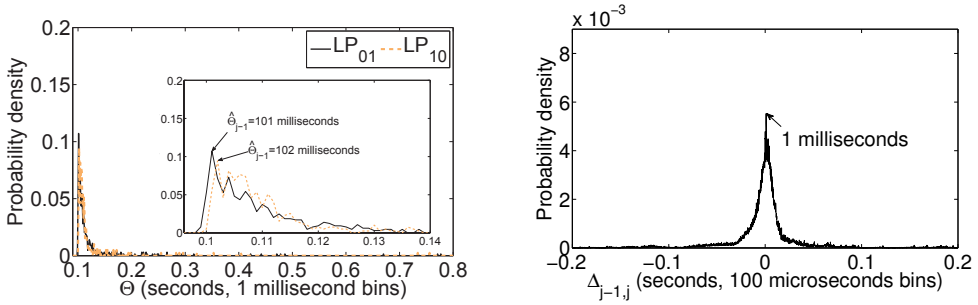
We also include the results for the reverse-path loss pairs based on `P00xR10` and `P00xR01` in Figures 5.6(c)-5.6(d), and they are obtained by configuring $C^{(9)} = 50$ Mbits/s and $B^{(9)}/C^{(9)} = 100$ ms in the same test-

bed, and restoring the link capacity of $H^{(5)}$ to 100 Mbits/s with unlimited buffer. As a result, $h^* = 3$ remains unchanged, but $h' = 9$. Figure 5.6(c) shows that the corresponding $\hat{\Theta}_{j-1}$ and $\hat{\Theta}_j$ are still relatively close to $B^{(h')}/C^{(h')}$ and experience a similar variation due to the significant cross traffic introduced by $\mathbb{X}_1$ and $\mathbb{X}_2$. Moreover, as shown in Figure 5.6(d), $\Delta_{j-1,j}$ is quite similar to the forward-path results.



(a) Θ_{j-1} and Θ_j for $h' = 5$.

(b) $\Delta_{j-1,j}$ for $h' = 5$.



(c) Θ_{j-1} and Θ_j for $h' = 9$.

(d) $\Delta_{j-1,j}$ for $h' = 9$.

Figure 5.6: Path queueing delays and their differences for LP_{10} and LP_{01} with $S = 1500$ bytes on the testbed for which $C^{(h^*)} = 10$ Mbits/s, $C^{(h')} = 50$ Mbits/s, $h^* = 3$, and $h' = \{5, 9\}$.

5.3 Loss pairs in the Internet

We conducted end-to-end Internet path measurement using HTTP/OneProbe between 26 February 2010 20:00 UTC and 17 March 2010 09:00 UTC, inclusively. The measurement covered a total of 112 ($= 8 \times 14$) network paths between eight local universities in Hong Kong, denoted by UA–UH, as the sources of the paths and the 14 PlanetLab nodes listed in Table 5.2 as the destinations. Since HTTP/OneProbe performs measurement in a legitimate web session, we installed a `mini_httpd` (a web server) [181] at each PlanetLab node.

Table 5.2: PlanetLab nodes used for the Internet path measurements.

Aliases	IP addresses	Locations	Average RTTs
PL001	212.235.18.114	Israel	308.24 ms
PL002	216.48.80.14	Canada	244.31 ms
PL003	202.112.28.98	China	83.67 ms
PL004	131.179.50.70	United States	–
PL005	128.143.6.134	United States	–
PL006	165.91.83.23	United States	229.55 ms
PL007	132.72.23.10	Israel	358.54 ms
PL008	210.123.39.168	Korea	53.34 ms
PL009	140.123.230.248	Taiwan	50.54 ms
PL010	134.151.255.181	UK	273.62 ms
PL011	142.104.21.241	Canada	248.99 ms
PL012	194.117.20.214	Portugal	–
PL013	198.82.160.239	United States	237.59 ms
PL014	137.132.80.110	Singapore	38.30 ms

To monitor the path measurement from multiple sources to multiple destinations, we deployed a management system to dispatch the measurement tasks to the measurement nodes, monitor the resource usages in the nodes, and retrieve measurement data from the nodes. Each measurement node executed the measurement tasks to measure the network

paths to the 14 destinations. To avoid self-induced network congestion, the destinations were evenly divided into two groups. The sources performed concurrent measurement for the paths in a group for one minute. Specifically, the sources launched HTTP/OneProbe to dispatch a sequence of Poisson-modulated probe pairs with a mean rate of 5 Hz and $S = 576$ bytes to each destination. To augment the path measurement with route information, tcptraceroute [219] was performed at both the sources and destinations. At the end of the minute, the nodes switched to the other group and repeated the same process. As a result, the average measurement traffic generated by each source was less than 48 KB/s (and less than 7 KB/s for each destination).

The measurement was conducted in the period during which HARNET [25]—a network through which the eight universities peered with one another—changed the service provider. In the switch-over process, the eight universities' networks were first switched to a temporary network one by one between 24 February 2010 14:00 UTC and 27 February 2010 23:00 UTC. They were then migrated back to the new service provider's network between 5 March 2010 11:00 UTC and 7 March 2010 2:00 UTC. As a result of these changes, we observed diverse network path characteristics even for the same source-destination pair.

In the stage of pre-processing the measurement data, we identified and removed a number of measurement artifacts. In particular, we identified artifacts associated with each source by correlating its measurement results. If there is a consistent pattern, such as persistent packet reordering, appearing in all the results, we conclude that the pattern is

originated from the source or the path segment close to the source. This diverse-path-correlation method reveals the following measurement artifacts:

1. Forward-path and reverse-path reordering for UF between 27 February 2010 and 05 March 2010, and

2. Forward-path and reverse-path reordering for UH during the entire period.

Besides the artifacts, we observed system failures in three PlanetLab nodes PL004, PL005, and PL012 during the measurement period. After eliminating the paths to these destinations, we analyzed the remaining 88 paths for general packet loss statistics and loss-pair measurement.

5.3.1 Packet loss behavior: An overview

In this section, we present the overview results on the packet loss behavior, particularly the loss pairs, observed from the network paths. We consider the forward and reverse paths separately, because HTTP/OneProbe can distinguish the two paths for loss measurement. For the forward path, we define a *loss frequency* $f_{FL} = \sum_{i=1}^{M} \mathbf{1}_{\{L_i>0\}}/M$, where L_i represents the delivery status (i.e., `00`, `01`, `10`, or `11`) of the i^{th} probe pair, $\mathbf{1}$ is the indicator function, and M is the total number of packet pairs dispatched during a given time period. $L_i > 0$ if there is at least a packet loss in the pair, and $L_i = 0$, otherwise. For the reverse path, we apply a similar procedure to compute the loss frequency denoted by f_{RL}.

5.3.1.1 Prevalence of packet losses and loss pairs

We summarize in Table 5.3 the packet-loss and loss-pair statistics measured from all the paths for the entire measurement period. The table is organized based on the destinations. That is, the statistics for each destination is computed based on an aggregation of the path measurement from the eight sources to the destination. The statistics are also separated into forward and reverse paths. Besides the f_{FL} and f_{RL}, we also report $f_{\mathtt{P10}}$, $f_{\mathtt{P01}}$, and $f_{\mathtt{P11}}$ which give the respective percentages of P10, P01, and P11 in the set of lossy probe pairs. The columns for $f_{\mathtt{R10}}$, $f_{\mathtt{R01}}$, and $f_{\mathtt{R11}}$ give similar statistics for the reverse path.

Table 5.3: Packet loss and loss pair statistics (in %) grouped by destinations.

Destinations	Forward Paths				Reverse Paths			
	f_{FL}	$f_{\mathtt{P10}}$	$f_{\mathtt{P01}}$	$f_{\mathtt{P11}}$	f_{RL}	$f_{\mathtt{R10}}$	$f_{\mathtt{R01}}$	$f_{\mathtt{R11}}$
PL001	0.04	29.01	33.02	37.97	0.13	24.19	33.60	42.21
PL002	0.16	13.09	57.52	29.39	0.35	19.41	42.53	38.06
PL003	0.23	47.72	51.03	1.25	2.22	41.73	41.76	16.51
PL006	0.01	20.56	43.89	35.55	0.10	30.84	38.10	31.06
PL007	0.07	23.08	25.73	51.19	0.15	25.86	33.86	40.28
PL008	0.67	15.25	34.80	49.95	0.33	25.56	31.26	43.18
PL009	0.29	44.41	44.97	10.62	0.69	44.50	45.80	9.71
PL010	0.01	21.72	36.55	41.73	0.16	29.03	37.38	33.59
PL011	0.17	44.75	49.62	5.62	0.09	36.05	41.61	22.34
PL013	0.04	33.15	40.08	26.77	0.11	29.06	36.78	34.16
PL014	0.93	44.46	47.06	8.49	1.93	47.38	48.02	4.59

We observe the following results from Table 5.3:

1. Both f_{PL} and f_{RL} were less than 1% for most of the paths, and the highest loss frequency was 2.2% (i.e., PL003's f_{RL}).

2. Except for PL008 and PL011, the reverse paths suffered from more

severe packet loss than the corresponding forward paths according to the loss frequencies. This result, however, is most likely location dependent.

3. Loss pairs were prevalent in the lossy packet pairs, because f_{P11} and f_{R11} were generally below 50% (except for PL007's f_{P11}). The frequencies for some of the paths were even below 10%.

4. The LP_{01} dominated the LP_{10} in both forward paths and reverse paths, because f_{P01} (f_{R01}) was consistently higher than f_{P10} (f_{R10}).

5.3.1.2 Time series for packet loss events

To analyze the packet loss statistics as a function of time, we divide the entire measurement period into one-hour bins for each path. Each bin's value is set to 1 if there exists at least a one-minute session with loss frequency greater than 1%; otherwise, the bin value is set to 0. As a result, we obtain a time series of bin values for each path. We can combine the eight sources' time series for a given destination by adding their bin values. Alternatively, we can combine the 11 destinations' time series for a given source by also adding their bin values.

To effectively visualize the time series, we resort to heat-map diagrams. Figures 5.7(a) and 5.7(b) show the heat-map time series for the packet loss events in the forward paths grouped by the sources and destinations, respectively. Since there are 11 paths per source, the possible values in Figure 5.7(a) are $0, 1, \ldots 11$. A darker color corresponds to a higher value. We also grey out all the bins with no measurement data. Similarly, the

possible values in Figure 5.7(b) are $0, 1, \ldots 8$. Moreover, there are three vertical dash lines: the first indicates the completion time for the transition to the temporary network, the second the beginning of the transition to the new service provider, and the third the completion time for the transition to the new service provider. We also show the diagrams for the reverse paths in Figures 5.7(c) and 5.7(d).

The heat-map diagrams enable us to effectively evaluate the loss behavior in the spatial and temporal domains:

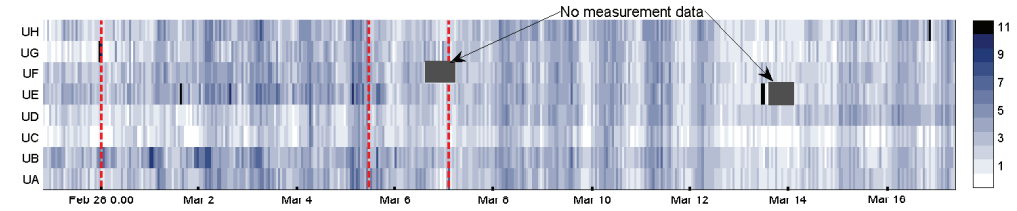
1. (Loss patterns) The heat maps can quickly identify loss patterns for a set of paths. In our case, the set of paths share either the same source or the same destination for forward/reverse paths. Figure 5.7(a) shows that there is no clear loss pattern for all the eight sets of source-identical forward paths. However, Figure 5.7(c) shows intense loss for some of the reverse paths during the network transition (between the first two dotted lines). On the other hand, Figures 5.7(b) and 5.7(d) depict that the loss is much more prevalent for some destination-identical paths.

2. (Loss correlations) The heat maps also reveal strong correlation among different sets of source/destination-identical paths. The most notable one is the periodic, intense losses for the UE, UF, UG, and UH paths in Figure 5.7(c). The similar pattern suggests that they probably shared the same loss origins. Moreover, Figure 5.7(d) shows that all 11 sets of destination-identical paths share similar reverse-path loss patterns during the network transition, but they are no longer

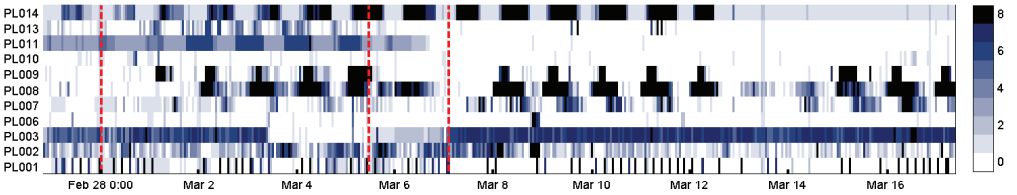
similar after migrating to the new service provider's network.

3. (Loss diagnosis) We use the heat maps to further diagnose the loss behavior by correlating the source-identical paths and destination-identical paths. Going back to the intense losses for the UE, UF, UG, and UH paths in Figure 5.7(c), we can obtain more insights by comparing Figure 5.7(c) and Figure 5.7(d) in the same five periods of heavy losses. Figure 5.7(d) shows that some destinations contributed losses to most of the reverse paths (notably PL014). Therefore, the losses for the four paths actually occurred on multiple locations: some on the destination side and others on the source side.

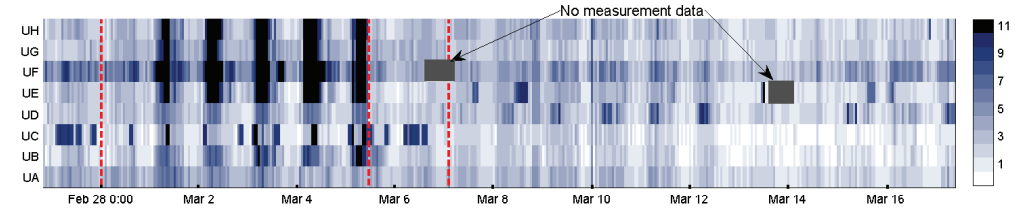
4. (Loss anomalies) The heat maps also help reveal loss anomalies. A time-correlation of the forward-path and reverse-path measurements based on the destinations shows that PL014 is a "congested" node. The paths to and from this node experienced high loss for all paths in a diurnal pattern until 13 March 2010. The loss could occur as a result of congestion at the node or the node's network. Since this path's loss measurement is heavily biased by the destination, a more useful path measurement can be obtained by replacing this with another node in the same vicinity. The forward paths to PL008 and PL009 also experienced periodic high losses. Unlike PL014, the loss patterns continued to the end of the measurement period.



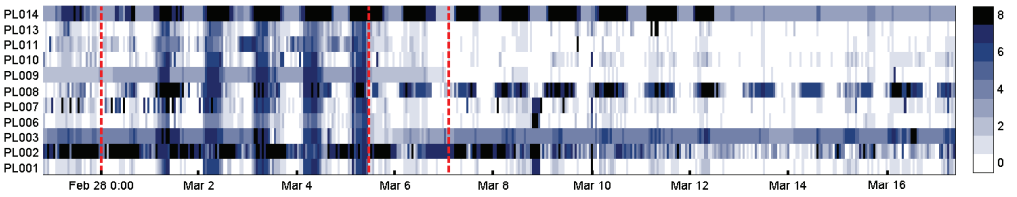
(a) Forward paths (grouped by sources).



(b) Forward paths (grouped by destinations).



(c) Reverse paths (grouped by sources).



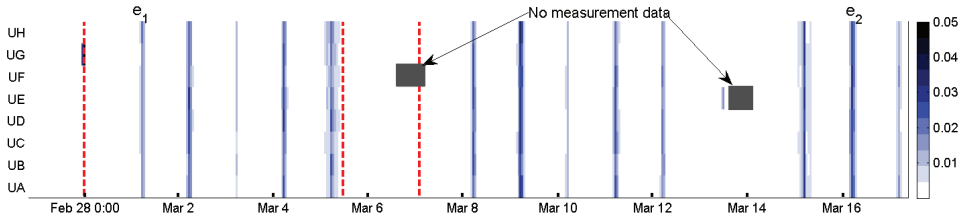
(d) Reverse paths (grouped by destinations).

Figure 5.7: Heat-map time series for the packet loss events in the forward and reverse paths.

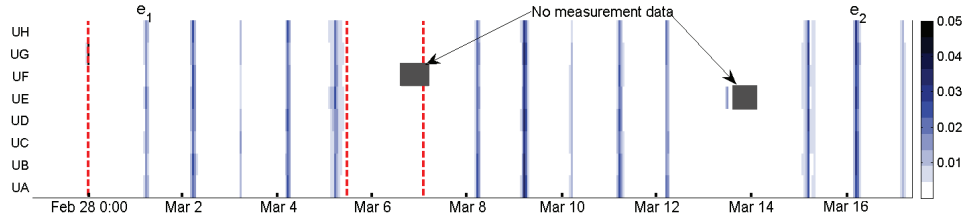
5.3.2 Loss-pair analysis of the paths to PL009

In this section, we use the eight forward paths to PL009 as a case study of loss-pair analysis. Figure 5.8 shows the heat-map time series for the frequencies of events P01x– and P10xR00 obtained from the eight paths. We compute the frequency for each path using one-hour bins and grey

out all the bins with no measurement data. As shown, the two heat maps are very similar. The loss-pair frequencies of the forward paths were 1-3%, and they distributed in several loss episodes, each of which lasted for several hours. In the ensuing discussion, we zoom into two loss episodes e_1 and e_2 in Figure 5.8 observed on 1 March 2010 (during the period of the temporary network operation) and 16 March 2010 (after the transition to the new service provider), respectively, with the same time period between 02:00 and 11:00 UTC on each day.



(a) P01x–.



(b) P10xR00.

Figure 5.8: Heat-map time series for the frequencies of events P01x– and P10xR00 from UA–UH to PL009.

5.3.2.1 The loss episode e_1

Figure 5.9 shows the RTT time series for the first packets (i.e., p_{j-1}) for the paths from UA–UH during e_1. In each time series, we also superimpose the residual packets' RTTs for events P01x– and P10xR00 observed from the corresponding path. As shown, the time series for UC is simi-

lar to that for UB, and UF–UH to UE. The following highlights the main observations from Figure 5.9:

1. A minimum RTT (minRTT) of 30 ms was found for the path from UD and 32 ms for the others. Most of the RTTs were found below 100 ms for each path.

2. Except for the UA path, other paths experienced two RTT surges at around 02:15 and 06:15 UTC.

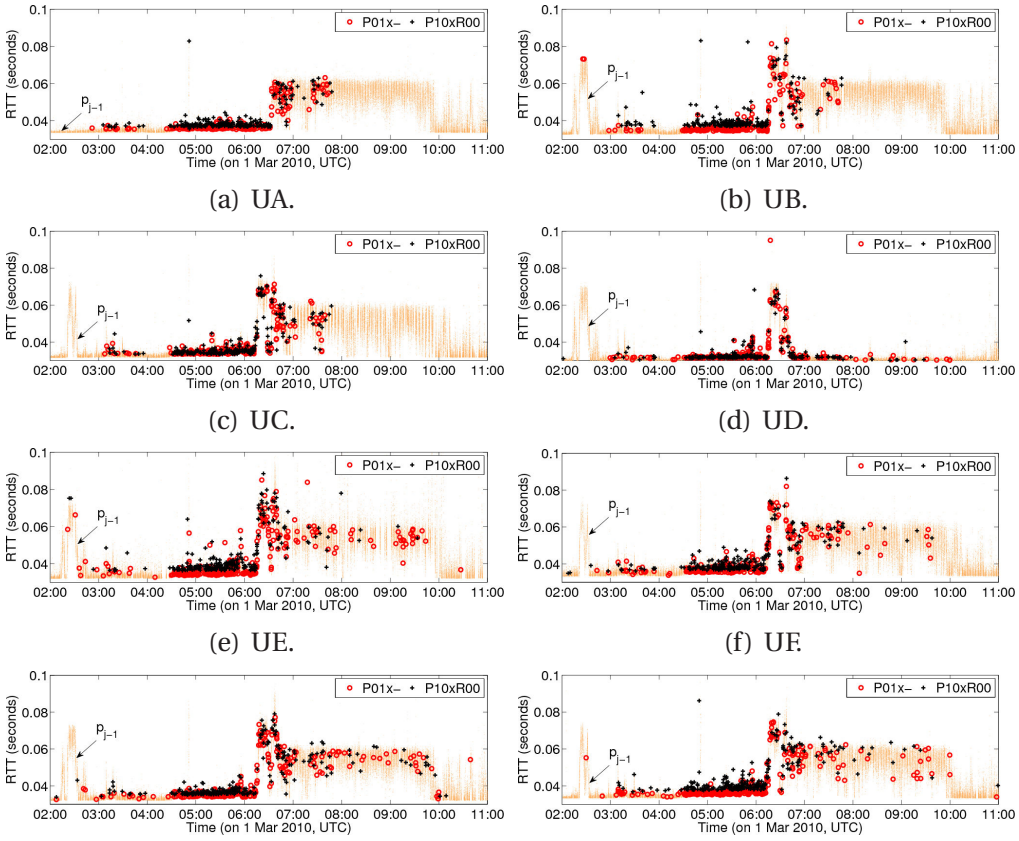
3. Forward-path loss pairs were observed between 03:00 and 07:45 UTC from all the paths.

4. Forward-path loss pairs were observed between 07:45 and 11:00 UTC only from the UD–UH paths.

5. The first packets' RTTs remained low and relatively stable between 02:35 and 06:15 UTC for all the paths. The loss pairs' RTTs clustered around the peaks and most of residual packets' RTTs for event P10xR00 were higher than that for event P01x– (which is consistent with our analysis in Section 5.2).

6. The first packets' RTTs became high and unstable after 06:15 UTC (especially between 06:15 and 09:45 UTC) except for the UD path. A significant variation was also observed from the loss pairs' RTTs, and many residual packets' RTTs for event P10xR00 were found lower than that for event P01x–.

Since diverse RTT characteristics were observed among the eight paths, we further analyze the tcptraceroute results for both the forward and re-



(a) UA. (b) UB.

(c) UC. (d) UD.

(e) UE. (f) UF.

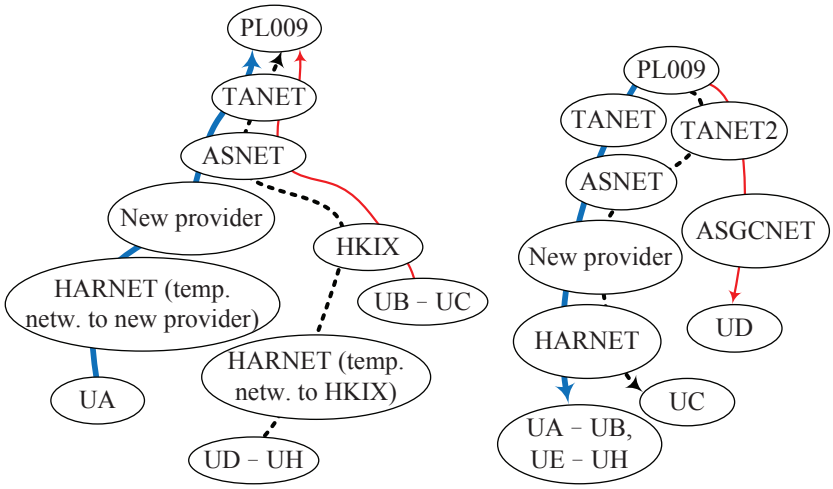
(g) UG. (h) UH.

Figure 5.9: RTT time series for the paths from UA–UH to PL009 during the loss episode e_1.

verse paths between the eight sources and the destination, and find that the sources actually used different IP routes to reach the destination. The tcptraceroute results also reveal that no IP route change occurred during e_1.

Figure 5.10(a) shows the eight forward paths to PL009. As shown, while the forward paths from UB–UH went through the peering of HKIX towards ASNET, the path from UA actually went through the new service provider to ASNET. As a result, the two RTT surges (point 2 above) were

probably introduced by the HKIX network. Besides, we observe that only ASNET and TANET were involved in all the eight forward paths. Therefore, the loss pairs observed between 03:00 and 07:45 UTC (point 3) were probably introduced by a congestion point near the destination. However, since only UD–UH went through the temporary network to HKIX during e_1, the loss pairs observed between 07:45 and 11:00 UTC from their paths (point 4) were likely due to the congestion in this temporary network.



(a) Forward paths. (b) Reverse paths.

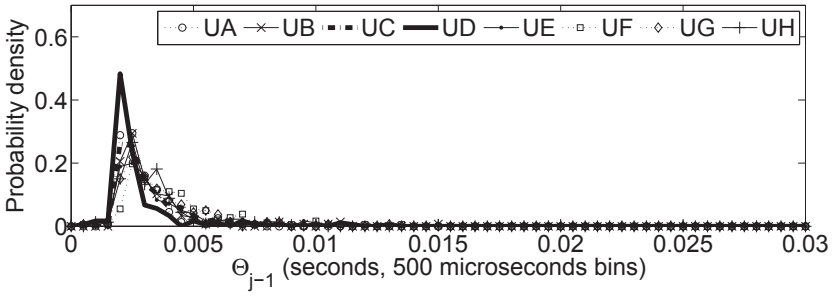
Figure 5.10: A comparison of forward and reverse paths between UA–UH and PL009 during the loss episode e_1.

On the other hand, the tcptraceroute for the reverse paths provide additional information to reveal the effect of the reverse-path networks on the observed first packets' and loss pairs' RTTs. Figure 5.10(b) shows the reverse paths to the eight sources. As shown, only the reverse path to UD went through the ASGCNET network (with at least three router hops shorter). This observation suggests that the shorter minRTT observed

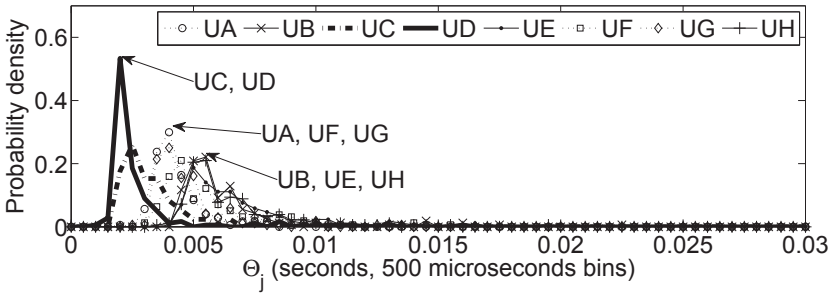
from the UD path (point 1) was probably due to the shorter IP reverse route. While the other reverse paths from PL009 went through ASNET, the new service provider, and then HARNET to the sources, these paths actually shared only three common router hops in ASNET. Therefore, the RTT fluctuation after 06:15 UTC observed from most of the paths, except for the UD path (point 6), was introduced by another common congestion point in the ASNET network on the reverse paths.

The observations above indicate that the eight paths exhibit relative stable RTTs and similar loss pairs' patterns between 02:35 and 06:15 UTC. To further characterize the properties for the eight forward paths, we compute the distributions of the residual packets' path queueing delays for events P01x– (i.e., Θ_{j-1}) and P10xR00 (i.e., Θ_j) in Figures 5.11(a)–5.11(b). Figure 5.11(a) shows that the modes of the path queueing delays for event P01x– were around 2 ms for the eight sources; therefore, the sources probably shared the same congestion point on their forward paths (which further supports our above findings). Moreover, by studying the distributions of the path queueing delays for event P10xR00 shown in Figure 5.11(b), we obtain additional fingerprints for the eight paths and can further classify the sources into three groups: (i) UC and UD; (ii) UA, UF, and UG; and (iii) UB, UE, and UH.

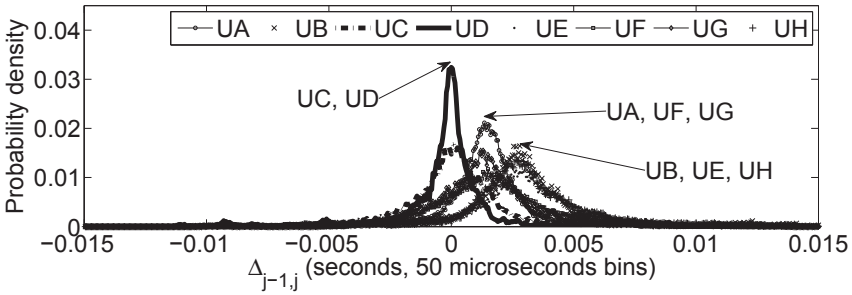
Figure 5.11(c) shows the $\Delta_{j-1,j}$ distribution for each path based on the mutual differences between the corresponding residual packets' path queueing delays for events P01x– and P10xR00. As shown, the $\Delta_{j-1,j}$ distributions for the three groups are distinct from each other, meaning that they experienced different $H^{(h^*)}$'s configurations during the time period.



(a) Θ_{j-1}.



(b) Θ_j.



(c) $\Delta_{j-1,j}$.

Figure 5.11: Path queueing delays for loss-pair events P01x– and P10xR00 and their differences obtained from UA–UH to PL009 between 02:35 and 06:15 UTC during the loss episode e_1.

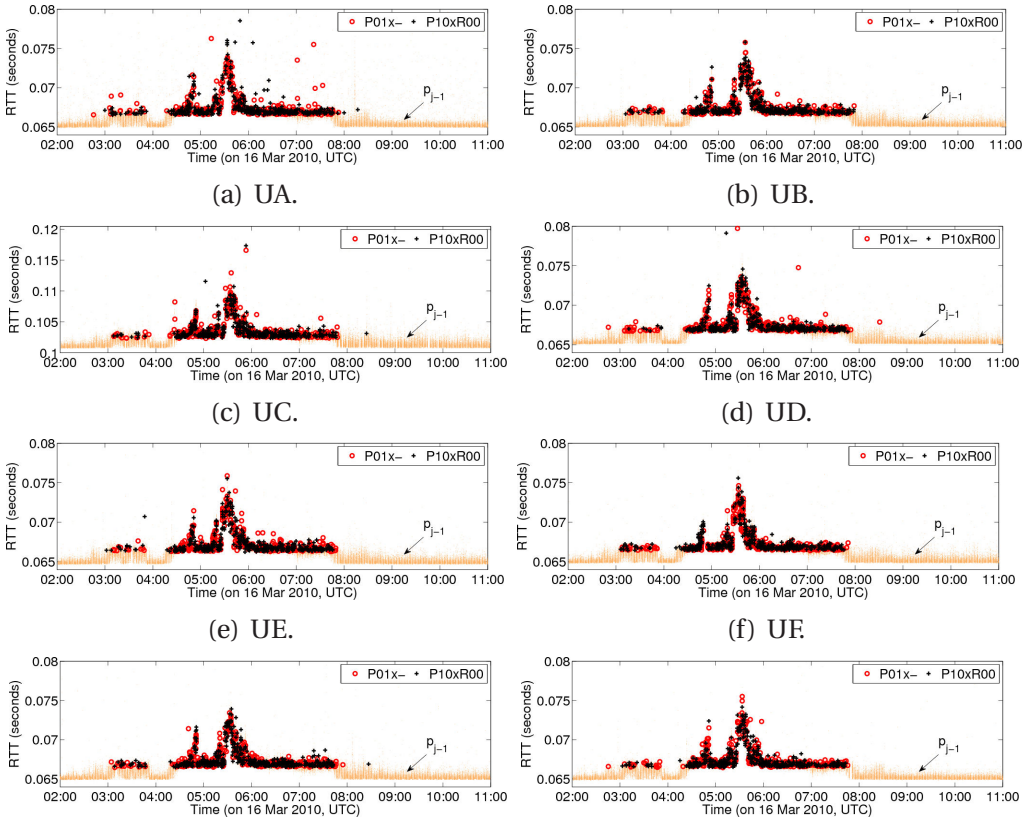
For group (i), the figure shows that the corresponding link capacity was at least greater than 100 Mbits/s. However, we are unable to determine the exact value due to the coarse packet timestamp resolution. For groups (ii) and (iii), the estimated link capacities were at least 3 Mbits/s and 1.5

Mbits/s, respectively. Overall, the loss-pair analysis provides more comprehensive comparison of the eight paths and their characteristics which would not be easily discovered by considering only the loss frequencies (Figures 5.7(a) and 5.8) or the packet-pair RTTs (Figure 5.9).

5.3.2.2 The loss episode e_2

Figure 5.12 plots the time series of the first packets' RTTs observed from the eight paths to PL009 during e_2. Similarly, we superimpose the residual packets' RTTs for events P01x– and P10xR00 on the first packets' RTTs. Since this loss episode was located after the transition to the new service provider, it gives different path characteristics as compared with e_1. The figure shows that the minRTT was around 65 ms and most of the RTTs fell below 75 ms, except for the UC path whose RTTs ranged between 101 ms and 119 ms. However, notice that all the eight paths exhibit very similar RTT time series patterns, including two RTT surges at 04:50 and 05:30 UTC.

Moreover, forward-path loss pairs were observed from all the eight paths between 03:00 and 07:45 UTC. It is interesting to note that this time period is exactly the same as that in e_1 when forward-path loss pairs also existed in all the paths, although the loss pairs' RTTs found in e_2 mostly hit the highest values. This observation suggests that the transition event did *not* affect the congestion point in the forward path. Our tcptraceroute results for the forward paths obtained in e_2 also show that the forward paths still went through the same hops in ASNET and TANET observed during e_1.



(a) UA. (b) UB.

(c) UC. (d) UD.

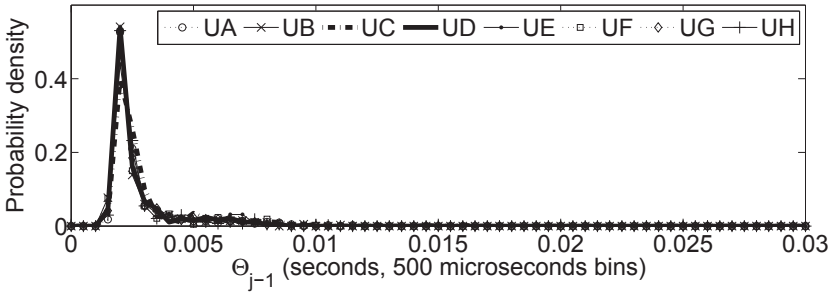
(e) UE. (f) UF.

(g) UG. (h) UH.

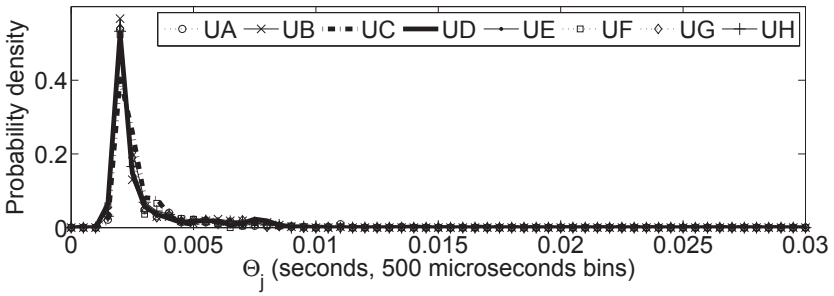
Figure 5.12: RTT time series for the paths from UA–UH to PL009 during the loss episode e_2.

We obtain the path queueing delays to further examine the impact of the transition to the new service provider. Figure 5.13 plots the distributions of the path queueing delays for events P01x– and P10xR00, and their differences obtained from the eight paths between 02:35 and 06:15 UTC during e_2. Figure 5.13(a) shows that the modes of the path queueing delays for event P01x– were still around 2 ms. Therefore, the transition probably had no impact on the congestion point encountered by the eight paths. However, Figures 5.13(b) and 5.13(c) show that the transition affected the configuration of $H^{(h^*)}$ for the eight paths (comparing

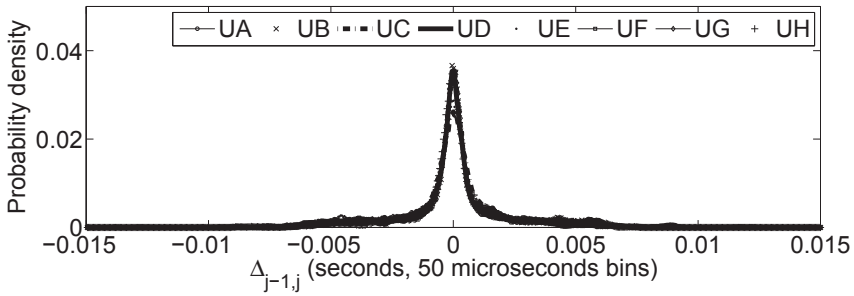
with Figures 5.11(b) and 5.11(c)), where the distributions for both $\Delta_{j-1,j}$ and path queueing delays for event P10xR00 are very similar among the eight sources. As a result, the sources likely shared the same hop at $H^{(h^*)}$ after the transition.



(a) Θ_{j-1}.



(b) Θ_j.

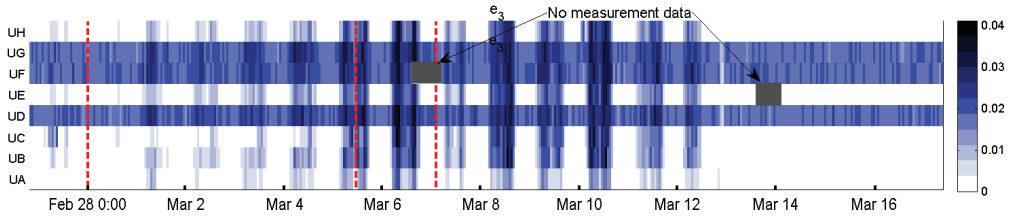


(c) $\Delta_{j-1,j}$.

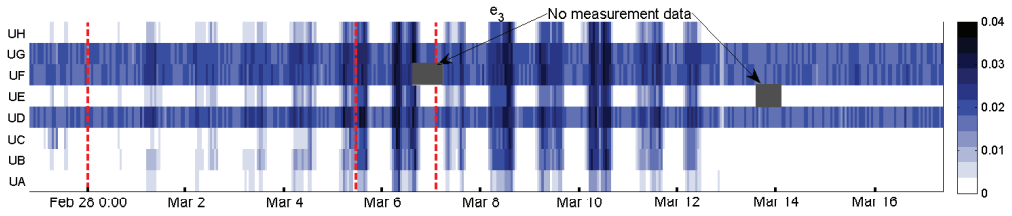
Figure 5.13: Path queueing delays for loss-pair events P01x– and P10xR00 and their differences obtained from UA–UH to PL009 between 02:35 and 06:15 UTC during the loss episode e_2.

5.3.3 Loss-pair analysis of the paths to PL014

In this section, we apply the loss-pair analysis to the eight sources' reverse paths from PL014. Recall from Figure 5.7(d) that the reverse paths from PL014 exhibited significant packet loss during the measurement period. Figure 5.14 shows the heat-map time series for the frequencies of events P00xR01 and P00xR10. Similarly, the grey areas indicate the periods with no measurement data. We also note that the two heat maps exhibit a similar pattern. Figure 5.14 shows that the frequency for each path was less than 4%. The reverse paths to UD, UF, and UG suffered a long-term loss episode for the entire measurement period, and the paths to others encountered several loss episodes before 13 March 2010.



(a) P00xR01.



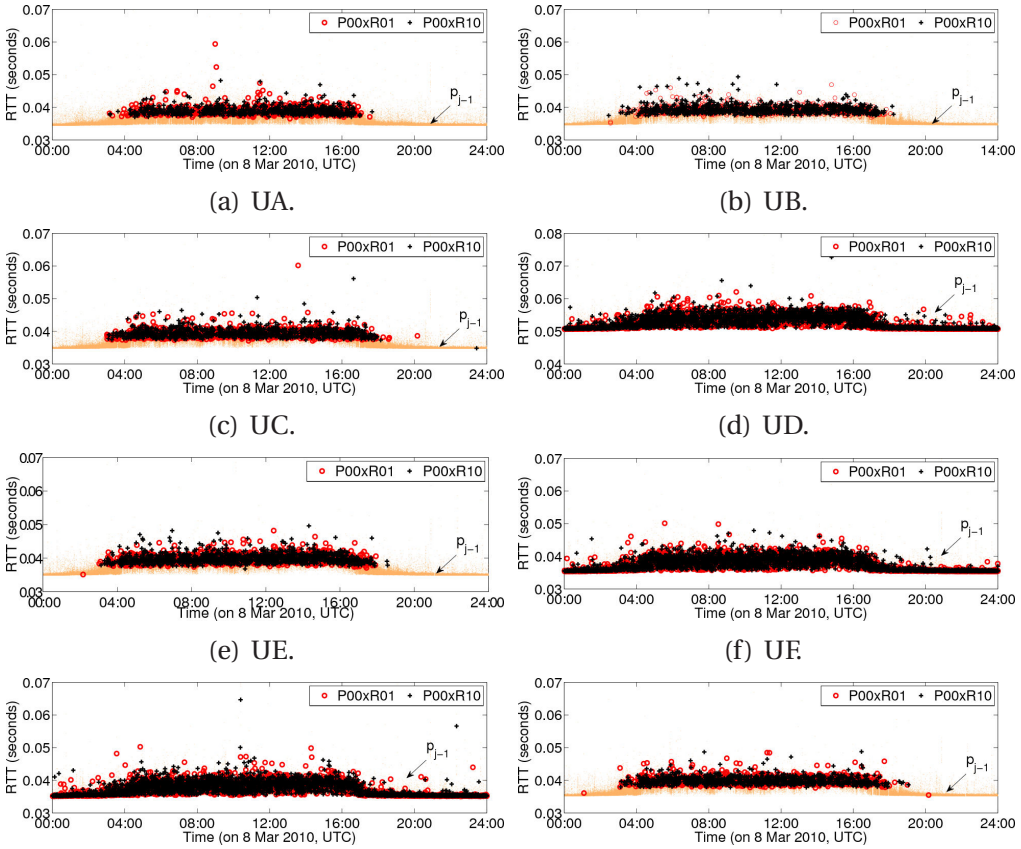
(b) P00xR10.

Figure 5.14: Heat-map time series for the frequencies of events P00xR01 and P00xR10 from UA–UH to PL014.

5.3.3.1 The loss episode e_3

We analyze a reverse-path loss episode (labeled as e_3 in Figure 5.14) between 00:00 and 23:59 UTC on 8 March 2010. Figure 5.15 plots the RTT time series of the first packets and the residual packets' RTTs for events P00xR01 and P00xR10 obtained from the paths for UA–UH to PL014 during e_3. As shown, the time series for UA, UB, UC, UE, and UH exhibit an RTT inflation period between 03:00 and 18:00 UTC, and most of the loss pairs were found within this RTT inflation period. On the other hand, the time series for UD, UF, and UG also show an RTT inflation period, but the loss pairs can be found throughout the measurement period. Therefore, we classify the eight sources into two groups: (i) UA, UB, UC, UE, and UH; and (ii) UD, UF, and UG. Moreover, we observe a minRTT of 50 ms for the UD path, and 35 ms for the other paths.

To characterize the congestion node's state encountered by the two groups of paths, we plot the path queueing delays for events P00xR01 and P00xR10 and their differences during e_3 in Figures 5.16(a)-5.16(c). In particular, Figures 5.16(a)-5.16(b) show that group (i) exhibits a single mode at around 4 ms for the path queueing delays for both events P00xR01 and P00xR10. According to Figure 5.15, the path queueing delays represent the congestion experienced by the group of sources during the RTT inflation period, during which almost all the loss pairs were found. We also notice that group (ii) exhibits a similar (but weaker) mode at around 3.5 ms, but the mode vanishes if we only consider the loss pairs outside the RTT inflation period. Consequently, it seems that all eight sources suf-



(a) UA. (b) UB.

(c) UC. (d) UD.

(e) UE. (f) UF.

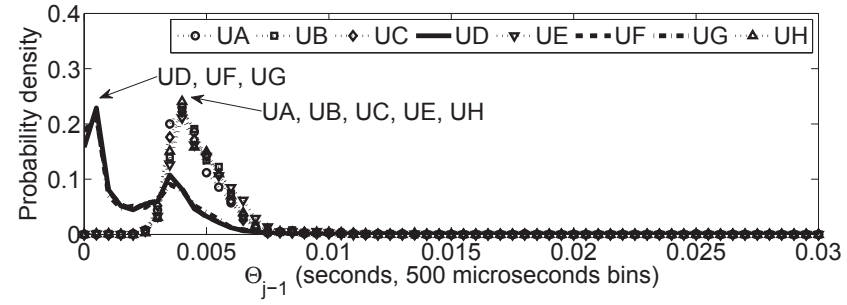
(g) UG. (h) UH.

Figure 5.15: RTT time series for the paths from UA–UH to PL014 during the loss episode e_3.

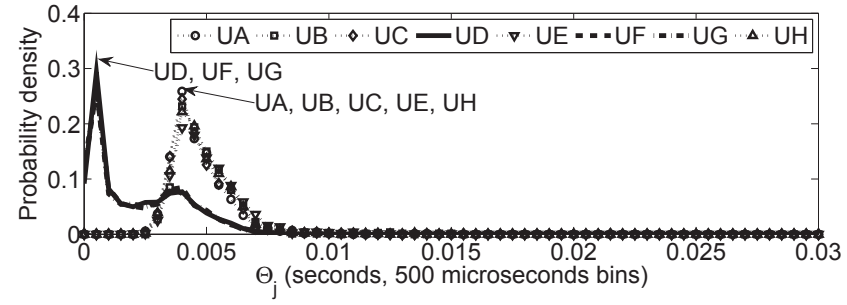
fered from the same congestion point during the RTT inflation. Based on the tcptraceroute results, the congestion point was very likely a router hop in the destination's network which was the only one present in all eight reverse routes.

Figures 5.16(a)-5.16(b) also show that group (ii) exhibits another stronger mode at around 500 μs. A further investigation finds that the mode was contributed by the loss pairs across the loss episode e_3 (instead of only outside the RTT inflation period). This observation suggests that multi-

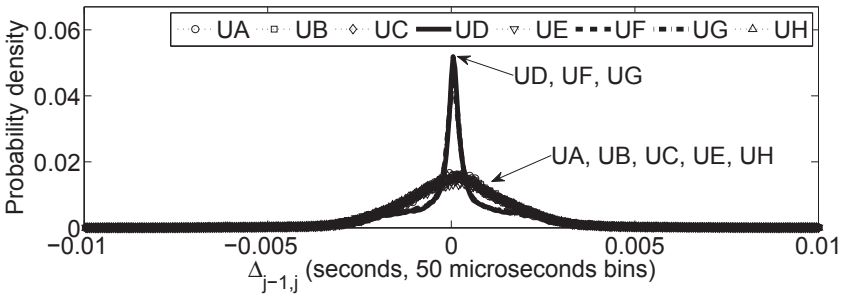
ple congestion points existed across e_3 for the paths in group (ii). Moreover, it is interesting to note from Figure 5.16(c) that the two groups experienced a similar $H^{(h^*)}$'s link capacity estimate of at least 100 Mbits/s.



(a) Θ_{j-1}.



(b) Θ_j.



(c) $\Delta_{j-1,j}$.

Figure 5.16: Path queueing delays for loss-pair events P00xR01 and P00xR10 and their differences obtained from UA–UH to PL014 during the loss episode e_3.

5.4 Discussion

There are some limitations to our loss-pair measurement technique. The obvious one is that the applicability of the loss-pair measurement depends on the prevalence of the loss-pair event. Besides, depending on the packet timestamp resolution, the link capacity of the preceding bottleneck hop is not always measurable especially for a small probe (response) packet size. However, we notice that such technique is still a viable method to obtain additional characteristics of Internet paths.

Although HTTP/OneProbe distinguishes all the six loss-pair events shown in Figure 5.1, our analysis did not consider the residual packets' delays for the two loss-pair events (i.e., P10xR10 and P10xR01) for which loss pairs exist in both uni-directional paths. Comparing the two loss-pair events, we note that the difference between the delivery statuses for their response packets can be exploited to further analyze the reverse-path characteristics. For example, since the pair of response packets are elicited by a single probe packet, we could obtain the distribution of the mutual difference between the path queueing delays for the two loss-pair events to examine the configuration of preceding bottleneck hop which is located in the reverse path.

Finally, we do not have a complete understanding of how multiple congested hops affect the residual packets' delays of the loss pairs. Although previous work [132] showed that the distribution of the LP_{10}'s delay depends on the buffer sizes of the congested hops, the distribution of the LP_{01}'s delay and the effect of preceding bottleneck hop have yet to

explore. On the other hand, Active Queue Management (AQM) schemes (e.g., RED [88] and BLUE [86]) can introduce significant fluctuation in the loss pairs' delays [132]. Extending our analysis to consider these common network scenarios is a subject for future work.

5.5 Conclusions

In this chapter, we revisited the loss-pair measurement method proposed a decade ago. Based on our new analysis and Internet measurement results, we concluded that the loss-pair measurement is a very useful method for correlating a packet loss event and the delay that would be experienced by this lost packet. This correlation provides insight into, for example, the congested node's state upon packet drop, capacity of a link preceding the congested node, and loss-pair asymmetry. Moreover, we have successfully incorporated the loss-pair measurement into HTTP/OneProbe to obtain useful path signatures for path fingerprinting and detecting common congestion points for multiple paths.

6

Conclusions and Future Work

Our goal of this thesis is to design robust techniques for measuring network path properties. We proposed and studied three new measurement methods based on packet pairs: the minimum delay difference (MDDIF) method for mitigating cross-traffic interference on path capacity measurement, TRIO for eliminating measurement-traffic interference on asymmetric capacity measurement, and the last one for measuring additional end-to-end properties, including loss-pair frequency and path queueing delay, from forward and reverse paths. While the packet-pair dispersion (PPD) has played an essential role in the end-to-end network measurement, it suffers from a number of inherent limitations that have been disregarded by previous measurement techniques. To deal with

the limitations, our proposed methods consider individual packet delays measured from packet pairs and infer various asymmetric-path properties from the delays. Moreover, unlike many existing techniques that require mutual cooperation between a local and a remote measuring endpoints, our implementations for the three methods seamlessly measure the asymmetric-path properties with non-cooperative remote endpoints.

First, we noticed that the minimum delay sum (MDSUM) method, based on the PPD of a packet pair that are unaffected by cross traffic, is not the most efficient approach to measuring path capacity, because an unaffected packet pair is generally more difficult to obtain than a single unaffected probe packet. Therefore, we proposed the MDDIF method to achieve fast and accurate path capacity measurement. Our approach uses minimal possible packet delays (minDelays) of packet pairs as a means for estimating an unbiased PPD. A minDelay is defined as the delay experienced by a packet in a packet pair for which the packet does not encounter any cross-traffic-induced queueing delay on the network path.

We demonstrated the effectiveness of the MDDIF method through a series of analysis and experiments. Based on the first-passage-time analysis, we showed that the MDDIF method consistently requires no more time than the MDSUM method to obtain an unbiased PPD, and demonstrated the impacts of the probe packet size and cross-traffic packet size on acquiring the first and second packets' minDelay. Moreover, we evaluated the speed gain of the MDDIF method over the MDSUM method under a multi-hop stochastic network model and an in-lab testbed. The

Internet experiments showed that the MDDIF method avoids the PPD variability issue observed from an ADSL environment in [70] and accurately estimates both the forward-path capacity and reverse-path capacity.

Second, since the PPD introduced by a local bottleneck hop (LBH) on a round-trip path can be distorted by subsequent LBHs on the same path, existing methods—SProbe, AsymProbe, and DSLprobe—implement two probing strategies, namely round-trip probe (RTP) and two-way probe (TWP), and acquire the probes' PPD from a local endpoint for asymmetric capacity measurement. Due to the packet-size restriction, the RTP suffers from two types of interferences—probe interference and response interference—in the midst of high degree of capacity asymmetry. Although the TWP is immune from both interferences, it only aims for the reverse-path capacity measurement.

We therefore designed and implemented TRIO, a non-cooperative method for measuring any degree of capacity asymmetry. A key feature of TRIO is to obtain minimum round-trip times (minRTTs) from an interleaved sequence of RTPs and TWPs instead of their PPDs. Such novel perspective enables TRIO to eliminate the probe and response interferences and measure both forward-path capacity and reverse-path capacity at the same time. Moreover, we implemented TRIO and designed three self-diagnosis tests to filter out artifacts due to packet reordering and loss, and invalid minRTTs and capacity estimates. Our evaluation considered the TRIO's performance under various degrees of capacity asymmetry in the testbed and the Internet, and compared and contrasted its perfor-

mance with the performance of SProbe, AsymProbe, and DSLprobe.

Finally, we notice that the PPD measurement discards a packet pair if at least one of the probe packets is lost. However, the residual packet in a loss pair (i.e., a packet pair experiencing a single packet loss) is still useful for inferring the network path condition encountered by the lost packet. Despite this additional benefit shared by existing methods based on packet pairs, the loss-pair measurement has so far received much less attention. Therefore, we developed a non-cooperative method for measuring both forward-path and reverse-path loss pairs from a single endpoint. Moreover, such method can work seamlessly together with the MDDIF method and TRIO for discovering additional network path properties.

We analyzed the two loss-pair events (P10xR00 and P00xR10) for the loss pair LP_{10} (where the first packet of a packet pair is discarded by a congested node) and the other two (P01x– and P00xR01) for the LP_{01} (where the second packet of a packet pair is discarded) on the round-trip path, and showed that the residual packet's delay for the LP_{10} always includes additional bias introduced by a hop preceding the congested node. Therefore, leveraging the residual packet's delay for the LP_{01} is generally more accurate on inferring the the congested node's state upon packet loss. Nonetheless, the additional bias is useful for exploring the link capacity of the preceding bottleneck hop.

Our testbed experiments and Internet measurements between eight local measurement points and 11 PlanetLab nodes demonstrated that the loss-pair measurement is a viable method for determining the con-

gested node's state upon dropping a packet, link capacity of the preceding bottleneck hop, loss-pair asymmetry, and fingerprinting a congested network path. We also demonstrated the usability of heat-map diagram for correlating the spatial and temporal domains on the perceived loss-pair behavior, such as the variability and significance of loss pairs across the two domains.

6.1 Future work

Although we believe that our works given in this thesis have shown the feasibility and benefits of inferring network path properties based on packet delay and non-cooperative measurement, there are still a few issues to explore:

1. The variability of cross traffic can affect the queueing probabilities of the first and second packets in the packet pair. Therefore, we will extend the current analytical model to examine the effects of different cross traffic distributions (e.g., deterministic, Pareto `ON/OFF`, and long range dependence (LRD)) on the performance of the MD-DIF method. We will also consider the burstiness and correlation structure of cross traffic introduced by, for example, TCP self-clocking [109] and segmentation of UDP messages [108]. Our analytical model will be further extended to evaluate the speed of TRIO and other existing methods for measuring capacity asymmetry. Moreover, we will study various statistical tests (e.g., the bootstrap method [81]

and the non-parametric interval estimation [79]) to detect the convergence of minDelay.

2. Another logical extension of the MDDIF method and TRIO is to consider the path capacity measurement in the presence of multichannel bottleneck link [175]. In particular, we will incorporate the MDDIF method into packet trains with a range of lengths to accommodate the multichannel effects. For a k-channel link, when the last packet of a packet train of length $k+1$ could queue behind a preceding probe packet at one of the k channels, we conjecture that the minDelays of the preceding packet and last packet could be exploited for the capacity estimation. Moreover, we will extend the MDDIF method and TRIO to measure both the peak rate and sustainable rate of a traffic shaper.

3. We will consider other practical issues that could impede the capacity measurement conducted by the MDDIF method and TRIO. These include the traffic regulation (e.g., token bucket regulator in DOCSIS-compliant cable broadband network) [129], packet aggregation in optical packet-switched networks [96], packet-forwarding prioritization (e.g., priority queueing and weighted fair queueing) [114, 135], and packet compression (e.g., robust header compression, ROHC) [44, 63].

4. We will devise new measurement methods for other asymmetric-path metrics. For example, we are designing a non-cooperative approach for measuring available bandwidth asymmetry. To the best

of our knowledge, all the existing non-cooperative methods measure available bandwidth for either the round-trip path [47], forward path [49, 71, 101, 149], or reverse path [35] (refer to Table 2.3).

5. A possible direction to extending the loss-pair measurement is to design a systematic approach for obtaining useful signatures for path fingerprinting and detecting common congestion points for multiple network paths. The seminal work given by Sinha *et al.* [205] has shown the feasibility of using one-way PPD to fingerprint Internet paths. We believe that using the loss-pair measurement and neighbor-cooperative measurement technique [55] can supplement the PPD-based path fingerprinting by analyzing different types of loss pairs observed by a set of neighboring measurement nodes. On the other hand, we will further analyze the residual packets' delays for the two loss-pair events (i.e., P10xR10 and P10xR01) for which loss pairs exist in both the forward and reverse paths, and study the effects of multiple congested hops and AQM schemes on the loss-pair measurement.

6. Our current measurement focuses on the fixed (wired) network path where both the source and destination are static and unique throughout the measurement period. However, we notice that, different from the wired network, wireless networks could encounter a significant amount of node arrivals and departures. With the widespread deployment of wireless networks, the network churning should be regarded as an important issue for the measurement conducted in

the wireless network and is an area for future work. For example, we will devise new measurement methods for handling dynamic measurement nodes that can leave the network without notice. On the other hand, we will also study the performance of our proposed methods for measuring link capacity in wireless network (e.g., 802.11 link capacity), where the capacity can be dynamically adjusted due to, for example, the rate adaption algorithms (e.g., ARF [112] and RBAR [99]), energy saving schemes (e.g., MiSer [188]), and interference from other transmissions in the network [117, 233].

A

Derivations of expected FPT and relative gain of the MDDIF method

A.1 Expected FPT of the MDDIF method

The expected FPT of the MDDIF method is given by

$$E[T_{DIF}] = \pi_0 \mathbf{t},$$

where $\mathbf{t} = [t_0, t_1, t_2]^T$ for $t_k,\ k = 0, 1, 2$, is the expected number of steps taken prior to reaching the absorbing state, given that the chain begins from state k, and π_0 is the vector of the initial state probabilities. From

[120], $\mathbf{t} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{c}$, where $\mathbf{I}$ is an identity matrix, $\mathbf{c} = [1\ 1\ 1]^T$, and $(\mathbf{I}-\mathbf{Q})^{-1}$ is the fundamental matrix of the Markov chain. Since $\mathbf{Q}$ is given by

$$\mathbf{Q} = \begin{bmatrix} p_{XY}(0,0) & p_{XY}(0,1) & p_{XY}(1,0) \\ 0 & 1-p_X & 0 \\ 0 & 0 & 1-p_Y \end{bmatrix},$$

we have

$$(\mathbf{I}-\mathbf{Q})^{-1} = \begin{bmatrix} \frac{1}{1-p_{XY}(0,0)} & \frac{p_{XY}(0,1)}{(1-p_{XY}(0,0))p_X} & \frac{p_{XY}(1,0)}{(1-p_{XY}(0,0))p_Y} \\ 0 & \frac{1}{p_X} & 0 \\ 0 & 0 & \frac{1}{p_Y} \end{bmatrix}.$$

Therefore,

$$\begin{aligned} \mathbf{t} &= (\mathbf{I}-\mathbf{Q})^{-1}\mathbf{c}, \\ &= \begin{bmatrix} \frac{1}{1-p_{XY}(0,0)}\left(1+\frac{p_{XY}(0,1)}{p_X}+\frac{p_{XY}(1,0)}{p_Y}\right) \\ \frac{1}{p_X} \\ \frac{1}{p_Y} \end{bmatrix}. \end{aligned}$$

Since the MDDIF method starts from state 0, $\pi_0 = [1\ 0\ 0]$. Therefore,

$$\begin{aligned} E[T_{DIF}] &= \pi_0\mathbf{t}, \\ &= \frac{1}{1-p_{XY}(0,0)}\left(1+\frac{p_{XY}(0,1)}{p_X}+\frac{p_{XY}(1,0)}{p_Y}\right). \end{aligned}$$

A.2 Relative gain of the MDDIF method

The relative gain of the expected FPT for the MDDIF method is given by:

$$\begin{aligned}
\Psi &= \frac{E[T_{SUM}] - E[T_{DIF}]}{E[T_{SUM}]}, \\
&= \left[\frac{1}{p_{XY}(1,1)} - \frac{1}{1-p_{XY}(0,0)}\left(1 + \frac{p_{XY}(0,1)}{p_X} + \frac{p_{XY}(1,0)}{p_Y}\right)\right] \Big/ \frac{1}{p_{XY}(1,1)}, \\
&= \frac{p_{XY}(0,1)p_{XY}(1,0)(p_X + p_Y)}{(1-p_{XY}(0,0))p_X p_Y}, \\
&= \frac{\sigma}{\sigma + \xi},
\end{aligned}$$

where

$$\begin{aligned}
\sigma &= p_{XY}(0,1)p_{XY}(1,0)(p_X + p_Y), \\
\xi &= p_{XY}(1,1)[p_{XY}(0,1)(p_X + p_Y) + p_X^2].
\end{aligned}$$

Bibliography

[1] Alexa the web information company. http://www.alexa.com/.

[2] CAIDA : tools. http://www.caida.org/tools/.

[3] Debian worldwide mirror sites. http://www.debian.org/mirror/list.

[4] endace. http://www.endace.com/.

[5] Fedora public active mirrors. http://mirrors.fedoraproject.org/publiclist/.

[6] GENI. http://www.geni.net/.

[7] Gentoo Linux – Gentoo Linux mirrors. http://www.gentoo.org/main/en/mirrors2.xml.

[8] IP performance metrics (IPPM) working group. http://www.ietf.org/html.charters/ippm-charter.html.

[9] Iperf. http://iperf.sourceforge.net/.

[10] M-Lab. http://www.measurementlab.net/.

[11] MAWI working group traffic archive. http://mawi.wide.ad.jp/mawi/.

[12] Mirrors - openSUSE. http://mirrors.opensuse.org/.

[13] Netcraft. http://news.netcraft.com/.

[14] NLANR AMP. http://amp.nlanr.net/.

[15] Nmap. http://nmap.org/.

[16] One-way ping (OWAMP). http://e2epi.internet2.edu/owamp/.

[17] PlanetLab. http://www.planet-lab.org.

[18] TCPDUMP/LIBPCAP public repository. http://www.tcpdump.org/.

[19] Telecom Argentina. http://www.telecom.com.ar/adsl/20M.html.

[20] The FAB-Probe Project. http://www.eurecom.fr/∼btroup/fabprobe.html.

[21] Welcome to OneLab. http://www.onelab.eu/.

[22] DOCSIS - Cisco Support Community. https://supportforums.cisco.com/docs/DOC-1239, 2009.

[23] OECD broadband statistics. http://www.oecd.org/sti/ict/broadband/, Oct. 2009.

[24] Network monitoring tools. http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html, 2010.

[25] The Joint Universities Computer Centre (JUCC). http://www.jucc.edu.hk/jucc/harnet.html, May 2010.

[26] A. Adams, J. Mahdavi, M. Mathis, and V. Paxson. Creating a scalable architecture for Internet measurement. In *Proc. INET*, 1998.

[27] M. Aida, N. Miyoshi, and K. Ishibashi. A scalable and lightweight QoS monitoring technique combining passive and active approaches. In *Proc. IEEE INFOCOM*, 2003.

[28] A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area Internet bottlenecks. In *Proc. ACM/USENIX IMC*, 2003.

[29] M. Allman. Measuring end-to-end bulk transfer capacity. In *Proc. ACM SIGCOMM IMW*, 2001.

[30] M. Allman, W. Eddy, and S. Ostermann. Estimating loss rates with TCP. *ACM SIGMETRICS Perform. Eval. Rev.*, 31(3):12–24, 2003.

[31] G. Almes, S. Kalidindi, and M. Zekauskas. A one-way delay metric for IPPM. RFC 2679, IETF, September 1999.

[32] G. Almes, S. Kalidindi, and M. Zekauskas. A one-way packet loss metric for IPPM. RFC 2681, IETF, September 1999.

[33] G. Almes, S. Kalidindi, and M. Zekauskas. A round-trip delay metric for IPPM. RFC 2680, IETF, September 1999.

[34] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. ACM SOSP*, 2001.

[35] D. Antoniades, M. Athanatos, A. Papadogiannakis, E. Markatos, and C. Dovrolis. Available bandwidth measurement as simple as running Wget. In *Proc. PAM*, 2006.

[36] B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the Internet. In *Proc. ACM/USENIX IMC*, 2007.

[37] B. Augustin, T. Friedman, and R. Teixeira. Multipath tracing with Paris traceroute. In *Proc. E2EMON*, 2007.

[38] F. Baccelli, S. Machiraju, D. Veitch, and J. Bolot. The role of PASTA in network measurement. In *Proc. ACM SIGCOMM*, 2006.

[39] F. Baccelli, S. Machiraju, D. Veitch, and J. Bolot. On optimal probing for delay and loss measurement. In *Proc. ACM/USENIX IMC*, 2007.

[40] H. Balakrishnan, V. Padmanabhan, G. Fairhurst, and M. Sooriyabandara. TCP performance implications of network path asymmetry. RFC 3449, IETF, Feb. 2008.

[41] J. Bellardo and S. Savage. Measuring packet reordering. In *Proc. ACM SIGCOMM IMW*, 2002.

[42] S. Bellovin. A technique for counting NATted hosts. In *Proc. ACM SIGCOMM IMW*, 2002.

[43] J. Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proc. ACM SIGCOMM*, 1993.

[44] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. RFC 3095, IETF, July 2001.

[45] N. Brownlee and C. Loosley. Fundamentals of Internet measurement: A tutorial. Keynote Systems, 2001.

[46] CAIDA. Skitter. http://www.caida.org/tools/measurement/skitter/.

[47] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27-28:297–318, 1996.

[48] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple Network Management Protocol (SNMP). RFC 1157, IETF, May 1990.

[49] S. Chakravarty, A. Stavrou, and A. Keromytis. Linkwidth: A method to measure link capacity and available bandwidth using single-end probes. Technical report, Columbia University, 2008.

[50] E. Chan, X. Luo, and R. Chang. A minimum-delay-difference method for mitigating cross-traffic impact on capacity measurement. In *Proc. ACM CoNEXT*, 2009.

[51] E. Chan, X. Luo, W. Li, W. Fok, and R. Chang. Measurement of loss pairs in network paths. In *Proc. ACM/USENIX IMC*, 2010.

[52] M. Chan, Y. Lin, and X. Wang. A scalable monitoring approach for service level agreements validation. In *Proc. IEEE ICNP*, 2008.

[53] R. Chang, E. Chan, W. Fok, and X. Luo. Sampling TCP data-path quality with TCP data probes. In *Proc. PFLDNeT*, 2009.

[54] R. Chang, E. Chan, W. Li, W. Fok, and X. Luo. Could ash cloud or deep-sea current overwhelm the Internet? In *Proc. USENIX HotDep*, 2010.

[55] R. Chang, W. Fok, W. Li, E. Chan, and X. Luo. Neighbor-Cooperative measurement of network path quality. In *Proc. IEEE Globecom*, 2010.

[56] B. Chen, Z. Zhou, Y. Zhao, and H. Yu. Efficient error estimating coding: Feasibility and applications. In *Proc. ACM SIGCOMM*, 2010.

[57] L. Chen. AsymProbe. http://www.cs.ucla.edu/NRL/CapProbe/download.htm, 2005.

[58] L. Chen, T. Sun, G. Yang, M. Sanadidi, and M. Gerla. End-to-end asymmetric link capacity estimation. In *Proc. IFIP Networking*, 2005.

[59] L. Cheng and I. Marsic. Accurate bandwidth measurement in xDSL service networks. *Computer Communications*, 25(18):1699–1710, 2002.

[60] L. Cheng and I. Marsic. Java-based tools for accurate bandwidth measurement of digital subscriber line networks. *Integr. Comput.-Aided Eng.*, 9(4):333–344, 2002.

[61] Y. Cheng, V. Ravindran, and A. Leon-Garcia. Internet traffic characterization using packet-pair probing. In *Proc. IEEE INFOCOM*, 2007.

[62] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and analysis of a streaming-media workload. In *Proc. USENIX USITS*, 2001.

[63] P. Chimento and J. Ishac. Defining network capacity. RFC 5136, IETF, Feb. 2008.

[64] B. Choi, S. Moon, R. Cruz, Z. Zhang, and C. Diot. Practical delay monitoring for ISPs. In *Proc. ACM CoNEXT*, 2005.

[65] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast (keynote address). In *Proc. ACM SIGMETRICS*, 2000.

[66] L. Ciavattone, A. Morton, and G. Ramachandran. Standardized active measurements on a tier 1 IP backbone. *IEEE Communications Mag.*, 41(6):90–97, 2003.

[67] M. Coates, A. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Processing Mag.*, 19(3):47–65, 2002.

[68] M. Coates and R. Nowak. Network loss inference using unicast end-to-end measurement. In *Proc. ITC Conf. IP Traffic, Modeling and Management*, 2000.

[69] L. Colitti, G. Di Battista, M. Patrignani, M. Pizzonia, and M. Rimondini. Investigating prefix propagation through active BGP probing. *Microprocess. Microsyst.*, 31(7):460–474, 2007.

[70] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. Biersack. Capacity estimation of ADSL links. In *Proc. ACM CoNEXT*, 2008.

[71] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. Biersack. Non-cooperative available bandwidth estimation towards ADSL links. In *Proc. IEEE Global Internet Symposium*, 2008.

[72] C. Demichelis and P. Chimento. IP packet delay variation metric for IP performance metrics (IPPM). RFC 3393, IETF, Nov. 2002.

[73] L. Deng and A. Kuzmanovic. Monitoring persistently congested Internet links. In *Proc. IEEE ICNP*, 2008.

[74] G. Denisa, H. Nguyen, M. Kurant, K. Argyraki, and P. Thiran. Netscope: Practical network loss tomography. In *Proc. IEEE INFOCOM*, 2010.

[75] M. Dischinger, A. Haeberlen, K. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proc. ACM/USENIX IMC*, 2007.

[76] M. Dischinger, A. Mislove, A. Haeberlen, and K. Gummadi. Detecting BitTorrent blocking. In *Proc. ACM/USENIX IMC*, 2008.

[77] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *Proc. IEEE INFOCOM*, 2001.

[78] C. Dovrolis, P. Ramanathan, and D. Moore. Packet dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Trans. Netw.*, 12(6):963–977, 2004.

[79] A. Downey. Using pathchar to estimate Internet link characteristics. In *Proc. ACM SIGCOMM*, 1999.

[80] N. Duffield, L. Presti, V. Paxson, and D. Towsley. Network loss tomography using striped unicast probes. *IEEE/ACM Trans. Netw.*, 14(4):697–710, 2006.

[81] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1994.

[82] T. En-Najjary and G. Urvoy-Keller. PPrate: A passive capacity estimation tool. In *Proc. E2EMON*, 2006.

[83] T. En-Najjary and G. Urvoy-Keller. Passive capacity estimation: Comparison of existing tools. In *Proc. SPECTS*, 2008.

[84] X. Fan and J. Heidemann. Selecting representative IP addresses for Internet topology studies. In *Proc. ACM/USENIX IMC*, 2010.

[85] N. Feamster and H. Balakrishnan. Packet loss recovery for streaming video. In *Proc. International Packet Video Workshop*, 2002.

[86] W. Feng, K. Shin, D. Kandlur, and D. Saha. The BLUE active queue management algorithms. *IEEE/ACM Trans. Netw.*, 10(4):513–528, 2002.

[87] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, IETF, June 1999.

[88] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[89] H. Fujinoki. Improving reliability for multi-home inbound traffic: MHLB/I packet-level inter-domain load-balancing. In *Proc. ARES*, 2009.

[90] L. Gharai, C. Perkins, and T. Lehman. Packet reordering, high speed networks and transport protocol performance. In *Proc. IEEE ICCCN*, 2004.

[91] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path-quality monitoring in the presence of adversaries. In *Proc. ACM SIGMETRICS*, 2008.

[92] A. Haeberlen, M. Dischinger, P. Gummadi, and S. Saroiu. Monarch: A tool to emulate transport protocol flows over the Internet at large. In *Proc. ACM/USENIX IMC*, 2006.

[93] K. Harfoush, A. Bestavros, and J. Byers. Robust identification of shared losses using end-to-end unicast probes. In *Proc. IEEE ICNP*, 2000.

[94] K. Harfoush, A. Bestavros, and J. Byers. Measuring bottleneck bandwidth of targeted path segments. In *Proc. IEEE INFOCOM*, 2003.

[95] K. Harfoush, A. Bestavros, and J. Byers. Measuring capacity bandwidth of targeted path segments. *IEEE/ACM Trans. Netw.*, 17(1):80–92, 2009.

[96] J. He and S. Chan. TCP and UDP performance for Internet over optical packet-switched networks. *Comput. Netw.*, 45(4):505–521, 2004.

[97] S. Hemminger. Network emulation with NetEm. In *Proc. Australia's National Linux Conference*, 2005.

[98] F. Heusden. httping. http://www.vanheusden.com/httping/.

[99] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *Proc. ACM MOBICOM*, 2001.

[100] N. Hu. IGI/PTR. http://www.cs.cmu.edu/∼hnn/igi/.

[101] N. Hu, L. Li, Z. Mao, P. Steenkiste, and J. Wang. Locating Internet bottlenecks: Algorithms, measurements, and implications. In *Proc. ACM SIGCOMM*, 2004.

[102] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, 2003.

[103] B. Huffaker, D. Plummer, D. Moore, and k. claffy. Topology discovery by active probing. In *SAINT-W '02: Proceedings of the 2002 Symposium on Applications and the Internet (SAINT) Workshops*, 2002.

[104] V. Jacobson. Pathchar: A tool to infer characteristics of Internet paths. ftp://ftp.ee.lbl.ogv/pathchar/.

[105] V. Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM*, 1988.

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

[107] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *Proc. USENIX OSDI*, 2000.

[108] H. Jiang and C. Dovrolis. Source-level IP packet bursts: Causes and effects. In *Proc. ACM/USENIX IMC*, 2003.

[109] H. Jiang and C. Dovrolis. Why is the Internet traffic bursty in short time scales? In *Proc. ACM SIGMETRICS*, 2005.

[110] G. Jin and B. Tierney. System capability effects on algorithms for network bandwidth measurement. In *Proc. ACM/USENIX IMC*, 2003.

[111] S. Kalidindi and M. Zekauska. Surveyor: An infrastructure for Internet performance measurements. In *Proc. INET*, 1999.

[112] A. Kamerman and L. Monteban. WaveLAN-II: A high-performance wireless LAN for the unlicensed band. *Bell Lab Technical Journal*, pages 118–133, 1997.

[113] S. Kang, A. Bhati, D. Loguinov, and X. Liu. On estimating tight-link bandwidth characteristics over multi-hop paths. In *Proc. IEEE ICDCS*, 2006.

[114] P. Kanuparthy and C. Dovrolis. DiffProbe: Detecting ISP service discrimination. In *Proc. IEEE INFOCOM*, 2010.

[115] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi. CapProbe: A simple and accurate capacity estimation technique. In *Proc. ACM SIGCOMM*, 2004.

[116] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A nonstationary Poisson view of Internet traffic. In *Proc. IEEE INFOCOM*, 2004.

[117] A. Kashyap, S. Ganguly, and S. Das. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Proc. ACM MOBICOM*, 2007.

[118] S. Katti, D. Katabi, C. Blake, E. Kohler, and J. Strauss. MultiQ: Automated detection of multiple bottleneck capacities along a path. In *Proc. ACM/USENIX IMC*, 2004.

[119] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, T. Anderson, and A. Krishnamurthy. Reverse traceroute. In *Proc. USENIX NSDI*, 2010.

[120] J. Kemeny and J. Snell. *Finite Markov Chains*. Springer, 1976.

[121] S. Keshav. A control-theoretic approach to flow control. In *Proc. ACM SIGCOMM*, 1991.

[122] L. Kleinrock. *Queueing Systems, Vol. 2: Computer Applications*. Wiley-Interscience, 1976.

[123] E. Kohler. The Click Modular Router Project. http://read.cs.ucla.edu/click/.

[124] R. Koodli and R. Ravikanth. One-way loss pattern sample metrics. RFC 3357, IETF, August 2002.

[125] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants). In *Proc. ACM SIGCOMM*, 2003.

[126] K. Lai and M. Baker. Measuring bandwidth. In *Proc. IEEE INFOCOM*, 1999.

[127] K. Lai and M. Baker. Measuring link bandwidths using a deterministic model of packet delay. In *Proc. ACM SIGCOMM*, 2000.

[128] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proc. USENIX Symp. Internet Technologies and Systems*, 2001.

[129] K. Lakshminarayanan, V. Padmanabhan, and J. Padhye. Bandwidth estimation in broadband access networks. In *Proc. ACM/USENIX IMC*, 2004.

[130] W. Li, J. Tang, D. Zhang, and G. Xie. Improved sample filtering method for measuring end-to-end path capacity. *Journal of Central South University of Technology*, 14(3):399–403, 2007.

[131] H. Lim, J. Hou, and C. Choi. Constructing Internet coordinate system based on delay measurement. *IEEE/ACM Trans. Netw.*, 13(3):513–525, 2005.

[132] J. Liu. *Characterizing Network Elements and Paths Using Packet Loss Behavior*. PhD dissertation, Boston University, 2003.

[133] J. Liu and M. Crovella. Using loss pairs to discover network properties. In *Proc. ACM SIGCOMM IMW*, 2001.

[134] J. Liu, I. Matta, and M. Crovella. End-to-end inference of loss nature in a hybrid wired/wireless environment. In *Proc. WiOpt*, 2003.

[135] G. Lu, Y. Chen, S. Birrer, F. Bustamante, and X. Li. POPI: A user-level tool for inferring router packet forwarding priority. *IEEE/ACM Trans. Netw.*, 18(1):1–14, 2010.

[136] M. Luckie, Y. Hyun, and B. Huffaker. Traceroute probe method and forward IP path inference. In *Proc. ACM/USENIX IMC*, 2008.

[137] X. Luo, E. Chan, and R. Chang. Design and implementation of TCP data probes for reliable and metric-rich network path monitoring. In *Proc. USENIX Annual Tech. Conf.*, 2009.

[138] X. Luo and R. Chang. Novel approaches to end-to-end packet reordering measurement. In *Proc. ACM/USENIX IMC*, 2005.

[139] X. Luo and R. Chang. On a new class of pulsing denial-of-service attacks and the defense. In *Proc. NDSS*, 2005.

[140] X. Luo, R. Chang, and E. Chan. Performance analysis of TCP/AQM under denial-of-service attacks. In *Proc. IEEE MASCOTS*, 2005.

[141] S. Machiraju. *Theory and Practice of Non-intrusive Active Network Measurements*. PhD dissertation, University of California at Berkeley, 2006.

[142] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, and T. Anderson. iPlane: An information plane for distributed services. In *Proc. USENIX OSDI*, 2006.

[143] V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: Path prediction for peer-to-peer applications. In *Proc. USENIX NSDI*, 2009.

[144] B. Mah. pchar: A tool for measuring Internet path characteristics. http://www.kitchenlab.org/∼bmah/Software/pchar/.

[145] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet path diagnosis. In *Proc. ACM SOSP*, 2003.

[146] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *Proc. ACM SIGCOMM*, 2002.

[147] R. Mahajan, M. Zhang, L. Poole, and V. Pai. Uncovering performance differences among backbone ISPs with Netdiff. In *Proc. USENIX NSDI*, 2008.

[148] J. Mahdavi and V. Paxson. IPPM metrics for measuring connectivity. RFC 2678, IETF, September 1999.

[149] C. Man, G. Hasegawa, and M. Murata. Available bandwidth measurement via TCP connection. In *Proc. E2EMON*, 2004.

[150] C. Man, G. Hasegawa, and M. Murata. ImTCP highspeed: Inline network measurement for high-speed networks. In *Proc. PAM*, 2006.

[151] A. Markopoulou, F. Tobagi, and M. Karam. Loss and delay measurements of Internet backbones. *Computer Communications*, 29(10):1590–1604, 2006.

[152] M. Mathis and M. Allman. A framework for defining empirical bulk transfer capacity metrics. RFC 3148, IETF, July 2001.

[153] M. Mathis and J. Mahdavi. Diagnosing Internet congestion with a transport layer performance tool. In *Proc. INET*, 1996.

[154] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Proc. IEEE GLOBECOM*, 2000.

[155] J. Mogul. Observing TCP dynamics in real networks. In *Proc. ACM SIGCOMM*, 1992.

[156] S. Moon, J. Kurose, and D. Towsley. Correlation of packet delay and loss in the Internet. Technical report, University of Massachusetts, Amherst, 1998.

[157] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser. Packet reordering metrics. RFC 4737, IETF, November 2006.

[158] M. Neginhal, K. Harfoush, and H. Perros. Measuring bandwidth signatures of network paths. In *Proc. IFIP Networking*, 2007.

[159] R. Nelson. *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modelling*. Springer, 1995.

[160] T. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proc. IEEE INFOCOM*, 2002.

[161] H. Nguyen and M. Roughan. On the correlation of Internet packet losses. In *Proc. ATNAC*, 2008.

[162] L. Nussbaum and O. Richard. A comparative study of network link emulators. In *Proc. CNS*, 2009.

[163] A. Pásztor and D. Veitch. Active probing using packet quartets. In *Proc. ACM SIGCOMM IMW*, 2002.

[164] A. Pásztor and D. Veitch. The packet size dependence of packet-pair like methods. In *Proc. IWQoS*, 2002.

[165] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. ACM SIGCOMM*, 1998.

[166] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proc. ACM NOSSDAV*, 2002.

[167] P. Papageorge, J. McCann, and M. Hicks. Passive aggressive measurement with MGRP. In *Proc. ACM SIGCOMM*, 2009.

[168] K. Papagiannaki, R. Cruz, and C. Diot. Network performance monitoring at small time scales. In *Proc. ACM/USENIX IMC*, 2003.

[169] K. Park and V. Pai. Deploying large file transfer on an HTTP content distribution network. In *Proc. USENIX WORLDS*, 2004.

[170] K. Park and V. Pai. Scale and performance in the CoBlitz large-file distribution service. In *Proc. USENIX NSDI*, 2006.

[171] C. Partridge, J. Bennett, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Trans. Netw.*, 7(6):789–798, 1999.

[172] A. Pathak, H. Pucha, Y. Zhang, Y. Hu, and Z. Mao. A measurement study of Internet delay asymmetry. In *Proc. PAM*, 2008.

[173] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Trans. Netw.*, 5(5):601–615, 1997.

[174] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD dissertation, University of California Berkeley, 1997.

[175] V. Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Trans. Netw.*, 7(3):277–292, 1999.

[176] V. Paxson. Strategies for sound Internet measurement. In *Proc. ACM/USENIX IMC*, 2004.

[177] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for IP performance metrics. RFC 2330, IETF, May 1998.

[178] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An architecture for large-scale Internet measurement. *IEEE Communications Mag.*, 36(8):48–54, 1998.

[179] D. Pezaros, D. Hutchison, R. Gardner, F. Garcia, and J. Sventek. In-line measurements: A native measurement technique for IPv6 networks. In *Proc. INCC*, 2004.

[180] A. Pietro, D. Ficara, S. Giordano, F. Oppedisano, and G. Procissi. PingPair: A lightweight tool for measurement noise free path capacity estimation. In *Proc. IEEE ICC*, 2008.

[181] J. Poskanzer. mini_httpd: small HTTP server. http://www.acme.com/software/mini_httpd/.

[182] J. Postel. Internet Control Message Protocol. RFC 792, IETF, September 1981.

[183] J. Postel. Transmission control protocol. RFC 793, IETF, Sep. 1981.

[184] R. Prasad, C. Dovrolis, and B. Mah. The effect of layer-2 switches on pathchar-like tools. In *Proc. ACM SIGCOMM IMW*, 2002.

[185] R. Prasad, C. Dovrolis, and B. Mah. The effect of layer-2 store-and-forward devices on per-hop capacity estimation. In *Proc. IEEE INFOCOM*, 2003.

[186] R. Prasad, M. Murray, C. Dovrolis, and k. claffy. Bandwidth estimation: Metrics, measurement techniques, and tools. *IEEE Network*, 17(6):27–35, 2003.

[187] H. Pucha, Y. Zhang, Z. Mao, and Y. Hu. Understanding network delay changes caused by routing events. In *Proc. ACM SIGMETRICS*, 2007.

[188] D. Qiao, S. Choi, A. Jain, and K. Shin. MiSer: An optimal low-energy transmission strategy for IEEE 802.11a/h. In *Proc. ACM MOBICOM*, 2003.

[189] P. Ribeiro and V. Leung. Minimum delay path selection in multihomed systems with path asymmetry. *IEEE Communications Letters*, 10(3):135–147, 2006.

[190] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, and R. Baraniuk. Multifractal cross-traffic estimation. In *Proc. ITC Specialist Seminar on IP Traffic Measurement, Modeling, and Management*, 2000.

[191] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient available bandwidth estimation for network paths. In *Proc. PAM*, 2003.

[192] A. Rocha, R. Leão, and E. Silva. A non-cooperative active measurement technique for estimating the average and variance of the one-way delay. In *Proc. IFIP Networking*, 2007.

[193] M. Roughan. Fundamental bounds on the accuracy of network performance measurements. In *Proc. ACM SIGMETRICS*, 2005.

[194] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Trans. Netw.*, 10(3):381–395, 2002.

[195] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy. An analysis of Internet content delivery systems. *SIGOPS Oper. Syst. Rev.*, 36:315–327, 2002.

[196] S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. MMCN*, 2002.

[197] S. Saroiu, P. Gummadi, and S. Gribble. SProbe: A fast technique for measuring bottleneck bandwidth in uncooperative environments. In *Proc. IEEE INFOCOM*, 2002.

[198] S. Savage. Sting: A TCP-based network measurement tool. In *Proc. USENIX Symp. Internet Tech. and Sys.*, 1999.

[199] R. Schoonderwoerd. Network Performance Measurement Tools - A Comprehensive Comparison. Master dissertation, Vrije Universiteit Amsterdam, November 2002.

[200] Y. Schwartz, Y. Shavitt, and U. Weinsberg. A measurement study of the origins of end-to-end delay variations. In *Proc. PAM*, 2010.

[201] P. Sharma, Z. Xu, S. Banerjee, and S. Lee. Estimating network proximity and latency. *SIGCOMM Comput. Commun. Rev.*, 36(3):39–50, 2006.

[202] R. Sherwood and N. Spring. Touring the Internet in a TCP sidecar. In *Proc. ACM/USENIX IMC*, 2006.

[203] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and k. claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *Proc. PAM*, 2005.

[204] C. Simpson. *Analysis of Passive End-to-End Network Performance Measurements*. PhD dissertation, Georgia Institute of Technology, 2006.

[205] R. Sinha, C. Papadopoulos, and J. Heidemann. Fingerprinting Internet paths using packet pair dispersion. Technical Report 06-876, University of Southern California, Computer Science Department, 2005.

[206] J. Sommers. *Caliberated Network Measurement*. PhD dissertation, University of Wisconsin-Madison, 2007.

[207] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving accuracy in end-to-end packet loss measurement. In *Proc. ACM SIGCOMM*, 2005.

[208] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and efficient SLA compliance monitoring. In *Proc. ACM SIGCOMM*, 2007.

[209] J. Sommers, P. Barford, N. Duffield, and A. Ron. A geometric approach to improving active packet loss measurement. *IEEE/ACM Trans. Netw.*, 16(2):307–320, 2008.

[210] J. Sommers, P. Barford, and W. Willinger. Laboratory-based calibration of available bandwidth estimation tools. *Microprocess. Microsyst.*, 31(4):222–235, 2007.

[211] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public Internet measurement facility. In *Proc. USENIX Symp. Internet Tech. and Sys.*, 2003.

[212] E. Stephan. IP performance metrics (IPPM) metrics registry. RFC 4148, IETF, Aug. 2005.

[213] K. Stordahl. Long-term broadband evolution - forecasts and impact of new technologies. *Telektronikk*, 104(3/4):4–17, 2008.

[214] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Proc. ACM/USENIX IMC*, 2003.

[215] L. Sun, G. Wade, B. Lines, and E. Ifeachor. Impact of packet loss location on perceived speech quality. In *Proc. IPTEL*, 2001.

[216] A. Tanenbaum. *Computer Networks*. Prentice Hall, 2002.

[217] M. Tariq, A. Dhamdhere, C. Dovrolis, and M. Ammar. Poisson versus periodic path probing (or, does PASTA matter?). In *Proc. ACM/USENIX IMC*, 2005.

[218] M. Tariq, M. Motiwala, N. Feamster, and M. Ammar. Detecting network neutrality violations with causal inference. In *Proc. ACM CoNEXT*, 2009.

[219] M. Toren. tcptraceroute. http://michael.toren.net/code/tcptraceroute/.

[220] D. Tran, K. Hua, and T. Do. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proc. IEEE INFOCOM*, 2003.

[221] T. Tsugawa, G. Hasegawa, and M. Murata. Implementation and evaluation of an inline network measurement algorithm and its application to TCP-based service. In *Proc. E2EMON*, 2006.

[222] H. Uijterwaal. A one-way packet duplication metric. RFC 5560, IETF, May 2009.

[223] S. Valcourt. *A Comparison of the Current State of DSL Technologies*, chapter 10, pages 163–172. John Wiley & Sons, 2005.

[224] F. Wang, M. Mao, J. Wang, L. Gao, and R. Bush. A measurement study on the impact of routing events on end-to-end Internet path performance. In *Proc. ACM SIGCOMM*, 2006.

[225] Y. Wang, C. Huang, J. Li, and K. Ross. Queen: Estimating packet loss rate between arbitrary Internet host. In *Proc. PAM*, 2009.

[226] Z. Wang, A. Zeitoun, and S. Jamin. Challenges and lessons learned in measuring path RTT for proximity-based applications. In *Proc. PAM*, 2003.

[227] W. Wei, B. Wang, D. Towsley, and J. Kurose. Model-based identification of dominant congested links. In *Proc. ACM/USENIX IMC*, 2003.

[228] Z. Wen, S. Triukose, and M. Rabinovich. Facilitating focused Internet measurements. In *Proc. ACM SIGMETRICS*, 2007.

[229] L. Wenwei, Z. Dafang, Y. Jinmin, and X. Gaogang. On evaluating the differences of TCP and ICMP in network measurement. *Computer Communications*, 30(2):428–439, 2007.

[230] B. Wong, A. Slivkins, and E. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proc. ACM SIGCOMM*, 2005.

[231] M. Yajni, S. Moon, J. Kurose, and D. Towsley. Measurement and modelling of the temporal dependence in packet loss. In *Proc. IEEE INFOCOM*, 1999.

[232] P. Yalagandula, S. Lee, P. Sharma, and S. Banerjee. Correlations in end-to-end network metrics: Impact on large scale network monitoring. In *Proc. IEEE Global Internet Symposium*, 2008.

[233] G. Yan, D. Chiu, and J. Lui. Determining the end-to-end throughput capacity in multi-hop networks: Methodology and applications. In *Proc. ACM SIGMETRICS/Performance*, 2006.

[234] Y. Zhang and N. Duffield. On the constancy of Internet path properties. In *Proc. ACM SIGCOMM IMW*, 2001.

[235] Y. Zhang, Z. Mao, and M. Zhang. Detecting traffic differentiation in backbone ISPs with NetPolice. In *Proc. ACM/USENIX IMC*, 2009.

[236] Z. Zou, B. Lee, C. Fu, and J. Song. Packet Triplet: A novel approach to estimate path capacity. *IEEE Communications Letters*, 9(12):1076–1078, 2005.

[237] T. Zseby. Deployment of sampling methods for SLA validation with non-intrusive measurements. In *Proc. PAM*, 2001.