UNIVERSITE NICE SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour l'obtention du grade de

Docteur en Sciences

de l'Université Nice Sophia Antipolis

Mention : Informatique

présentée et soutenue par

Riccardo RAVAIOLI

ACTIVE INFERENCE OF NETWORK NEUTRALITY

Thèse dirigée par Guillaume URVOY-KELLER
et Chadi BARAKAT

soutenue le *13 juillet 2016*

Jury :

Prof. Françoise Baude	Président
Prof. Kavé SALAMATIAN	Rapporteur
Ass. Prof. Marco MELLIA	Rapporteur
Ass. Prof. Timur FRIEDMAN	Examinateur
Prof. Guillaume URVOY-KELLER	Directeur de thèse
Dr. Chadi BARAKAT (HDR)	Directeur de thèse

Abstract

In the last decade, some ISPs have been reported to discriminate against specific user traffic, especially if generated by bandwidth-hungry applications (e.g., peer-to-peer, video streaming) or competing services (e.g. Voice-over-IP). Network neutrality, a design principle according to which a network should treat all incoming packets equally, h as been widely debated ever since. In this thesis we present ChkDiff, a novel tool for the detection of traffic differentiation at the Internet access. In contrast to existing work, our method is agnostic to both the applications being tested and the shaping mechanisms deployed by an ISP. The experiment comprises two parts, in which we check for differentiation separately on upstream and downstream traffic that we previously dump directly from the user. In the upstream direction, ChkDiff replays the user's outgoing traffic with a modified TTL value in order to check for differentiation on routers at the first few hops from the user. By comparing the resulting delays and losses of flows that traversed the same routers, and analyzing the behaviour on the immediate router topology spawning from the user end point, we manage to detect instances of traffic shaping and attempt to localize shapers. Our study on the responsiveness of routers to TTL-limited probes consolidates our choice of measurements in the upstream experiment. In the downstream experiment, we replay the user's incoming traffic from a measurement server and analyze per-flow one-way delays and losses, while taking into account the possibility of multiple paths between the two endpoints. Along the chapters of this thesis, we provide a detailed description of our methodology and a validation of our tool.

Résumé

Durant la dernière décennie, des FAI ont été accusés de discriminer certains types de trafic utilisateur générés par des applications gourmandes en bande passante (peer-to-peer, streaming vidéo) ou par des services concurrents (Voice-over-IP). La neutralité des réseaux, un principe selon lequel un réseau devrait traiter tous les paquets entrants de la même manière, a été largement débattue. Dans cette thèse, nous présentons ChkDiff, un nouvel outil pour la détection de la différentiation du trafic dans le réseau d'accès. Contrairement aux travaux existants, notre méthode est agnostique à la fois vis-à-vis des applications testées et des mécanismes de shaping déployés. ChkDiff comprend deux phases dans lesquelles nous testons séparément le trafic montant et descendant capturé auparavant dans la machine de l'utilisateur. Dans la direction montante, ChkDiff rejoue le trafic sortant de la machine de l'utilisateur avec une valeur TTL modifiée afin de pouvoir tester les routeurs aux premiers sauts. En comparant les délais et les pertes des paquets des flux qui ont traversé les mêmes routeurs et en analysant les résultats sur la topologie des routeurs traversés, nous montrons que nous pouvons détecter les cas de différentiation et localiser les shapers. Notre étude sur la réactivité des routeurs aux sondes avec TTL limité consolide notre choix de mesures dans l'expérimentation sur le trafic montant. Dans la direction descendante, nous rejouons le trafic entrant dans la machine de l'utilisateur à partir d'un serveur de mesure et analysons pour chaque flux les délais unidirectionnels et les pertes, tout en tenant compte la possibilité de trajets multiples entre le serveur et l'utilisateur. Le long des chapitres de cette thèse, nous fournissons une description détaillée de notre méthodologie et une validation de notre outil.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Guillaume Urvoy-Keller and Chadi Barakat, for their constant guidance, tireless mentoring and invaluable insights throughout my years as a PhD student.

I would also like to thank my family for their precious unconditional support: without them, I would not be the person I am today.

This dissertation would not have been possible without funding from Labex UCN@Sophia.

Contents

List of Figures

List of Tables

1 Introduction

The increasing popularity of bandwidth-hungry applications, like peer-to-peer and video streaming, has induced some Internet Service Providers (ISPs) in the last decade to deploy some traffic management techniques that offer degraded performance instead of best-effort service to specific traffic flows. Reported cases abound: from blocking of BitTorrent traffic when a user is actively sharing files, to reduced performance of NetFlix by a few U.S. operators and throttling of YouTube during peak hours in the evening. Also competing services such as VoIP have been the target of ISPs, for instance when all Vonage calls were systematically blocked by a regional mobile operator and when Apple's FaceTime was disabled for mobile customers who did not opt for a more expensive data plan.

All these examples constitute clear violations of Internet neutrality, a principle according to which a network should treat all its incoming traffic equally, without deliberately offering worse or better performance to any traffic of its choice. There has been discussion[1] about whether the Internet has been conceived and implemented as a strictly level-playing field, but from a broader point of view it is generally agreed upon that all attempts that selectively deteriorate certain types of Internet traffic over others are, to say the least, controversial. Because of this, legislative efforts aiming at prohibiting cases like the above ones have appeared in a number of countries, the first of which to approve such laws were Chile in 2010 and the Netherlands in 2012.

Traffic differentiation is typically carried out in two steps: first a flow is identified by inspecting packet fields at the IP, transport or application layer, then a differentiation technique is applied to packets belonging to that flow, generally leading to the corresponding user application experiencing larger delays and more losses (translating into a lower throughput), when it is not blocked all together.

In the literature several tools have been described in recent years to try to establish whether an ISP applies traffic differentiation to specific applications (e.g., BitTorrent, YouTube, Skype, etc.)[2] and with the use of specific differentiation techniques (e.g., port blocking, token-bucket shaper, PowerBoost, etc.).[3]

In this thesis, we propose novel method for the detection of traffic differentiation at the access ISP that differs from existing work in

[1] Jon Crowcroft. Net neutrality: the technical side of the debate: a white paper. *SIGCOMM Comput. Commun. Rev.*, 37:49–56, January 2007

[2] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. Detecting BitTorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC'08)*, Vouliagmeni, Greece, October 2008

Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna Gummadi, Ratul Mahajan, and Stefan Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, Apr 2010

Partha Kanuparthy and Constantine Dovrolis. Diffprobe: detecting ISP service discrimination. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 1649–1657, Piscataway, NJ, USA, 2010. IEEE Press

[3] Robert Beverly, Steven Bauer, and Arthur Berger. The Internet is not a big truck: toward quantifying network neutrality. In *Passive and Active Network Measurement*, pages 135–144. Springer, 2007

Partha Kanuparthy and Constantine Dovrolis. Shaperprobe: End-to-end detection of isp traffic shaping using active methods. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 473–482. ACM, 2011

Udi Weinsberg, Augustin Soule, and Laurent Massoulié. Inferring traffic shaping and policy parameters using end host measurements. In *INFOCOM*, pages 151–155, 2011

that it is both independent from the applications tested by the user and the shaping technique deployed by the network operator. While most methods in the literature replay an application flow and a reference flow with a variety of techniques and compare the performance of the two flows to infer differentiation, we replay the whole user traffic with minimal modifications and compare the performance of each application run by the user against the rest of the user traffic, which we take as baseline. We detect differentiation by analyzing the distribution of delays and the number of losses of each flow, which are the two metrics that are directly affected by a shaper that wants to degrade the performance of selected traffic.

We implemented this in a tool we called ChkDiff,[4] which first dumps regular user traffic for a time window of a few minutes, during which the user is expected to run the Internet applications she intends to test, and then spawns two different experiments that replay the captured traffic in respectively the upstream and downstream direction. In both cases, we shuffle the trace in such a way that the position of the packets of each flow inside the trace can be modeled as a Poisson process, which guarantees us - according to the PASTA[5] property - that each replayed flow will see the same network conditions. Clearly, we preserve the order of packets of each flow while shuffling.

[4] The code is available on a dedicated web page: `http://chkdiff.gforge.inria.fr/`

[5] Poisson Arrivals See Time Averages.

In the upstream experiment,[6] we replay the user outgoing traffic against the routers in her immediate vicinity in order to test whether the user's access ISP deploys any shapers targeting any of the previously executed applications. For each hop at distance k to the user, we re-inject the trace into the network with a forged Time-To-Live (TTL) field in the IP header of each packet, so that routers at hop k will generate ICMP time-exceeded error messages, with which we compute per-flow Round-Trip Times (RTTs) and losses. ChkDiff infers that a flow has been differentiated when its empirical delay distribution fails the one-sided two-sample Kolmogorov-Smirnov test against the delays of all other flows in the trace, or when its loss count is not compatible with the overall loss rate of the rest of the trace, through a tailored binomial-inspired test. By analyzing the performance of flows across successive hops, we are also able to pinpoint the position of a shaper with respect to the user.

[6] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Towards a general solution for detecting traffic differentiation at the internet access. In *Teletraffic Congress (ITC 27), 2015 27th International*, pages 1–9. IEEE, 2015

In order to justify our choice of measurements in the upstream experiment, we performed a detailed study of the responsiveness of routers to TTL-limited probes and a characterization of how ICMP rate limitation is implemented.[7] With a large-scale measurement campaign from 180 PlanetLab nodes, we probed at increasing rates 850 routers up to hop 5 and analyzed their responsiveness. We found out that almost one third of the routers hit are fully responsive at least up to relatively high rates (2500 packets per second (*pps*)), a small fraction (3.9%) of routers is unresponsive and almost 60% implement ICMP rate limitation, whose most common form is an *on-off* pattern defined by the burst size of the generated packets and the inter-burst time, which we characterize in details. Also, most routers

[7] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Characterizing ICMP Rate Limitation on Routers. In *IEEE International Conference on Communications (ICC)*, 2015

are fully responsive at rates that are between one and two orders of magnitude higher than those commonly used by tools running these measurements. We also found no correlation between probing rates and the resulting RTTs, which means that our choice of probing rate does not reach any capacity limits on routers and does not cause any additional delay in the returned ICMP replies.

In the downstream experiment of ChkDiff,[8] we replay the user incoming traffic from a measurement server and measure per-flow one-way delays and losses in order to infer differentiation. Since there might be NATs, firewalls and middleboxes along the path between the user and our server, we take the necessary measures to open connections, find NAT mappings for the flows in the trace, adjust sequence numbers and avoid firewall timeouts. Unlike other proposals in the literature, we take into account the possibility of multiple paths between the two end points and base our analysis accordingly. Flow delays are grouped with a clustering algorithm that isolates *a posteriori* the different paths and flags as shaped those flows that could not be assigned to any discovered path. We integrate this delay analysis with the loss analysis that we also applied in the upstream case in order to detect traffic differentiation.

[8] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Testing for traffic differentiation with chkdiff: the downstream case. In *Teletraffic Congress (ITC 28), 2016 28th International*. IEEE, 2016

The thesis is organized as follows: in Chapter 2 we describe a few representative cases of traffic differentiation, introduce the concept of Internet neutrality and review existing methods for the detection of traffic differentiation; in Chapter 3 we study the responsiveness of routers to TTL-limited probes and characterize routers also according to their vendor; in Chapter 4 we provide a detailed description of the methodology of the upstream experiment in ChkDiff and validate the tool in a controlled setup with a wired and WiFi connection, under two different shaping scenarios, and also with a router implementing ICMP rate limitation; in Chapter 5 we illustrate the downstream experiment of ChkDiff and validate it using real Internet paths from three different server locations and over wired, WiFi and 3G connections; we give closing remarks and discuss future directions in Chapter 6.

2
Context and Related Work

Traffic management is commonplace in most network architectures: it is essential in case of congestion in the network and useful when certain managed services (e.g., voice over broadband, IPTV, etc.) have latency or jitter constraints. On the other hand, when traffic management is used to purposely discriminate certain types of traffic over others, it becomes a more controversial matter, as it directly harms the openness and neutrality of the Internet.

In this chapter we first present a few blatant cases of discriminatory practices performed by ISPs (Section 2.1), introduce the concept of Internet neutrality and offer arguments in favour and against it (Section 2.2); then, we explain how traffic differentiation is generally implemented in today's networks (Section 2.3) and summarize the legislative efforts in various countries to preserve a neutral and open Internet (Section 2.4). Finally, we describe and evaluate existing methods in the literature for the detection of traffic differentiation (Section 2.5).

2.1 A few examples of discriminatory practices

In the last decade, several cases of discriminatory practices have been reported by Internet users, concerning mostly bandwidth-hungry applications like peer-to-peer and video streaming, and competing services, like Voice-over-IP (VoIP) and even search engine queries. We detail here a few exemplary cases for each type of targeted service.

Peer-to-peer. One of the first cases that emerged is that of U.S. operator Comcast in 2005, when it deliberately blocked upstream traffic of BitTorrent users who were actively sharing files with end users located in other networks.[1] By injecting forged TCP RST packets when the transfer reached a certain threshold, Comcast was terminating uploads at the boundaries of its network, thus effectively reducing the cost of the extra connectivity that Comcast itself would have had to buy from backbone providers.

[1] Dslreports: comcast is using sandvine to manage p2p connections. URL: http://www.dslreports.com/forum/r18323368-Comcast-is-using-Sandvine-to-manage-P2P-Connections.

Video streaming. An ISP might also decide to target video streaming services, as for instance did U.S. wireless carrier MetroPCS in 2011,[2]

[2] MetroPCS 4G Data-Blocking Plans May Violate Net Neutrality. URL: http://www.wired.com/2011/01/metropcs-net-neutrality/

when it openly blocked all video streaming web sites except for YouTube and went as far as offering unlimited YouTube access, most probably following a special agreement between the two providers. Other times YouTube was negatively impacted by ISPs, as reported in 2011 by hundreds of fixed network users in France[3] and Germany[4]: the download rate was significantly throttled during peak hours in the evening, up to a point in which, for the case of French operator Free, resolutions of 720p and 1080p appeared to be blocked and only 480p seemed to work, even though at a lowered speed. A more subtle way to degrade performance was observed in 2014 with U.S. operators Comcast and Verizon,[5] which for months failed to upgrade their peering infrastructure despite a general increase in NetFlix traffic. This was the consequence of a dispute between the two ISPs and NetFlix and resulted in a worse video experience for NetFlix users. Another example of an ISP targeting video services is that of French mobile operator Bouygues,[6] which in 2011 was found to be intercepting users' HTTP requests for flash videos that exceeded 20 MB, to which it would respond with a 403 Forbidden HTTP error message; furthermore, all TCP connections exceeding 10 MB were terminated.

[3] Respect my net. URL: http://respectmynet.eu/view/205

[4] Respect my net. URL: http://respectmynet.eu/view/196

[5] Netflix performance on Verizon and Comcast has been dropping for months, 2013. URL: http://arstechnica.com/information-technology/2014/02/netflix-performance-on-verizon-and-comcast-has-been-dropping-for-months

[6] Bouygues télécom filtre malhonnêtement son réseau 3G et inspecte vos données. URL: http://grapsus.net/blog/post/Bouygues-Telecom-filtre-malhonnetement-son-reseau-3G-et-inspecte-vos-donnees

Voice-over-IP. Impairment of VoIP has been the objective of some ISPs that offer a similar voice service. For instance, in 2005 a regional ISP in the U.S. blocked all traffic of VoIP application Vonage;[7] uproar among customers urged Vonage to file a complaint to the Federal Communications Commission (FCC), which eventually condemned such practice. Between 2007 and 2009, AT&T obtained from Apple a complete block of all VoIP applications on the iPhone,[8] until in 2009 it finally allowed Skype, even if only when the device was connected to WiFi. More recently, in 2012 AT&T customers owning an iPhone had Apple's video-calling application FaceTime blocked, unless they opted for a more expensive data plan.[9] WhatsApp calls, Skype and Viber are currently blocked in the United Arab Emirates, where the telecommunications authority delegated to the two licensed telecom providers of the country the power to allow or not VoIP services through their networks.[10]

[7] Vonage says broadband provider blocks its calls. URL: http://www.cnet.com/news/vonage-says-broadband-provider-blocks-its-calls/

[8] Group asks FCC to probe iPhone Skype restrictions. URL: http://fortune.com/2009/04/03/group-asks-fcc-to-probe-iphone-skype-restrictions/

[9] AT&T blocking iPhone's FaceTime app would harm consumers and break net neutrality rules. URL: http://www.freepress.net/press-release/99480/att-blocking-iphones-facetime-app-would-harm-consumers-and-break-net-neutrality

[10] WhatsApp Voice Calling Already Banned by UAE's Etisalat: Report. 2015. URL: http://gadgets.ndtv.com/apps/news/whatsapp-voice-calling-already-banned-by-uaes-etisalat-report-672283

Redirection of search queries. An ISP could also have its own search engine and might want to redirect customers to it. This is what Windstream Communications, a U.S. DSL provider, did in 2010 when it was found to redirect to its own search engine all queries that were made using the Google toolbar in web browser Firefox.[11] By using Deep Packet Inspection (DPI), the ISP was able to intercept users' search requests to Google, prevent them from reaching the actual destination, and provide its customers with the result of its own search engine to the same query. In 2012 a dozen U.S. ISPs were reported to hijack queries that included specific keywords and redirect them through affiliate networks, in order monetize on those particular searches.[12]

[11] Phone Company Helps Make the Case for Net Neutrality. URL: http://www.savetheinternet.com/blog/10/04/05/phone-company-helps-make-case-net-neutrality

[12] Widespread Hijacking of Search Traffic in the United States. URL: https://www.eff.org/deeplinks/2011/07/widespread-search-hijacking-in-the-us

VPN's. VPN's might have their speed throttled too, regardless of the traffic they carry (since it is encrypted). This was the case in 2014 of Comcast subscribers who wanted to run OpenVPN with its default configuration (UDP traffic over port 1194) and experienced a strangely low speed, which doubled as soon as they switched to TCP and a different port number.[13]

[13] I just doubled my PIA VPN throughput that I am getting on my router by switching from UDP:1194 to TCP:443. URL: http://www.reddit.com/r/VPN/comments/1xkbca/i_just_doubled_my_pia_vpn_throughput_that_i_am

2.2 *Internet Neutrality*

The cases described in Section 2.1 are clear examples of ways an Internet service provider can tamper with user traffic and discriminate certain applications. This is against the end-to-end principle in general and network neutrality in particular. The end-to-end design principle of network communications[14] states that all application-specific functions should be implemented on end hosts and intermediate nodes are exclusively in charge of forwarding packets and delivering them to the destination. Internet neutrality, a term first coined by Tim Wu[15] in 2003 but a long-standing implicit principle, goes one step further:[16]

[14] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end Arguments in System Design. *ACM Trans. Comput. Syst.*, 2(4):277–288, November 1984

[15] Tim Wu. Network neutrality, broadband discrimination. *Journal of Telecommunications and high Technology law*, 2:141, 2003

[16] http://www.timwu.org/network_neutrality.html

> "...a maximally useful public information network aspires to treat all content, sites, and platforms equally. This allows the network to carry every form of information and support every kind of application."
>
> *Tim Wu*

While there are different interpretations of this principle, from a radical one stating that all incoming packets should be treated equally by a network, to a looser one according to which similar applications should experience similar performance, what we are interested to verify in this thesis is that no traffic flow is offered significantly degraded performance compared to the rest of the user traffic.

2.2.1 *Arguments in favour*

A neutral Internet, as advocated for example by non-profit organization Save The Internet[17], would foster competition and innovation in Internet contents, since the success or failure of online companies has always been dictated by the quality of the services they offer and not by special deals with ISPs. The risk of a non-neutral Internet is to switch from a model where ISPs simply provide the infrastructure of a "dumb" network, as according to the end-to-end principle, to a model where they play the role of gatekeepers, deciding which contents are granted a faster access and which are not. Indeed, with the exception of court orders, an ISP should not enforce any control over the data it forwards. From a less technical point of view, a non-neutral Internet would ultimately harm free speech and democratic participation in the Web,[18] thus undermining its original aspiration to offer a wide variety of independent sources of information and to generate innovative contents. Vint Cerf,[19] co-inventor of TCP/IP and

[17] www.savetheinternet.com

[18] Lawrence Lessig and Robert W. McChesney. No Tolls on The Internet. June 2006. URL: http://www.washingtonpost.com/wp-dyn/content/article/2006/06/07/AR2006060702108.html

[19] Vint Cerf speaks out on net neutrality. URL: https://googleblog.blogspot.fr/2005/11/vint-cerf-speaks-out-on-net-neutrality.html

an advocate of neutrality, also points out that with little competition among broadband providers, as it is the case in the U.S. market, a non-neutral Internet would put in the hands of a few the power of affecting what people do and see.

Tim Wu,[20] one of the most active scholars in the debate, claims that the Internet inherently aspires to neutrality, however imperfect that might be. According to Wu, the end-to-end principle, a key design idea of the Internet, is proof of this. He continues to observe that the diversity of applications running on the Internet, each one with different requirements, makes it difficult to defend a definition of neutrality regardless of applications; it is more logical to think of a neutral network as one that enforces equal treatment among similar applications.

[20] Tim Wu. Network neutrality, broadband discrimination. *Journal of Telecommunications and high Technology law*, 2:141, 2003

2.2.2 *Arguments against*

On the other hand, opponents of network neutrality claim that bandwidth prioritization is necessary for future innovation, as ISPs are more inclined to improve and modernize their networks if they have additional revenues, especially if coming from content providers like YouTube or Netflix that base their business on a massive use of network bandwidth and could be accused of free-riding the existing network infrastructure. Another argument against enforcing network neutrality rules is that throughout the history of the Internet, there has never been a true level-playing field, since large companies have always had an advantage over smaller ones by the deployment of replicating servers and the purchase of high-bandwidth services.

Jon Crowcroft further elaborates on this,[21] stating that historically the Internet has never been made of just "dumb pipes"; on the contrary, it comprises or depends already on a number of elements that directly alter its performance and accessibility: TCP throughput and its dependence on round trip time (and thus on the distance between a server and a user), making cache and proxy servers essential to give a significant advantage to companies that deploy them; inter-domain routing algorithm BGP natively supporting routing discrimination; NATs and firewalls effectively cutting out portions of the Internet. Moreover, services like VoIP, IPTV, videoconferencing and online games come with expectations in performance and sometimes with a cost for users, in which case it is the users themselves that would demand that such services be prioritized.

[21] Jon Crowcroft. Net neutrality: the technical side of the debate: a white paper. *SIGCOMM Comput. Commun. Rev.*, 37:49–56, January 2007

Jon Crowcroft. Talk on Net Neutrality, December 2006. URL: `"http://www.cl.cam.ac.uk/~jac22/talks/neut.ppt.gz"`

Jon Crowcroft. Talk on Newt or Notrality, July 2011. URL: `"https://www.cl.cam.ac.uk/~jac22/talks/notrality.ppt"`

Pappas et al.[22] claim that a strong definition of network neutrality, in which all packets should be treated equally by a network, is not only unrealistic, but also not at all desirable: applications with different needs, having for instance latency or bandwidth constraints, naturally result in better Quality of Experience (QoE) for users if treated accordingly across the network. Rather then enforcing impractical rules on ISPs, it is more important to ask them for increased transparency about their traffic management practices.

[22] Christos Pappas, Katerina Argyraki, Stefan Bechtold, and Adrian Perrig. Transparency Instead of Neutrality. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, pages 22:1–22:7. ACM, 2015

Indeed - whatever the position - it is generally agreed that any

action taken by an ISP to discriminate traffic should be made public, so that users can make an informed decision and freely choose the provider that best suits their needs. Transparency is therefore what we also advocate here.

2.3 *Traffic differentiation*

In order to discriminate a flow, a shaper has to perform two steps, involving first the identification of the flow it wants to affect and then the application of the desired differentiation technique to it.

1. *Flow identification.* An incoming packet can be associated to a known application by inspecting its IP header (source and destination IP addresses), its transport protocol header (port numbers), or its application payload. In this last case, devices are said to apply Deep Packet Inspection (DPI). Additional criteria can be taken into account by an ISP, such as time of the day (for example during peak hours), network load (if the network is congested), or even user behaviour (in case of heavy bandwidth usage).

2. *Differentiation.* Once a packet has been identified as belonging to a certain application that is meant to receive different treatment than regular traffic, a variety of differentiation techniques are possible. Packets of a flow can be entirely dropped, as in the case of port blocking; they can be deprioritized in order to decrease the overall flow throughput; their rate can be shaped with token or leaky bucket-like techniques; they can be subject to a fixed or variable dropping rate; or they can be routed over a slower link. An ISP can also cause a lower throughput to a flow or block it by interfering with it directly at the transport and application layers: at the TCP layer, it can reduce the advertised window size by overwriting the corresponding header field, or it can inject a TCP RST packet in order to terminate the connection; at the application layer, it can for example intercept HTTP requests and inject HTTP error messages that effectively impair the download of the requested resource (as seen in Section 2.1). Still at the application layer, an ISP can redirect HTTP requests to servers of its choice.[23]

[23] This is a form of differentiation that does not necessarily alter the performance of a flow, but modifies the contents requested by a user without the user's consent.

While all these techniques can overlap with traffic management practices, they constitute violations of Internet neutrality when they are used to purposely offer worse performance to selected flows.

The First-Come-First-Serve (FCFS) service policy might not yield the best results in case of contention of network resources with certain types of traffic, for example for network control and management. This is why traffic differentiation has been integrated into Internet protocols right from the beginning, with the Type of Service (ToS)[24] field of the IP header originally allowing applications to indicate the quality of service desired along the network path. The ToS field was later redefined as Differentiated Services Code Points (DCSP),[25] which allowed to specify the requested per-hop behaviour.

[24] P. Almquist. RFC 1349: Type of Service in the Internet Protocol Suite. 1992. URL: `https://tools.ietf.org/html/rfc1349`

[25] S. Blake, F. Baker, and D. Black. RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. 1998. URL: `https://tools.ietf.org/html/rfc2474`

While today's end-user applications do not use this field anymore, it is still extensively used by ISPs to mark incoming packets and map them to different internal routes before forwarding them to an external network. It is worth noting, however, that there has never been a homogeneous end-to-end usage of the DSCP field across different network operators, for engineering and economic reasons.

More generally, every network architecture today includes ways of differentiating traffic,[26] especially to handle congestion events, but also to limit a user's bandwidth to their purchased rate, to offer better performance to IP-based services offered by the ISP (e.g., IPTV and carrier grade voice) and to comply with the QoS parameters of Service Level Agreements (SLAs) for business connectivity services.

[26] BITAG Technical Working Group. Differentiated Treatment of Internet Traffic. 2015

2.4 *Legislative efforts*

Historically, the idea of neutrality in a communication network came about in 1860 with a US federal law stating that "messages from any individual, company, or corporation, or from any telegraph lines" should be "impartially transmitted in the order of their reception, excepting that the dispatches of the government shall have priority".[27] Nevertheless, it is only in the last dozen years that the idea of guaranteeing neutrality inside the Internet drew considerable attention from the media, the research community (for the technical, economical and legislative aspects it entails) and also law makers.

[27] http://cprr.org/Museum/Pacific_Telegraph_Act_1860.html

The first countries to approve laws that explicitly enforce network neutrality in the Internet were Chile[28] in 2010 and the Netherlands[29] in 2012. In the United States, after a 5-year-long debate,[30] network neutrality rules were finally approved in February 2015, prohibiting arbitrary blocking and throttling of services, and increasing transparency at the access ISP and in backbone networks.[31] In France, following a national Telecoms Package of 2009, the National Regulatory Authority ARCEP[32] was assigned increased powers in the monitoring of network neutrality and issued a series of recommendations for ISPs in 2010 and 2012.

[28] Chile, primer país en incorporar la neutralidad en la red. URL: http://www.elmundo.es/elmundo/2010/07/16/navegante/1279272468.html

[29] Net neutrality enshrined in dutch law. URL: http://www.theguardian.com/technology/2011/jun/23/netherlands-enshrines-net-neutrality-law

[30] https://www.whitehouse.gov/net-neutrality

[31] FCC. Protecting and promoting the open internet. April 2015. URL: https://www.federalregister.gov/articles/2015/04/13/2015-07841/protecting-and-promoting-theopen-internet

[32] Autorité de régulation des communications électroniques et des postes. http://www.arcep.fr/

In the European Union, a review of the European Telecoms Package in 2009 introduced the first provisions for increased transparency and started discussions about more comprehensive regulations, which were approved in October 2015 and have been effective since 30th April 2016. Thanks to this, the European legislation now (i) forbids any discriminatory treatment of Internet traffic and guarantees the right of all users to access and distribute contents of their choice; (ii) limits the circumstances under which an ISP can apply restrictive traffic management policies to its network (for example in case of congestion, exceptional security threats or specific court orders); (iii) states explicitly that optimized services should not degrade the performance of regular traffic; (iv) calls for monitoring of the commercial practices of ISPs and (v) strengthens the transparency obligations of ISPs, especially pertaining traffic management. The text mandates that the National Regulatory Authorities (NRAs) should

be in charge of enforcing these rules.

However, currently there is no general consensus on how NRAs should verify that ISPs do not apply any discriminatory practices to user traffic. For example, a NRA could decide to adopt a reactive approach, by expecting users to file reports and act upon them, or to have a proactive approach, in which measurements should be performed on each ISP operating on the national territory. NRAs also face the problem of selecting the tools to run such measurements. It has been observed[33] that NRAs inside the European Union have adopted a variety of measurement tools,[34] most of which simply focus on network troubleshooting to conduct their verifications, and among the tools proposed by the research community (which we review in details in Section 2.5) only one, NANO, has been adopted by a handful of European NRAs. Nonetheless, the Body of European Regulators of Electronic Communications (BEREC)[35] has been releasing a number of documents providing NRAs with recommendations and best practices on how to enforce network neutrality and when to intervene. A framework for the evaluation of Quality of Service (QoS) has been discussed, as well as when and how minimum QoS requirements should be set.[36] Also, guidelines have been provided on a few key problems:[37] (i) what the regulatory issues are with regard to QoS and Internet neutrality; (ii) which instances of service degradation require further investigation by NRAs and (iii) when regulatory intervention is necessary. More recently, a BEREC report also discussed [38] the requirements for network measurements to have legal value and described possible measurement methodologies, which could be hardware or software based, targeting an ISP or going up to Internet Exchange Points (IXP's), using active or passive techniques and applicable to fixed, wireless and mobile setups. The ultimate goal is to reach a convergence of metrics and methods across European countries, so as to have a uniform framework and allow cross-country measurements.

[33] Ioannis Koukoutsidis. Public QoS and net neutrality measurements. *Journal of Information*, 5, 2015

[34] BEREC. Annex of monitoring quality of internet access services in the context of net neutrality. *BEREC Report BoR (14) 117*, September 2014. URL: `http://berec.europa.eu/eng/document_register/subject_matter/berec/download/1/4602-monitoring-quality-of-internet-access-se_1.pdf`

[35] `http://berec.europa.eu/`

[36] BEREC. A framework for Quality of Service in the scope of Net Neutrality. *BEREC Report BoR (11) 53*, December 2011. URL: `http://berec.europa.eu/eng/document_register/subject_matter/berec/download/0/117-a-framework-for-quality-of-service-in-th_0.pdf`

[37] BEREC. Guidelines for quality of service in the scope of net neutrality. *BEREC Report BoR (12) 131*, November 2012. URL: `http://berec.europa.eu/eng/document_register/subject_matter/berec/download/0/1101-berec-guidelines-for-quality-of-service-_0.pdf`

2.5 *Tools for the detection of traffic differentiation*

A number of methods for the detection of traffic differentiation have been proposed in recent years in the literature. When performing measurements, one can do it passively, by collecting information from existing user traffic, or actively, by injecting traffic on the network and carrying out measurements on it. Among the proposed tools that detect traffic differentiation, NANO (Section 2.5.6) is the only one using passive measurements, while all other ones use active probing. Among these, relative discrimination is the most common technique: an application flow mimicking the behaviour of an application to test and a reference (or baseline) flow that is not supposed to experience differentiation are replayed between a user and a measurement server. By comparing the performance of the two flows, one can infer whether the application flow has been differentiated. This is the technique adopted, with various modifications in

their methodology, in BT-Test (Section 2.5.1), Glasnost (Section 2.5.2), DiffProbe (Section 2.5.3), ShaperProbe (Section 2.5.4), Packsen (Section 2.5.5), NetPolice (Section 2.5.8) and Differentiation Detector (Section 2.5.7), which is also the only tool that is implemented specifically for mobile networks. Zhang et al. (Section 2.5.9) on the other hand proposed a model that show when it is feasible to observe and localize violations of network neutrality. Finally, Neubot (Section 2.5.10) does not directly infer neutrality violations, but routinely collects measurements from end users in order to have a global picture to monitor over time.

Table 2.1 summarizes the key aspects of each methodology, which we assess globally in Section 2.5.11.

2.5.1 BT-TEST

Reports on BitTorrent discrimination being performed at the time by Comcast in the United States were largely covered by Internet media.[39] Regular users, though, did not have tools to evaluate whether their ISP was differentiating BitTorrent traffic. The few existing network-troubleshooting tools required expert knowledge in order to be applied to this particular differentiation case.

[39] Packet forgery by ISPs: A report on the comcast affair. URL: `https://www.eff.org/wp/packet-forgery-isps-report-comcast-affair`

BT-Test[40] enables users to verify whether: (i) their ISP is blocking BitTorrent traffic using forged RST packets; (ii) their ISP is able to identify BitTorrent flows by looking at port numbers, BitTorrent protocol messages, or both and (iii) differentiation affects uploads, downloads, or both.

[40] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. Detecting BitTorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC'08)*, Vouliagmeni, Greece, October 2008

The tool checks for RST packet injection and only tests BitTorrent flows.

Outline of the method. BT-Test replays eight different flows between the user and a measurement server. Flows vary in the TCP port number (6881, a well-known BitTorrent port, or 4711, not associated with any protocol) the direction (upstream or downstream) and the application payloads (BitTorrent-like or randomized but with the same size and ordering as in BitTorrent flows). Each of the eight resulting combinations is run twice and lasts 10 seconds. The server performs a complete link-level packet capture, thus avoiding the burden of requiring administrator privileges on the client side, where the user simply runs a Java applet. The client only keeps track of Java exceptions with error messages of interest.

Differentiation is detected when (i) the server-side packet trace contains at least one incoming RST packet and (ii) IO exceptions signaling a terminated connection are seen by the applet. By examining the results obtained in each test, the tool identifies how the ISP classifies flows as BitTorrent, that is to say whether the discrimination is based on the port number or on the protocol messages, and whether it affects upstream or downstream BitTorrent flows.

Results from tests in the wild. It was found that Comcast blocked uploads only when a user had finished the download of a file and uploaded it altruistically. As for flows identified as BitTorrent through inspection of message protocols, by looking at the flow trace on the server it was possible to determine at which point of the message exchange the ISP triggered differentiation. Interestingly, contrary to what some ISPs had claimed at the time, no differentiation during peak-hours was evident from the large sample of gathered tests.

User's point of view. The user starts the test trough a Java applet included in a dedicated web page, waits for the test to complete and then results are immediately displayed. The tool is not currently available, as it was superseded by Glasnost (Section 2.5.2).

2.5.2 *Glasnost*

With no tool at the time being suitable to be used by the large public, Glasnost[41] aims to be a widely-usable tool whose primary focus is non-savvy users.

[41] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna Gummadi, Ratul Mahajan, and Stefan Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, Apr 2010

The three main design principles are:

- Low barrier of use (simple and intuitive interface, no installation of additional software required, no administrative privileges needed, quick computation of results, results immediately displayed to the user).
- Accuracy of results and no space for misinterpretation (reducing the effects of confounding factors, minimizing false positives at the expense of false negatives and retaining positive results in case the tool is challenged to justify its findings).
- Easiness of evolution (extendibility of the system and avoiding white-listing).

Outline of the method. The tool emulates an application flow and a reference flow, of which only the latter should trigger differentiation, if any shaper is deployed along the path. The difference between the two flows is on port numbers and application payload, whereas all fields at the IP layer remain unchanged. Glasnost interleaves these two flows and sends them between the user and a measurement server 5 times. Each flow lasts 60 seconds to allow TCP to achieve a stable throughput. Since throughput in TCP is directly affected by any differentiation technique, that is what the tool compares between the two flows.

Two steps are involved in the analysis:

1. *Noise filtering*. In order to identify noisy paths, the tool analyzes the variance of throughput of the two flows. Noise is calculated as the difference between maximum and median throughput, divided over the maximum throughput. All tests affected by high

noise are discarded. From empirical tests, up to a 20% of difference between median and maximum throughput is to be expected in the absence of considerable noise.

2. *Differentiation detection.* The idea is that in scenarios with low noise, most throughput measurements lie close to the maximum throughput. Since noise tends to lower the throughput, the maximum throughput is a good approximation of the throughput that flows could achieve without cross traffic. In the first implementation of the tool, differentiation is inferred when the maximum throughput of the two flows differs by more than 20%. In such version of Glasnost the total duration of the test was 20 minutes, which proved to be too long for most users, who abandoned the test before its completion. The test duration was then reduced down to 6 minutes and the minimum difference between maximum flow throughputs for differentiation detection was increased to 50%, in order to minimize false positives.

Results from tests in the wild. The first test implemented in Glasnost checked whether ISPs differentiated BitTorrent flows based on content or port numbers, downstream or upstream. 21% of users were affected by differentiation and not all customers of the same ISP which performed differentiation were affected. Detailed results are available online, sorted by country and ISP.[42]

[42] http://broadband.mpi-sws.org/transparency/results/

User's point of view. A user needs to go on the Glasnost web page[43] (hosted by M-Lab[44]) and select an application flow to test. The test is carried out through a Java applet, for a total duration of 6 minutes, after which the results are displayed. Currently, available tests are: BitTorrent, eMule and Gnutella for peer-to-peer applications; flash video for video streaming applications; POP, IMAP4, HTTP transfer, SSH transfer and Usenet for more generic services.

Advanced users are provided with a tool[45] which automatically generates a new Glasnost test from the packet-level trace of an application. It extracts packet size, payload, order of packets exchanged and inter-packet timing. The output is used directly by the Glasnost applet. The application flow preserves ordering at the expense of timing, since it is assumed that ISPs usually identify applications by looking only at the sequence of protocol messages. The reference flow differs from the application flow in its payloads and port numbers. The user uploading the trace can set port numbers to specific values, otherwise they are randomly chosen.

[43] http://www.measurementlab.net/tools/glasnost/

[44] Measurement Lab is an open platform for researchers to deploy Internet measurement tools. URL: http://www.measurementlab.net

Constantine Dovrolis, Krishna Gummadi, Aleksandar Kuzmanovic, and Sascha D. Meinrath. Measurement lab: overview and an invitation to the research community. *SIGCOMM Comput. Commun. Rev.*, 40:53–56, June 2010

[45] http://broadband.mpi-sws.org/transparency/createtest.html

2.5.3 *DiffProbe*

Starting from the assumption that any form of discrimination results in high delays or high loss rates, the goal of DiffProbe[46] is to detect if:

- the traffic of an application is being classified as low-priority by an ISP;

[46] Partha Kanuparthy and Constantine Dovrolis. Diffprobe: detecting ISP service discrimination. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 1649–1657, Piscataway, NJ, USA, 2010. IEEE Press

- the ISP performs delay discrimination, loss discrimination, or both;
- the scheduler type, implementing Strict Priority (SP) or Weighted Fair Queueing (WFQ), can be identified.

When no differentiation is taking place, the tool assumes that ISPs use the First-Come-First-Served (FCFS) scheduling discipline and DropTail buffer management policy.

Outline of the method. At first, an estimation of upstream and downstream path capacities between client and measurement server is performed using trains of packets. Then, an application (A) and a reference (P) flow are sent simultaneously during two replay phases and one-way delays, as well as the number of lost packets, are measured for the two flows. Flow A uses a pre-recorded trace of Skype or Vonage and it keeps the same port numbers, transport protocol, packet sizes, inter-packet gaps and payloads as in the respective original application, while the last four payload Bytes are overwritten with the sender timestamp for easier one-way delay measurement. Flow P is crafted with the same number of packets and the same packet sizes as in flow A; payloads are randomized (excluding the sender's timestamp) and port numbers are those not commonly associated with any service, so that packets are unlikely to be classified as low priority. Packets of flow P are sent at about the same time as those of flow A.

The experiment comprises two phases, which DiffProbe performs downstream and upstream:

- Balanced Flow Period (BLP). Both flows are sent at their nominal rates.
- Load Increase Period (LIP). The rate of flow P is scaled up so as to maximize the chances of queueing in the ISP network. This way, flows A and P should saturate the user's access link.

The first comparison DiffProbe performs is between the 90th-percentile of delays of flow P during the LIP phase and the median delay of flow P during the BLP phase. If the former is not significantly larger than the latter, differentiation cannot be detected, since the experiment did not seem to trigger any queueing. The ISP might still deploy some kind of differentiation, but it has no significant effect on the user's traffic.

Delay discrimination detection. The tool first checks for equality of the delay distributions of flow A and flow P during the LIP phase. It selects only the packets that might have experienced queueing by considering, for each packet in flow A, the first packet in P sent no later than the transmission time of an MTU-sized packet in the bottleneck link. If no such sample in P exists, this packet of flow A is discarded. If it does exist, a pair containing the delay of the two samples (one for flow A, one for flow P) is added to the set of delays to

analyze. After removing from this set the delays that are too close to the propagation delay, estimated as the minimum delay of each flow, the propagation delay is subtracted from each sample, so that the resulting delay values will approximate the queueing delays experienced by the packets of the two flows. On these values, the tool runs the non-parametric Kullback-Leibler (KL) divergence test, which rejects the two input sets if they do not follow the same distribution. When this is the case, DiffProbe goes on to test for inequality of the two empirical distributions: it checks that all higher percentiles (between 50th and 95th) of the empirical delay distribution of flow A are larger than the corresponding ones of flow P. If this holds true, the tool reports that delay discrimination was applied to flow A.

Furthermore, DiffProbe attempts to identify whether the packet scheduling algorithm used for differentiation is Strict Priority (SP) or Weighted Fair Queueing (WFQ). In a SP scheduler, an arriving P packet will never experience queueing delays due to backlogged low-priority packets, while in a WFQ scheduler it could. Therefore, for each burst of P packets the tool now only considers the first packet, which is the only one that is most likely to *not* have been backlogged behind other high priority packets. In this subset of P packets, the tool examines the variability of the delay distribution: with SP scheduling it should be very small, whereas with WFQ it should be significantly higher.

Loss discrimination detection. DiffProbe also compares the losses of flows A and P in order to discover discriminatory buffer managers. To ensure that the two flows sample the same congestion events, the same pairing as in the delay analysis is performed. On the resulting samples, the tool runs the two-proportion z-test for equal population proportions, which will fail if the loss events in the two flows differ considerably.

Validation and results from tests. The tool was validated in a simulated environment. Tests in residential ISPs, located in the US and in Europe, did not show any differentiation.

User's point of view. The user chooses between a Skype and a Vonage UDP trace, each 10 minutes long, and launches the test from the command line. Results are shown upon completion of the test.[47]

[47] `http://netinfer.net/diffprobe/`

2.5.4 *ShaperProbe*

An extension of DiffProbe (Section 2.5.3), ShaperProbe[48] attempts to verify whether an ISP is shaping user traffic and, more specifically, if the user connection drops automatically to a low rate after some time, due to a shaper.

Traffic shaping is modeled as a token bucket, allowing a maximum burst of traffic to be serviced at the peak capacity of the link, while any remaining traffic is serviced at a lower rate (i.e. the shaping rate).

[48] Partha Kanuparthy and Constantine Dovrolis. Shaperprobe: End-to-end detection of isp traffic shaping using active methods. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 473–482. ACM, 2011

PowerBoost is an example of a shaper based on such mechanism.[49,50]

Outline of the method. The capacity of the path between user and server is estimated in both directions before starting the actual experiment: the sender sends trains of back-to-back packets and the receiver estimates the path capacity by measuring the dispersion of each train. Then the sender probes the receiver at a rate equal to the path capacity. The receiver records the received-rate timeseries by discretizing time into intervals of duration D and estimating the receiving rate for each interval as the amount of bits received over interval D. Probing stops when either the receiver sees a level shift in the timeseries or the duration of the probing exceeds 60 seconds. The experiment is carried out in upstream and downstream directions. In case a level shift is detected, shaping parameters are estimated:

- the shaping rate as the median rate after the level shift;
- the burst size based on the number of Bytes sent before the level shift.

It is worth pointing out that, since shaping mechanisms like PowerBoost affect all flows regardless of their origin, the replayed trace reproduces a single flow which does not belong to any specific application.

Validation and results from tests in the wild. In order to test the accuracy of the method, ShaperProbe was validated on two ISPs where the shaping ground truth was known and in an emulated wide-area setup. The measurement server used to be hosted on M-Lab. Results from M-Lab users showed that U.S. operators Comcast and Cox were indeed shaping traffic of a large fraction of their users, while evidence from other U.S. operators only involved between 5 and 10% of their subscribers who ran DiffProbe.

User's point of view. The user runs the tool from the command line. The experiment lasts for about 3 minutes, after which results are immediately displayed.[51]

Passive variant for TCP flows. A passive technique which uses an optimized version of ShaperProbe was also proposed. It takes as input an application packet trace and it is particularly useful whenever an ISP performs shaping on flows based on packet fields that are difficult to replicate with active probing to a server, such as destination and source IP addresses. In order for ShaperProbe to apply to a TCP flow the authors had to take into account that:

- a level shift in TCP throughput occurs every time TCP decreases its window due to timeouts or packet losses;
- TCP does not send a constant rate stream, making it difficult to estimate the number of tokens in the token bucket - indeed, tokens

[49] In the version deployed by Comcast in the U.S, it allows faster downloads up to 20 MB of data and faster uploads up to 10 MB, after which it decreases the speed to the regular provisioned one. Since it increases a user's contractual speed, it is actually a form of *positive* shaping. `http://www.dslreports.com/faq/14520`

[50] Srikanth Sundaresan, Walter De Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Measuring home broadband performance. *Communications of the ACM*, 55(11):100–109, 2012

[51] `http://netinfer.net/diffprobe/shaperprobe.html`

can accumulate when the TCP throughput is below the shaping rate;

- Accumulation of tokens might also happen when shaping kicks in, as it might trigger TCP back-offs and timeouts.

Therefore it is essential to be able to distinguish throughput level shifts due to a shaper from those due to TCP back-offs. This is done by analyzing the variability of a carefully constructed timeseries of cumulative Bytes received at the receiver.

This passive method was validated using a large set of NDT traces collected at M-Lab. Also, consistent results were found between active and passive methods by having the passive variant analyze ShaperProbe experiment traces.

2.5.5 *Packsen*

Existing tools either focused on simply detecting the existence of flow discrimination (BT-Test, Glasnost, NANO) or required relatively long calculations to obtain accurate results (DiffProbe). Packsen[52] detects traffic shaping and infers its parameters, with a faster inference and still high detection accuracy, even in the presence of cross traffic. As seen in other methods in the literature, it replays an application (BitTorrent) and a reference (HTTP) flow, and compares their performance.

[52] Udi Weinsberg, Augustin Soule, and Laurent Massoulié. Inferring traffic shaping and policy parameters using end host measurements. In *INFOCOM*, pages 151–155, 2011

Outline of the method. The experiment is made of up to three phases, aimed at respectively detecting shaping, inferring the shaper type and its parameters, and inferring WFQ shaper weights in case of significant cross traffic. The computational cost increases with each phase.

1. *Detection of shaping*. Two short interleaved flows of 20 packets each are sent to a measurement server. The tool then compares the distribution of inter-arrival times of the two flows with the Mann-Whitney U-test, which is accurate also on small samples. If the test rejects the two flows, the tool infers that shaping took place and moves to the next phase.

2. *Inference of shaper type and parameters.* The experiment is repeated, but with flows of 100 packets each. Packsen then compares the values observed for sent and received bandwidth; the former is assumed to estimate the sender's bandwidth, while the latter should estimate the bandwidth induced by the shaper. A Strict Priority (SP) scheduler is recognized when the sent bandwidth dominates the received bandwidth. If this does not happen, the tool assumes that the scheduler implements Weighted Fair Queueing (WFQ) and estimates the flow weights.

3. *Inference of WFQ weights with cross traffic.* In the presence of heavy cross traffic, phase 2 does not yield correct results for WFQ schedulers. The tool analyzes the arrival times of selected flow packets

from phase 2 and infers WFQ weights, while taking into account cross traffic, which it models as a Poisson process. The experiment in phase 2 is repeated several times until the variance in the results is sufficiently low.

Results from tests. Validation was carried out both in a controlled testbed with emulated cross traffic and with a large set of PlanetLab nodes.

User's point of view. The tool is not available for public use.

2.5.6 NANO

Existing tools for detection of traffic differentiation are typically specific to an application or focus on a particular differentiation mechanism. Also, their use of active probing makes it easy for an ISP to detect injected measurement flows and treat them accordingly, by ether blocking or prioritizing them. Nano[53] overcomes all this by performing passive measurements and using causal inference to quantify the causal relationship between an ISP and the observed service degradation.

[53] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. Detecting network neutrality violations with causal inference. *ACM SIGCOMM CoNext*, page 289, 2009

Causal inference. Let the action taken by an ISP be treatment variable X and let the observed service performance be outcome variable Y. In order to establish a causal relation between X and Y, all the factors that might generate interference should be identified. They could be: (i) client-based: application, operating system, computer configuration and the user's local network service contract; (ii) network-based: client and ISP location (with respect to the location of servers on the Internet), or a path segment to a particular service provider not sufficiently provisioned and (iii) time-based: time of the day, since service performance might vary widely throughout the day, both for ISPs and service providers. Once measurements are gathered, they are divided into strata so as to have in each stratum similar values for each confounding variable. Within a stratum NANO can easily compute the measure of the observed effect (the association value), on outcome variable Y (the service performance), and claim that it converges to causal effect. Therefore association can be used as a metric of causal effect within a stratum.

Outline of the method. Instead of attempting to detect a specific behaviour, NANO directly observes the performance of traffic and tries to infer the causes of degraded performance in case of its occurrence.

A NANO agent resides on an end host, collects traffic data from it and aggregates traffic statistics that it later sends to a centralized server. In particular, an agent collects features that help to identify the client setup, it determines the topological location of the client and their ISPs and it timestamps all collected data. These will be the confounding factors sent to the server. An agent analyzes IP,

transport and application protocol headers to identify the services run by the user and for each flow it measures throughput and jitter, and counts losses as well as possible incoming TCP RST packets.

A server receives data from the agents and applies the following actions to detect traffic differentiation:

1. *It stratifies the data.* Bins (ranges of values) are created for each confounding variable. Adjacent strata are combined if the distribution of outcome variable Y conditioned on the treatment variable X is identical in more strata.

2. *It establishes baseline and causality.* It takes as baseline the average performance of all other clients with similar values for all confounding variables except for the access ISP.

3. *It infers the discrimination criteria.* A decision tree is used to generate rules indicating the discrimination criteria that the ISP applies.

Validation. The validation setup included:

- agents running on clients deployed on Emulab hosts, so that confounding variables on the client side could be controlled.

- ISPs represented by shapers also running on Emulab hosts, in order to directly apply various discrimination techniques (probabilistic dropping of packets on all flows or on flows exceeding a certain length, dropping of TCP acknowledgements, dropping of packets for a particular service or destination, and TCP RST injection);

- Content providers on PlanetLab nodes, which ensured that experiments used real Internet paths;

- a server on a regular Internet host.

NANO correctly identified discriminating and neutral ISPs in all validation tests.

Due to the methodology used to infer differentiation, a deployment of NANO in the wild would require a large user base in order to correctly establish causal inference.

User's point of view. NANO agents are voluntarily installed by end users,[54] who are asked to manually provide details about their network interface (wired or wireless), the type of contract with their ISP and their geographic location. The project appears to have been discontinued.

[54] `https://noise-lab.net/projects/old-projects/nano/`

2.5.7 *Differentiation Detector*

Recently Differentiator Detector,[55] a tool for the detection of traffic differentiation in mobile networks, was proposed. It addresses the issue of scalability to arbitrary applications by fully replaying between the user and a measurement server a previously dumped flow. It also

[55] Arash Molavi Kakhki, Abbas Razaghpanah, Anke Li, Hyungjoon Koo, Rajesh Golani, David Choffnes, Phillipa Gill, and Alan Mislove. Identifying traffic differentiation in mobile networks. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 239–251. ACM, 2015

avoids all sorts of rooting or jailbreaking on a user's mobile phone by moving the task of capturing traffic to a server and by replaying flows right from the application layer.

Outline of the method. The experiment comprises two phases:

- *Dump.* At first, the user is asked to run on her Android mobile phone the application she intends to test after connecting to Meddle,[56] a VPN proxy over IPsec, which relays traffic between user and destination and dumps the selected application flow. The VPN proxy then produces a transcript describing the packet exchange that took place in downstream and upstream directions, and the corresponding inter-packet times. This transcript is then delivered to both the user and a measurement server, so that they can replicate the behaviour of the selected flow. In the current version of the tool, this phase is skipped and the user is provided with a set of pre-recorded traces to replay.
- *Replay.* User and measurement server replay the flow first as it is, while of course overwriting the original server-side IP address with the IP address of the measurement server, and then through an encrypted VPN. The idea is that a shaper that wants to affect a given application will be able to identify the corresponding flow when replayed out of the VPN, but not when the packets are encrypted. The replay phase is carried out twice to minimize the effects of possible noise. For TCP flows, throughput and Round-Trip Times (RTT's) are measured on the captured trace from the server side, while for UDP flows the client computes the flow jitter from the inter-packet timings at the application layer. Losses are recorded for all flows.

[56] Ashwin Rao, Justine Sherry, Arnaud Legout, Arvind Krishnamurthy, Walid Dabbous, and David Choffnes. Meddle: middleboxes for increased transparency and control of mobile traffic. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pages 65–66. ACM, 2012

In order to evaluate the performance of the application and the control flows, that is to say the flows replayed respectively outside and inside the VPN, the tool first runs the two-sample Kolmogorov-Smirnov test, which assesses the maximum vertical distance between two empirical CDF's. Then, the tool applies a tailored test that rejects the flows if the normalized area between their empirical CDF's is greater than a given threshold. The reason behind this is to take into account also the horizontal distance between the two curves before declaring that a flow has been differentiated. The analysis on flow losses simply compares maximum and average flow losses over the two replays.

Validation and results from tests in the wild. Differentiation Detector was validated with two commercial shapers, which correctly identified application flows and disregarded control flows. The authors of the tool were able to reverse engineer the flow classification mechanism of these two shapers and found out that flows get mostly identified as belonging to a particular application with the use of regular expressions on packet payloads (Deep Packet Inspection) and by looking at port numbers.

Tests from users in most major mobile providers in the U.S. showed cases of shaping for YouTube, NetFlix and Spotify in the first semester of 2015.

User's point of view. The user needs to install Differentiation Detector[57] from Google Play and then selects one application flow to test between YouTube, NetFlix, Spotify, Skype, Viber and Google Hangout. The outcome of the test is shown right after the experiment.

[57] `http://dd.meddle.mobi/index.html`

2.5.8 NetPolice

As opposed to all other methods so far described, which test a user's access ISP, NetPolice[58] aims at detecting traffic differentiation in backbone ISPs. It selectively probes from a number of hosts ingress and egress routers of backbone ISPs with TTL-limited packets and analyzes the loss rates observed along the same ingress-egress path segments. It is able to detect content differentiation based on packet payloads or port numbers and routing differentiation based on the next-hop and previous-hop autonomous system (AS).

[58] Ying Zhang, Zhuoqing Morley Mao, and Ming Zhang. Detecting traffic differentiation in backbone isps with netpolice. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 103–115. ACM, 2009

Outline of the method. The methodology used by NetPolice can be broken down into three steps:

1. *Path selection.* Given a set of hosts running NetPolice, the tool runs traceroute to all prefix destinations on the Internet in order to discover which source-destination pairs cross which ingress and egress routers of backbone ISPs along the path and at which hops such routers are found. Afterwards the tool selects the set of paths to test so that it minimizes the number of experiments per host while still having an exhaustive picture of an ISP ingress and egress points, probed from various sources and with various IP destination addresses.

2. *Path measurement.* Each host is instructed to test a given path a number of times and does so by sending packets with application contents and the specified TTL values so that they expire at the targeted ingress and egress ISP routers and generate ICMP time-exceeded error messages. Probing is performed at a constant rate of 1 packet per second (pps) for 200 seconds, so as to minimize losses due to routers implementing ICMP rate limitation. Sent packets are the same size (200 Bytes) and have port numbers and application payloads from 5 different applications: BitTorrent, SMTP, PPLive, VoiP and HTTP.

3. *Differentiation detection.* For the same pair of ingress and egress routers, NetPolice computes the per-experiment loss rate of this internal path by subtracting the ingress loss rate from the egress loss rate. Then, it verifies whether probes belonging to the five applications, coming from different sources and destined to different IP prefixes, experienced similar loss rates along this path segment. In order to detect content-based differentiation, it runs

the Kolmogorov-Smirnov (KS) test between the loss rate distribution of experiments carrying HTTP traffic (used as baseline) to that of experiments carrying any of the other four tested applications. Similarly, but regardless of the applications, NetPolice runs the KS test on the loss rate distribution of experiments traversing the same AS right before reaching the target ingress-egress pair, against the loss rate distribution of experiments that crossed a different AS. It then runs the KS test on the set of experiments routed to different AS's after the same ingress-egress path.

Results from tests. A measurement campaign was performed from 200 PlanetLab hosts, targeting 18 backbone ISPs. Content-based differentiation was found in 4 ISPs, while routing-based differentiation was found in 10; the observed difference in loss rates was up to 5%. Since only a fraction of paths in a given ISP indicated traffic differentiation, it was conjectured that a high network load on a subset of all internal paths of an ISP might trigger differentiation. By also analyzing the Type of Service (TOS) field in the quoted IP packet of returned ICMP time-exceeded messages, a correlation was occasionally found between the observed TOS value and the resulting loss rates.

User's point of view. NetPolice is not intended for end users.

2.5.9 *Neutrality Inference through Network Tomography*

Zhang et al.[59] recently proposed a theoretical tomography-inspired framework where they claim that, given the topology of a network and external end-to-end path measurements over its paths, such observations are inconsistent with each other when the network is non-neutral. The authors describe the conditions that are necessary to observe neutrality violations and to localize them on specific links, independently of how traffic differentiation is implemented.

[59] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network neutrality inference. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 63–74, New York, NY, USA, 2014. ACM

Since network tomography essentially assumes that a network is neutral and that a neutral link has the same properties for all paths traversing it, a neutral network can be represented by a solvable system of equations where the sum of unknown specific link properties of a given path equals a measured metric on that path. The insight of this work is that if such system is unsolvable, it indicates a possible neutrality violation.

Outline of the model. The proposed model can be split into three contributions:

1. *Observability of neutrality violations.* For any network where a neutrality violation is detected there exists at least one equivalent neutral network that, given the same traffic input, would cause the same external observations as in the original non-neutral network. This equivalent neutral network is constructed so that any non-neutral link in the original non-neutral network is split into

two links: (i) a regular link that retains the same properties that originally applied to non-differentiated traffic, and (ii) an extra link, called virtual link, inserted in such a way in the network topology that only differentiated traffic flows through it and has the same properties originally applied to differentiated traffic. The model claims that a neutrality violation is externally observable if and only if the equivalent neutral network contains at least one virtual link that is distinguishable from any link in the original non-neutral network. That is to say, if there is at least one virtual link whose set of paths traversing it differs from the sets of paths traversing all other links in the original network.

2. *Localization of neutrality violations.* Once a network has been detected to be non-neutral, the non-neutral links can be pinpointed to a link sequence of the original network, as long as this link sequence is crossed by a sufficiently diverse number of paths. A system of equations is constructed as described above, but only over the network slice containing the suspected link sequence. If this system is unsolvable, then the link sequence is non-neutral.

3. *Algorithm.* Zhang et al. finally proposed an algorithm that, given as input the topology of a network and the results of measurements over its paths, it outputs any non-neutral link sequences. It first checks for observable neutrality violations and then attempts to localize them by verifying, for each link sequence, the unsolvability of the system of equations composed of pairs of paths that intersect exactly at this link sequence.

Validation. The method was validated in an emulated environment, first in a simpler case with only one shared link, where the shaper was deployed, and then in a more complex case with several shared links. The algorithm proved to produce no false negatives or false positives in both scenarios.

User's point of view. It is not intended for end users.

2.5.10 Neubot

A project that differs from the ones thus far described is Neubot[60], which monitors the access link of users by regularly running distributed performance measurements and collects the data on a centralized server, where results are made available to researchers for later use. The tool does not explicitly infer neutrality violations.

[60] Simone Basso, Antonio Servetti, and Juan Carlos De Martin. The network neutrality bot architecture: a preliminary approach for self-monitoring of Internet access QoS. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 1131–1136. IEEE, 2011

Outline of the method. The tool regularly performs four transmission tests from end users, in the background. Available tests are HTTP, BitTorrent, raw TCP and DASH (Dynamic Adaptive HTTP streaming). Every 30 minutes, Neubot measures upstream and downstream throughput, as well as the Round-Trip Time (RTT) over the four tests.

Results from tests. No global analysis with respect to traffic differentiation has been provided to this date. A comparison of throughput and RTT between access ISPs in the Turin area was published, but no further analysis was made. An aggregate study of available data over time and over different ISPs would be an added value to the project.

User's point of view. The user needs to install the tool,[61] which runs measurement tests periodically and upon user request. Users can also view statistics of the quality of their link over time on a dedicated web page.

[61] http://www.neubot.org/

2.5.11 Discussion

The main features of each of the reviewed methodologies are summarized in Table 2.1 for a more direct comparison.

We saw that most of the methods in the literature that make use of active measurements rely on the joint replay of an application and a control flow, of which a shaper targeting that same application should only affect the former and not the latter. This is the case of BT-Test, Glasnost, DiffProbe, Packsen, NetPolice and Differentiation Detector. This technique has a non-negligible drawback, since it takes for granted that the modifications applied to the control flow are such that a shaper will not affect it in any way. In order to create a control flow, the most common approach is to modify port numbers or payloads of an application flow. A port number can be modified so that it belongs to a different application that is supposed to not be shaped, as in the case of NetPolice with HTTP, or it can be chosen within the range of high values not registered for any application,[62] as in Glasnost, DiffProbe and Packsen. Expecting that one specific service, for instance HTTP, is never shaped, might be true in most cases, but it is not a robust assumption that could apply to any type of network. As for high unassigned port numbers, it might not also be a safe option, as reported by the authors of Differentiator Detector, who noticed that the two commercial shapers at their disposal labeled traffic with high port numbers as peer-to-peer. Another possibility is to randomize packet payloads, since Deep Packet Inspection (DPI) devices often run regular expression patterns on incoming packets to match them with known applications. While this would definitely avoid DPI classification, we cannot assume in advance that port numbers will not be taken into consideration by shapers in any network. Differentiation Detector tries to overcome this by replaying the control flow inside a VPN, but we saw in Section 2.1 that VPNs too might be the target of differentiation.

[62] Service Name and Transport Protocol Port Number Registry. *IANA*. URL: www.iana.org/assignments/port-numbers

Among the tools intended for end users, another common limit is that they do not scale easily to any application a user might want to test, even though the underlying methodology is not dependent on a particular application. The application flow is always provided directly by the tool developers from a set of pre-recorded traces,

which is the case of Glasnost, DiffProbe and Packsen. Glasnost works around this problem by allowing advanced users to create their own test from traces captured from running applications; it is definitely an added value, but it is not an easily scalable solution, since a new customized test has to be created each time a user wants to test a new application. Differentiation Detector, on the other hand, should in principle create a customized experiment from the traffic of the application selected by the user, but in the version released up to this date is only available with a set of pre-recorded traces. ShaperProbe instead focuses on level shifts in throughput, which for example with PowerBoost apply to any flow, and therefore it is not concerned with the type of traffic it replays.

Furthermore, only one application at a time can be tested in all the reviewed active methods. This is a direct consequence of detecting differentiation by comparing an application to a control flow. Even when multiple application traces are provided, a user still needs to run the tool separately on each one of them in order to test them all. This is not the case of NANO, the only passive method so far proposed in the literature: it continuously measures the performance of all user applications (with the possibility of excluding some for security reasons) while they run on the end host. The main limitation in NANO is that, in order for it to correctly infer differentiation, it needs a very large user base, in the order of thousands according to its authors.

It must also be pointed out that, if a differentiation detection tool becomes popular, a replayed pre-recorded application flow can be easily white-listed by an ISP and offered regular service, thus effectively circumventing the underlying methodology. An experiment in which the replayed traffic is not the same same across all users would definitely represent an advantage.

In the reviewed methods, the metrics analyzed to detect differentiation are mostly throughput, delays and losses. Indeed, the effect of a shaper on a flow translates into higher delays, possibly more losses, and consequently an overall lower throughput. Delays and losses alone already fully capture the effects of a shaping device and can also provide a better insight on the performance of a flow over time.

With the tool we propose in this thesis, ChkDiff, we try to overcome the limitations described above for active methods, as will be described in Chapters 4 and 5. The approach seen in NetPolice for backbone networks, making use of TTL-limited probes, is the most similar in spirit to the upstream experiment in ChkDiff. Before delving into the details of our methodology, we need to understand how routers respond to such probes, whether ICMP rate limitation is prevalent and how it is implemented. We do this in the next chapter, where we corroborate the validity of measurements using ICMP time-exceeded feedback from intermediate routers.

Table 2.1: Comparison of existing methods for the detection of neutrality violations.

Tool	Active or Passive	Experiment	Measured Metrics	Application Specific?	Detection Method	Availability of Results	Currently available
BT-Test	Active.	Replays synthetic control and application flows, varying port numbers, direction (upstream or downstream) and payload.	TCP RST injection.	Yes, it only tests BitTorrent flows.	Checks for RST injection and determines what factor (port, payload, direction) triggered differentiation.	Immediately.	No, but upgraded by Glasnost.
Glasnost	Active.	Replays synthetic control and application flows, varying port numbers, direction (upstream or downstream) and payload.	Throughput.	Yes, but extendable. Available tests: BitTorrent, eMule, Gnutella, Flash, POP, IMAP4, HTTP transfer, SSH transfer, Usenet.	Checks whether the throughput of the two flows differs by more than 50%.	Immediately.	Yes.
DiffProbe	Active.	Replays control and application flows at original and sustained rates.	One-way delays, losses.	No, but limited to the set of synthetic traces provided by the authors (Skype, Vonage).	Checks for delay discrimination by running Kullback-Leibler test on the delay distribution of selected packets of the two flows. Infers scheduler type (SP or WFQ). Checks for loss discrimination by running two-proportion z-test on the losses of the two flows in selected instants of the experiment.	Immediately.	Yes.
ShaperProbe	Active.	Replays a default trace at the path capacity.	Instantaneous received throughput.	No, but the replayed trace does not belong to any application.	Checks for a level shift in the received throughput. Estimates shaping rate and burst size.	Immediately.	Yes.
Packsen	Active.	Replays control and application flows, interleaved. The first replay lasts 20 s, the second one 100 s.	Inter-arrival times and bandwidth.	No, but limited to the synthetic trace provided by the authors (BitTorrent).	In the first experiment, it runs Mann-Whitney U-test on the distributions of inter-arrival times of the two flows to detect differentiation. Then, it compares sent and received bandwidth in the second experiment to infer the scheduler type (SP or WFQ).	Immediately.	Not available.
NANO	Passive.	It passively measures the performance of the applications run by the user.	Throughput, jitter, losses, TCP RST injection.	No.	It compares the user performance to a baseline of all other users with a similar setup, location, and time of the experiments, but connected to a different ISP.	Immediately, provided that a large enough baseline of users with similar characteristics is available.	Discontinued.

Continues on the next page.

Table 2.1: Comparison of existing methods for the detection of neutrality violations.

Tool	Active or Passive	Experiment	Measured Metrics	Application Specific?	Detection Method	Availability of Results	Currently available
Differentiator Detector	Active.	Replays the packet exchange of a given application through (control flow) and out (application flow) of a VPN.	For TCP flows: RTT's, throughput, losses. For UDP flows: jitter, losses.	No, but the current tool only provides pre-recorded application traces (YouTube, Netflix, Spotify, Skype, Viber and Google Hangout).	It runs the two-sample Kolmogorov-Smirnov test on the RTT distribution of the two flows, as well as a tailored test on the area between the two CDF curves. It compares maximum and average losses of the two flows over two replays.	Immediately.	Yes.
NetPolice	Active.	It probes ingress and egress routers of a given ISP from several hosts with TTL-limited packets.	Losses.	No, but limited to the synthetic traces provided by the authors. Available traces: BitTorrent, SMTP, PPLive and VoIP. HTTP trace is used as baseline for content-based differentiation detection.	To detect content-based differentiation on a given ingress-egress router pair, it runs the two-sample Kolmogorov-Smirnov test on the distribution of losses of application and control flows, from all the experiments targeting that path segment. Similarly, it runs the same test to detect routing differentiation based on previous-hop AS and next-hop AS.	Not intended for end users.	Not intended for end users.
Tomography-inspired model	Active.	End-to-end path measurements across all paths of a given network.	Works in principle on any end-to-end measurement.	No.	Given the exact topology of a network, it forms a system of equations where the sum of unknown link properties of a given path equals the measured end-to-end metric along that path. If the system is unsolvable, the network is not neutral. The model also tries to identify non-neutral links.	Not intended for end users.	Not intended for end users.
Neubot	Active.	Four transmission tests to a server every 30 minutes.	Throughput, RTT.	No, but limited to the synthetic traces provided by the authors: HTTP, BitTorrent, raw TCP and DASH .	It does not directly infer differentiation, but it monitors a user's access link over time.	Statistics are available on a dedicated web page.	Yes.

3
ICMP rate limitation

Several tools proposed by the research community rely on feedback from intermediate routers in order to infer network characteristics. Traceroute-based path discovery[1] is a notable example: by sending probes with increasing Time-To-Live (TTL) values until a given destination is reached, traceroute elicits ICMP time-exceeded errors on intermediate routers and the whole path to a destination is uncovered. By merging paths to many destinations, we can even infer the topology of the Internet (e.g. CAIDA's skitter project[2]), or the topology of specific ISP networks (e.g. Rocketfuel[3]). ICMP feedback from routers is also used to discover path performance properties such as bandwidth and delay. For example, Pathneck[4] tries to localize and characterize the bottleneck link on a given path, and Pathchar[5] leverages the relationship between transmission time and delay to infer the bit rate of network links.

The main problem that arises when making use of TTL-limited probes is that ICMP feedback from routers is often neither instantaneous nor entirely reliable. Indeed, as the generation of ICMP error messages takes place in the slow path of the data plane, manufacturers and operators impose a low priority on it, in order to minimize the overall load on routers. Other internal tasks mostly related to the control plane, like route computation and management operations, might take precedence over it, especially when resources are shared between slow path and control plane. In addition, in order to further reduce the impact of ICMP message generation, the responsiveness to expiring packets is often limited by a hard-wired or configurable maximum rate,[6] above which routers simply ignore any expired packets and stay silent. All these limitations of the ICMP generation process can have repercussions on measurement tools and need to be thoroughly understood.

In this chapter, we try to shed light on the way ICMP rate limitation is implemented on routers and analyze its impact on active measurements. We proceeded for this purpose with a large-scale measurement campaign on PlanetLab, where we targeted at various probing rates 850 routers located at different depths into the network. Our contributions are the following:

- We identify an *on-off* pattern in the way ICMP rate limitation is

[1] Van Jacobson. traceroute. URL: `ftp://ftp.ee.lbl.gov/traceroute.tar.gz`

Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 153–158. ACM, 2006

[2] K Claffy, Tracie E Monk, and Daniel McRobb. Internet tomography. *Nature*, 7(11), 1999

[3] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002

[4] Ningning Hu, Li Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. A measurement study of internet bottlenecks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1689–1700. IEEE, 2005

[5] Allen B Downey. Using pathchar to estimate internet link characteristics. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 241–250. ACM, 1999

[6] RA Steenbergen. A practical guide to (correctly) troubleshooting with traceroute. *North American Network Operators Group*, pages 1–49, 2009

most often implemented and devise a taxonomy of routers accordingly.

- We determine the most popular configuration parameters on rate-limited routers, with respect also to their vendors.
- We demonstrate that the measured round-trip time for TTL-limited probes is not correlated with the choice of probing rate.

Developers of measurement tools who might have a concern about exceeding rate limitation thresholds can draw lessons from our findings, as we show that it is relatively easy to observe an answering pattern and possibly filter it out. On the other hand, when it is essential to have a high answering rate, it might be of use to know in advance which settings are the most common across vendors.

For the purpose of the upstream experiment in ChkDiff (Chapter 4), we will see that we can indeed use TTL-limited probes to measure RTTs and we can send packets at relatively high probing rates without hitting any capacity limits of routers.

We presented the results of this chapter in a paper published at ICC '15.[7]

[7] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Characterizing ICMP Rate Limitation on Routers. In *IEEE International Conference on Communications (ICC)*, 2015

3.1 Overview

Before delving into an accurate description of our measurement setup, we provide upfront two results: one introduces the effect of ICMP rate limitation on active probes, the other adds an important insight on the delays obtained.

First, we ask ourselves which fraction of routers is affected by ICMP rate limitation and at which probing rates this is most evident. In Figure 3.1, we plot the loss rates that we experienced when probing a large set of routers with TTL-limited packets at different sending rates in the range of [1, 2543] packets per second (pps). The loss rate, which we define as the number of received ICMP time-exceeded messages over the number of sent probes, is drawn in a separate bean plot for each probing rate. Bean plots show a scatter plot of all individual values on top of a kernel density estimate of the probability density function of the data. Median and mean are denoted, respectively, by a blue cross and a short green horizontal line. Two trends are evident in Figure 3.1: (i) the median loss rate remains very low for probing rates up to 372 packets per second (pps) and it progressively increases up to 90% with higher probing rates; (ii) a non-negligible fraction of experiments shows the above behaviour already for rates higher than 54 pps. But when exactly does ICMP rate limitation take place? How is it most commonly implemented and what are the most frequent configurations? We will address these questions in Sections 3.3 and 3.4.

Besides being concerned about collecting a sufficient number of responses, we are interested to know whether the delay associated to them is in any way biased by the sending rate that we choose. In

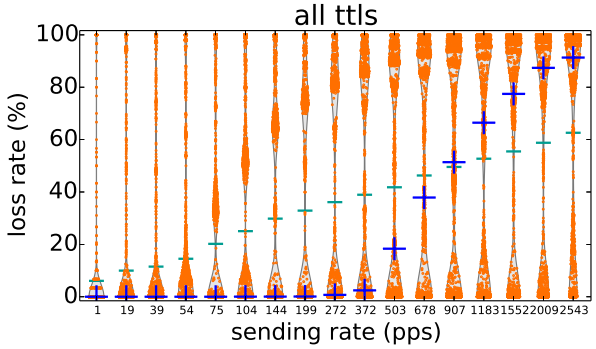


Figure 3.1: Loss rates for all experiments, arranged by probing rate.

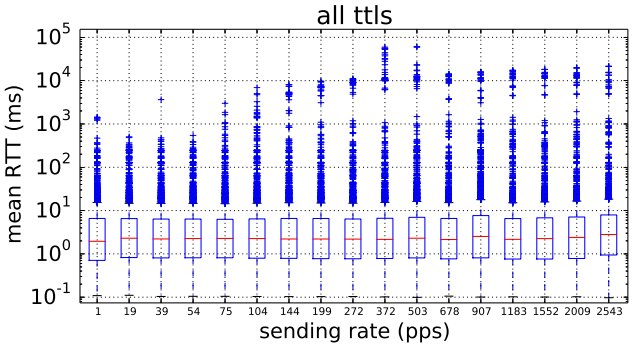


Figure 3.2: Mean RTT box plots for all experiments, arranged by probing rate.

order to verify this, we plot in Figure 3.2 the mean Round-Trip Time of each experiment, arranged by probing rate (a full description of the measurement campaign is presented in Section 3.2). The box plots are self-explanatory: both the medians and the IQRs[8] appear steady across all the tested probing rates. We can therefore conclude that the choice of probing rate has no visible influence on the average round-trip time. We can also observe that a few experiments resulted in mean values that are far away from the core of the distribution (at 1, 10 and up to 60 seconds). These outliers start to appear at 75 pps and seem to follow a linear increase. Such behavior was limited to a dozen routers that persistently behave in this way and constitute an exception as compared to the rest of routers in our dataset. We further discuss the delay distribution in Section 3.5.

[8] IQR is the Inter-Quartile Range, defined as the difference between the 75^{th} and the 25^{th} percentiles of the considered distribution. It corresponds, by definition, to the length of the box and aggregates 50% of the samples.

3.2 *Experimental setup*

We conducted our measurement campaign from 180 PlanetLab hosts, each of which located at a different site in order to avoid overlapping measurements. Each run lasted 30 seconds and targeted one single hop along the path from a PlanetLab host to a fixed IP destination. A single experiment consisted in sending at a constant rate ICMP echo-request probes (similar to what the Ping tool uses) with a forged IP Time-To-Live (TTL) value, so that the packets would expire on the router at the desired hop and generate ICMP time-exceeded error messages. Each sent probe was 28 Bytes in length (20 Bytes for the IP header and 8 Bytes for the ICMP header), with no extra payload added to the end of the packet.

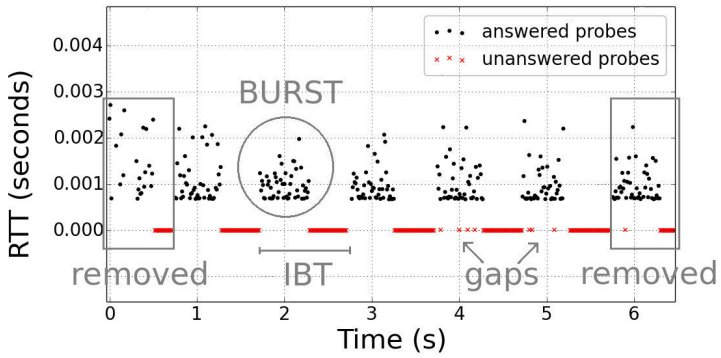


Figure 3.3: A typical timeseries for an on-off rate-limited router.

The choice of ICMP for our probes instead of UDP or TCP followed the results by Gunes and Sarac,[9] who showed with large-scale measurements that, in comparison, ICMP probes elicit the highest number of responses. We decided to target the first 5 hops from each PlanetLab host in order to include in our dataset both edge and backbone routers. The IP destination address was a public IP address assigned to a machine in our use. It was kept unchanged in all probes, thus effectively reaching no more than one router per hop, except for only 3 cases in which per-packet load balancers were encountered (a low prevalence in line with what was observed in large-scale measurements by the authors of Paris Traceroute[10]). We selected 17 exponentially-spaced sending rates between 1 and 4000 packets per seconds, in order to capture the behaviour of routers at low and relatively high probing regimes. We note here that the majority of PlanetLab hosts could not fully keep up with the desired sending rates, especially with the highest ones ($>$ 1500 pps). Consequently in all figures, for a desired probing rate r, we actually show the median of all measured sending rates achieved by the PlanetLab hosts when instructed to probe at rate r. Every $(host, hop, rate)$ tuple was tested 3 times. We tried to stress routers as little as possible between experiments by shuffling the order of sending rates and by never choosing the same hop in consecutive experiments. The measurement campaign was performed in the first two weeks of February 2014 using NEPI,[11] a Python-based library for the deployment of experiments on network evaluation platforms. Our resulting dataset includes over 45000 path measurements and 850 distinct routers. Only 53 routers appeared in more than one path, generally at hops 4 or 5 from hosts in the same country.

[9] Mehmet H Gunes and Kamil Sarac. Analyzing router responsiveness to active measurement probes. *Passive and Active Network Measurement*, pages 23–32, 2009

[10] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring load-balanced paths in the internet. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 149–160. ACM, 2007

[11] Alina Quereilhac, Mathieu Lacage, Claudio Freire, Thierry Turletti, and Walid Dabbous. Nepi: An integration framework for network experimentation. In *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, pages 1–5. IEEE, 2011

3.3 *Analysis of ICMP rate limitation*

In the dataset we collected, two types of rate limitation were present. In the first one, which we call *on-off*, the router followed a clear answering pattern. In the second one, which we will just call *rate-limited* (*rl*), the overall answering rate is constant, but without a visible pattern.

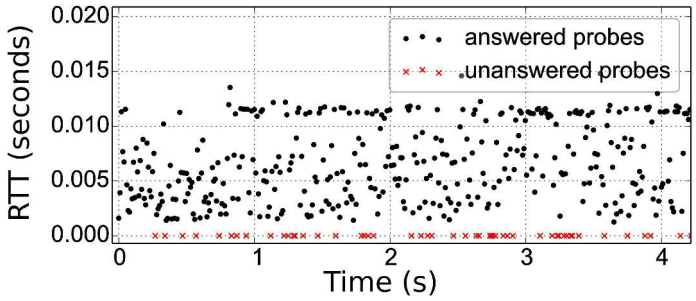


Figure 3.4: A typical timeseries for a generically rate-limited router.

3.3.1 *On-off*

A typical example of an experiment targeting an ICMP rate-limited on-off router is displayed in the timeseries in Figure 3.3, where we report on the x-axis the time at which the probe was sent and on the y-axis the corresponding round-trip time if an ICMP time-exceeded packet was received for this probe. Whenever no reply was received, a red cross appears on the x-axis at y=0. We define:

- a **burst** as any series of consecutive answered probes delimited by unanswered probes. What we are interested in is the *size* of a burst (**BS**), in packets.
- an **Inter-Burst Time (IBT)** as the time interval between the first probe of a burst and the first probe of the next burst.

A *period* coincides therefore with the occurrence of a burst, followed by a series of unanswered probes.

Tolerance to noise. In experiments similar to the one in Figure 3.3, occasional unanswered probes inside a burst would split that burst into smaller ones, according to the above definition. If instead we tolerate the occurrence of a few unanswered probes (that we call a *gap* and denote its size by g) and decide to end a burst only when more than g unanswered probes appear in a row, we are able to catch also those cases of visually identifiable bursts that a more conservative approach would miss. This allows us to account for potential losses in the network and occasional interruptions in the generation of ICMP time-exceeded packets on routers when more urgent tasks, for example related to fast path operations, need to take place.

On-off behaviour. If we represent each experiment as a series of burst sizes (BS's) and Inter-Burst Times (IBT's), we can determine the degree of regularity of the observed burstiness by examining the coefficient of variation[12] (CoV) of the two metrics, which we call CoV_{BS} and CoV_{IBT}. Through a careful visual inspection of many timeseries, we observed that a significant fraction of routers featured a typical *on-off* pattern when answering to our probes, where an *on* state is a state in which all or nearly all probes are answered and an *off* state is a state during which no answer is received. This essentially corresponds to a variant of a token bucket where tokens, valid only for the

[12] The coefficient of variation is defined as the ratio between the standard deviation of a random variable and its mean. It measures the variability of a distribution in units of the mean and enables a direct comparison between distributions having different means.

duration of one period, are generated periodically and in bursts, with each token being a ticket to send out an ICMP reply when needed. Once all tokens expire, the router simply ignores any arising event requiring the transmission of an ICMP reply and continues to do so until the next period, when a new burst of tokens arrives. Taking into consideration the presence of noise as previously described, we define an experiment as *on-off* when both its CoV_{BS} and CoV_{IBT} lie below an empirically chosen threshold of 0.05. That is to say that, when the series of BS's and of IBT's contain values that do not differ from one another by more than 5%, the router is flagged as on-off for this experiment. The threshold of 0.05 appeared to be a good trade-off that minimizes the incidence of false positives.

3.3.2 *Generic rate limitation*

We will see in Section 3.4 that a fraction of routers in our dataset are indeed rate-limited when being probed at rates higher than a router-specific threshold, but the way in which they respond does not follow an on-off or any other recognizable pattern. An example is provided in Figure 3.4, where it is clear that, even though the number of unanswered probes is approximately the same every second, the order at which the router decides whether to generate ICMP time-exceeded replies is not as predictable as in the on-off case. We now move to a more detailed characterization of routers based on their responsiveness.

3.4 *Characterization of routers*

In our measurement campaign, we observed that the vast majority of routers behave according to three responsiveness phases depending on the rate at which we probe. As we probe at increasingly high rates, we have, in order of appearance:

- **fully responsive phase**, in which a router replies to the probes it receives in a timely manner. The loss rate is $< 5\%$, with sporadic exceptions attributable to the router load, network congestion or other minor causes. $[r_{min}, r_0)$ is the range of probing rates at which this phase takes place.

Table 3.1: Number of routers in each category.

Category	Description	# of routers (%)
fr	Fully Responsive	257 (30.2%)
fr-irr	Fully Responsive then Irregular	51 (6.0%)
fr-rl	Fully Resp. then Generically Rate Limited	118 (13.9%)
fr-onoff	Fully Responsive then On-Off	211 (24.8%)
fr-onoff-irr	Fully Resp. then On-Off then Irregular	180 (21.2%)
unresponsive	No answer	33 (3.9%)

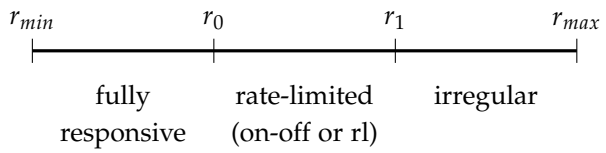


Figure 3.5: Responsiveness of routers to TTL-limited probes, broken down into three phases, according to the rate at which we send probes.

- **rate-limited phase**, where ICMP rate limitation is turned on as a result of a probing rate higher than a router-specific threshold. If on-off, the router responds at a constant rate and any excess probes at every time period will be simply discarded. If generically rate-limited (*rl*), the router has an overall constant answering rate, but excess probes are dropped without a clear order. Refer to Section 3.3 for more details. This phase is defined in $[r_0, r_1]$.
- **irregular phase**, during which the router reaction is less predictable than before. The rate at which we probe hits a capacity limit and the router fails to reply in a regular way. Generally, two things might happen: the loss rate gradually increases to 100%, or the on-off pattern is not as precisely observable as before, with the overall responsiveness being nonetheless roughly unchanged. This is the behaviour observed in $(r_1, r_{max}]$.

The three phases are displayed in Figure 3.5, where we called: r_{min} the minimum probing rate at which we hit a router (in our case 1 pps); r_0 the minimum probing rate at which rate limitation appears (with r_0 being necessarily $\geq r_{rl}$, the rate at which a router is configured to reply when in the rate-limited phase); r_1 the highest probing rate at which we notice a well-defined rate limitation; r_{max} the maximum rate at which we probe. Our classification of routers is based on which of the three phases occur in the set of probing rates in $[r_{min}, r_{max}]$ used in our measurement campaign. As mentioned in Section 3.2, we tested exponentially-spaced rates in the range $[1, 4000]$ pps, but most PlanetLab nodes failed to achieve the highest values. Typically, a router will have a fully-responsive (*fr*) phase, followed by a rate-limited phase (if configured), and possibly by an irregular phase (*irr*). We can now devise the following taxonomy:

- **Rate-limited routers.** Routers where a rate-limited phase occurred. We define three sub-categories:
  a) ***fr-onoff.*** Routers that are fully responsive in $[r_{min}, r_0)$ and on-off for higher probing rates, that is to say in $[r_0, r_{max}]$, with $r_1 \geq r_{max}$.
  b) ***fr-onoff-irr.*** Routers that are on-off only within a given range of probing rates, above which they exhibit an irregular behaviour. They are fully responsive, as in the previous case, in $[r_{min}, r_0)$, but have an on-off phase in $[r_0, r_1]$, with $r_1 < r_{max}$. An irregular phase appears in $(r_1, r_{max}]$.
  c) ***fr-rl.*** Routers that are fully responsive up to r_0, after which their answering rate is constant at a value r_{rl} (with r_{rl} neces-

sarily $\leq r_0$), without the occurrence of any on-off pattern. We note here that no *fr-rl-irr* routers were encountered.

- **Non rate-limited routers.** Routers in which the rate-limited phase is not observed. We have two cases:

  a) ***fr.*** Routers that are fully responsive for the entire range $[r_{min}, r_{max}]$, in which they reply to all (or nearly all) probes. The condition verified here is that the loss rate in the majority of the experiments is $< 5\%$, a value arbitrarily chosen as an indication of high responsiveness. This essentially corresponds to the case in which no rate-limited or irregular phase took place for probing rates in $[r_{min}, r_{max}]$.

  b) ***fr-irr.*** Routers that are fully responsive (loss rate $< 5\%$) up to a given probing rate r_{irr}, after which their loss rate starts to increase, without any rate-limiting pattern being observed.

- **Unresponsive.** Routers that are configured to never reply to expiring IP packets. 33 such cases were encountered, amounting to a mere 4% of the total number of routers in the dataset.

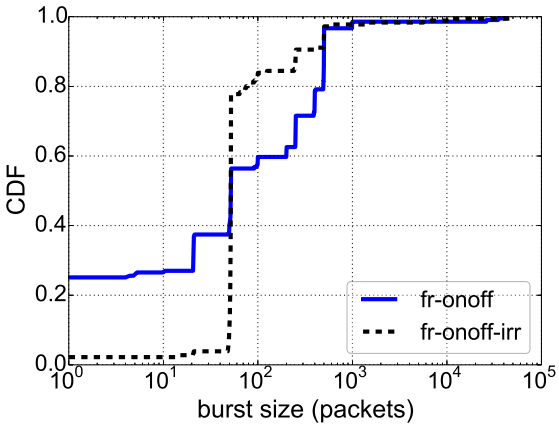
Interestingly, the six categories appeared quite evenly spread out with regard to the hop-distance from the PlanetLab hosts.

In our subsequent analysis, we chose to use a gap value of 4, since it maximizes the number of on-off rate-limited routers without being so high as to include possible false positives. The breakdown for our set of 850 routers is detailed in Table 3.1. The vast majority (96%) of routers are responsive, which is good news, as tools like traceroute are key instruments for network troubleshooting. 30% of the routers answer to all probes (up to the maximum probing rate used in our experiments), while about 60% feature rate limitation at some point, either with a clear on-off pattern (46%) or without (13.9%).

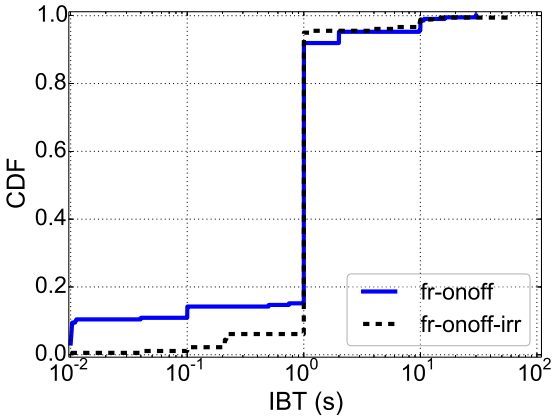
3.4.1 On-off routers

For a given on-off router, its on-off behaviour can be described by its (BS, IBT) pair. If we divide BS by IBT, we obtain the rate limitation threshold of an on-off router, or in other words its answering rate r_{onoff} during the on-off phase.

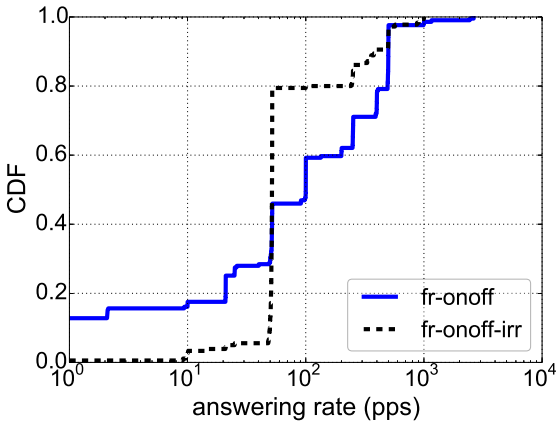
Distribution of Burst Size. In Figure 3.6(a) we compare the cumulative distribution functions (CDF's) of burst sizes for *fr-onoff* and *fr-onoff-irr* routers. In the case of the 211 *fr-onoff* routers (as seen in Table 3.1), burst sizes span from 1 to 39,000 packets. The most common values are: 1 (25%), 50 (20%), 500 (15%), 20 (10%), and 250 (10%). Almost all remaining values are multiples of 50, each one of them not occurring in more than 5% of all cases. As for the 180 *fr-onoff-irr* routers, while the set of burst sizes is similar to that of the *fr-onoff*, we found a much larger fraction of routers configured at 50 pps. The most frequent values are the following: 50 (70%), 250 (7%) and 500 (7%). Even though the two CDF's look fairly different, the



(a) Burst Size CDF.



(b) Inter-Burst Time CDF.



(c) Answering rate CDF by router

Figure 3.6: CDFs of BS, IBT and overall answering rate for on-off routers.

set of burst sizes for both router categories includes reasonable values that a network administrator could have very well picked. We will see in Section 3.4.3 that this is due to a dominance, among the *fr-onoff-irr*, of Juniper routers, which are mostly rate-limited at 50 pps.

Distribution of Inter-Burst Time. Similarly to what we did for the burst size, we now study the distribution of the Inter-Burst Time, shown in Figure 3.6(b). The two curves are rather similar, with 1 second as the most frequent value. We also observe a larger fraction of *fr-onoff* routers at $IBT = 0.01\,s$. In the case of *fr-onoff* routers, the values span from 4 ms to 30 s with the most common ones being: 1 s (70%), 0.01 s (10%), 2 s (5%) and 10 s (5%). For *fr-onoff-irr* routers the observed range is (0.01 s, 60 s), where we have: 1 s (85%), 0.02 s (5%) and 10 s (5%). We can also notice outliers on the right-hand side of the CDF curves: about a dozen PlanetLab hosts used in our measurement campaign were heavily loaded at the time and our 30-second experiments lasted in reality up to 2 minutes (with the measured probing rates being considerably smaller than the desired ones); surprisingly, this let us find a couple of cases where a

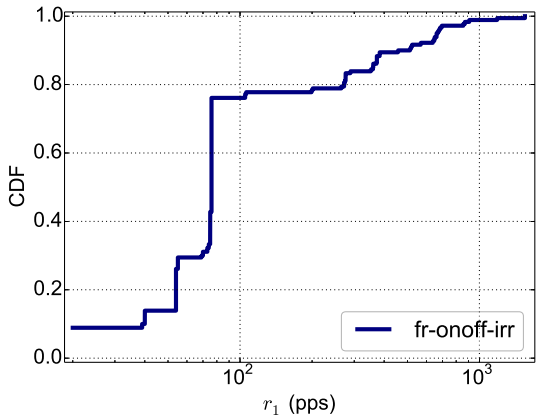


Figure 3.7: Distribution of r_1 for fr-onoff-irr routers.

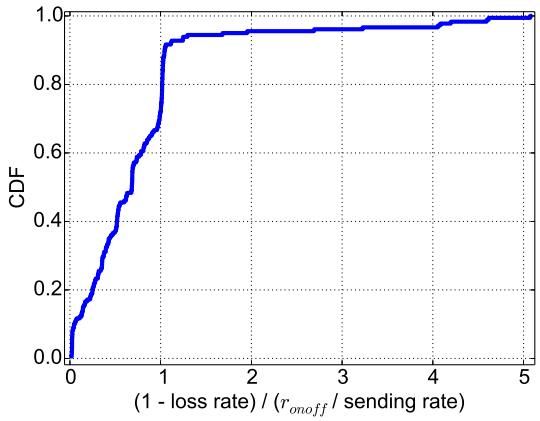


Figure 3.8: Answering rate over *expected* answering rate in *irr* phase for on-off routers.

very long series of answered probes (see the outliers in the BS CDF) was interrupted every 30 or 60 seconds by a relatively short series of unanswered packets. Our suspicion is that this is most likely due to an internal process requesting the router resources at those precise intervals.

Distribution of answering rate r_{onoff}. We consider the resulting answering rate in the on-off phase by dividing *BS* over *IBT*. With 1 second as the most common *IBT*, there are no surprises in Figure 3.6(c): the values are roughly the same seen in Figure 3.6(a). The most common values for *fr-onoff* routers are (in packets per second): 500 (28%), 50 (18%), 100 (15%), 1 (15%), 20 (10%), 250 (10%), 400 (9%) and 250 (5%). In the case of *fr-onoff-irr* routers, the CDF shows a dominant value: 50 (75%). The other observed answering rates are: 250 (8%), 500 (8%), 400 (5%) and 10 (5%). We can also notice that 50% of *fr-onoff* routers have a rate limitation threshold $\leq$ 100 pps, whereas 60% of the *fr-onoff-irr* have a lower threshold: 50 pps or less.

Behaviour in $(r_1, r_{max}]$. We said that on-off routers of type *fr-onoff-irr* cease to be on-off at probing rates higher than r_1, which varies from router to router. In Figure 3.7 we show the distribution of r_1,

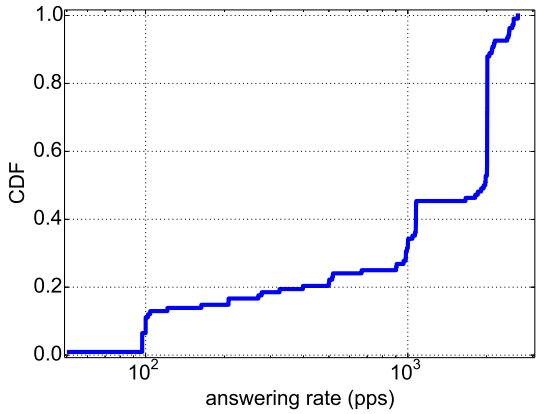


Figure 3.9: Answering rate for generically rate-limited (*rl*) routers.

which reveals that 76% of these routers moved to the irregular phase already when we probed them at less than 76 pps, with this value being r_1 for 45% of them. In order to study what happens during the irregular phase, we plotted in Figure 3.8 the CDF of the (per-router) average ratio between the actual answering rate of each experiment in the *irr* phase and the answering rate if the router were still in the *on-off* phase. In the curve, values close to 0 indicate little (or no) responsiveness, which is the case for around 10% of these routers. Values on 1 suggest that, even though the on-off pattern is not as strictly enforced as before, the overall answering rate is the same as in the *on-off* phase. This is true for around 20% of routers in this category. Values between 0 and 1, where more than 50% of routers lie, show instead a decreased answering rate. We leave for future study an explanation for the outliers on the right-hand side of the curve, which indicate an unexpected increase in responsiveness.

3.4.2 *Generically rate-limited routers*

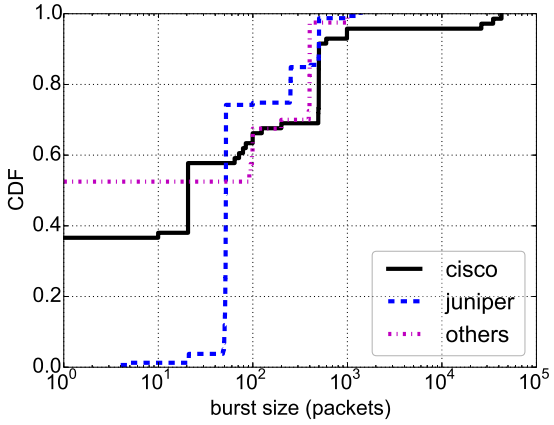
Generically rate-limited routers keep the answering rate constant for all probing rates in the rate-limited phase. In Figure 3.9 we study the distribution of answering rates for these routers. Compared to the answering rates seen for the on-off category, here the values are sensibly higher: almost 50% are rate-limited at 2000 pps, 20% at roughly 1000 pps and 15% at 100 pps.

3.4.3 *Vendors*

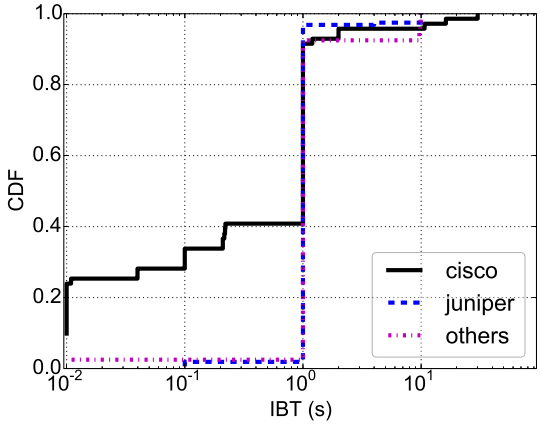
ICMP rate-limitation parameters can be hard-coded into a router or configurable by its administrator, depending on the router vendor. For example, according to Cisco documentation,[13] Cisco 6500 and 7600 routers can be configured with the desired answering rate and burst size. Steenbergen[14] reports that in Cisco GSR routers the limitation rate is hard-coded, which is also the case for Juniper routers, whose answering rates vary depending on the model: 50, 250 or 500 pps. With this in mind, we fingerprinted the routers in our dataset in order to determine their most common configurations. For routers

[13] Cisco. TTL expiry attack identification and mitigation. URL: `http://www.cisco.com/web/about/security/intelligence/ttl-expiry.html`

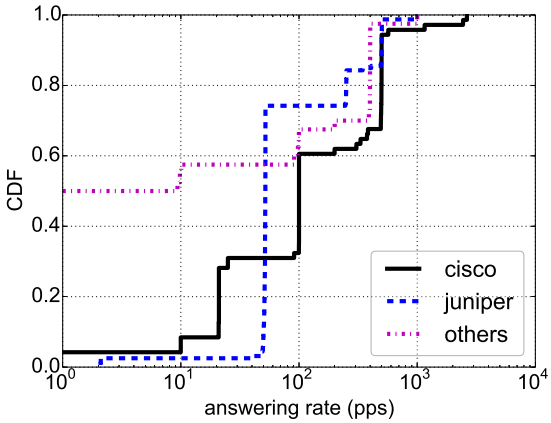
[14] RA Steenbergen. A practical guide to (correctly) troubleshooting with traceroute. *North American Network Operators Group*, pages 1–49, 2009



(a) Burst Size CDF by vendor.



(b) Inter-Burst Time CDF by vendor.

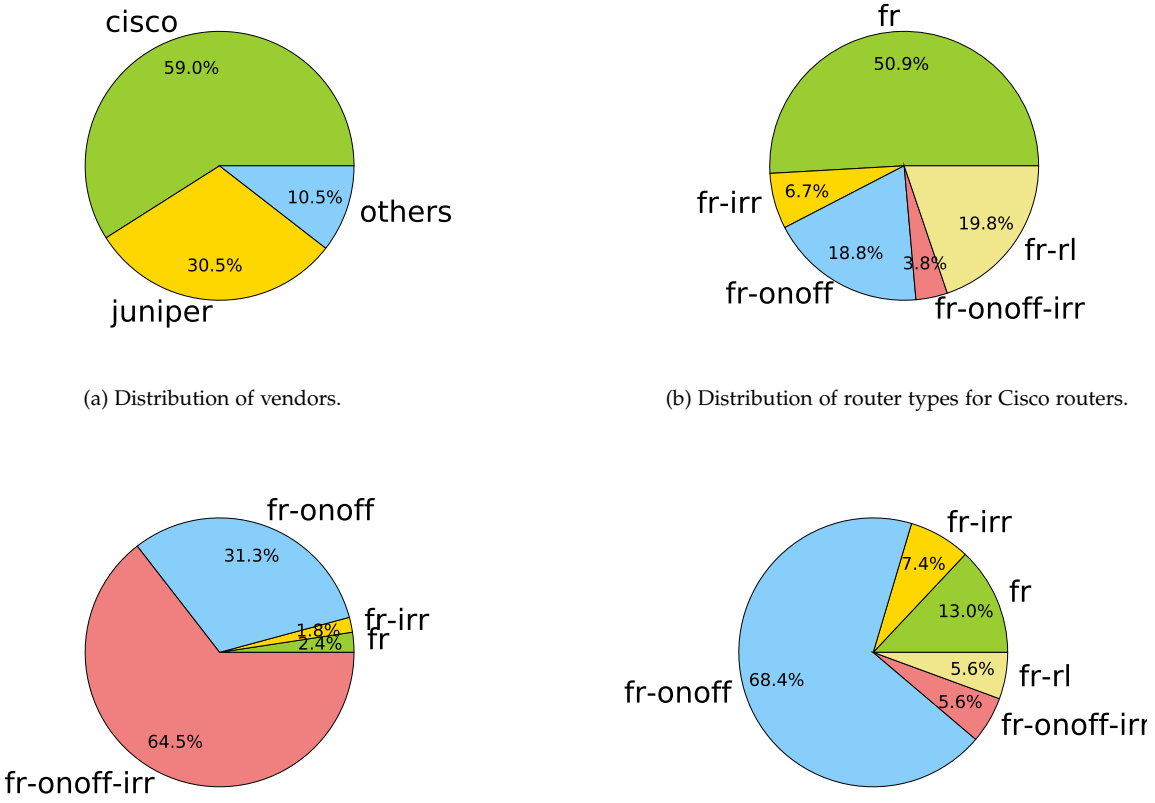


(c) Answering rate CDF by vendor.

Figure 3.10: CDFs of BS, IBT and answering rate for on-off routers, arranged by router vendor.

at hop 1 we simply looked up their MAC address, whereas for all other ones we used the technique described by Vanaubel et al.,[15] through which we can identify a router as Cisco, Juniper or "others" by considering the IP TTL of the response to an ICMP echo-request and the IP TTL of the response to a TTL-limited probe. The distribution of vendors seems to follow the known prevalence of Cisco in the market: 59% of routers were labeled Cisco, 30.5% Juniper and 10.5% others. In Figure 3.11 we see how the different categories are distributed for each vendor: Cisco has a large fully-responsive component (50.8%), almost all (95.8%) Juniper routers are on-off and those marked as "others" are also mainly on-off (74.1%). For on-off routers, we show in Figure 3.10 the distribution of BS, IBT and answering rate arranged by vendor. A few considerations: (i) Cisco routers are mostly configured at 20, 100 or 500 pps; (ii) IBT is generally always 1 second, except for Cisco routers, which display more values (probably as a consequence of being configurable in software) and (iii) most Juniper routers are rate-limited at 50 pps, the dominant value we saw previously for *fr-onoff-irr* routers. It is no coincidence, as these are mostly Juniper.

[15] Yves Vanaubel, Jean-Jacques Pansiot, Pascal Mérindol, and Benoit Donnet. Network fingerprinting: TTL-based router signatures. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 369–376. ACM, 2013



(a) Distribution of vendors.

(b) Distribution of router types for Cisco routers.

(c) Distribution of router types for Juniper routers.

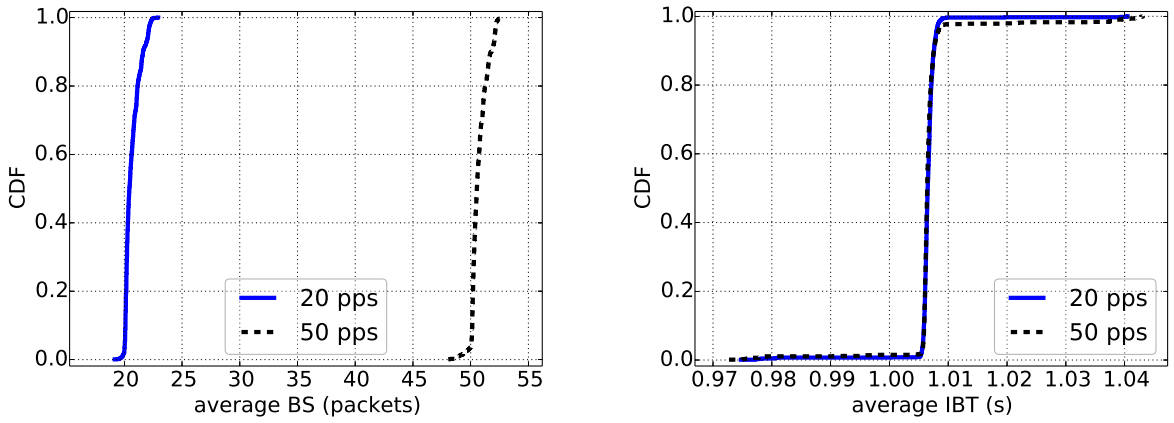
(d) Distribution of router types for "other" routers.

Figure 3.11: Distribution of vendors and percentage of routers in each category for Cisco, Juniper and "others".

3.4.4 *Validation by controlled experiments*

We resorted to controlled experiments in order to verify that (i) we infer the correct on-off parameters, and (ii) PlanetLab hosts do not add any bias . We arbitrarily chose 60 machines from the PlanetLab testbed, each one of which located at a different site, and instructed them to probe (in sequence) a machine in our control, where we emulated the on-off behaviour of routers with the use of Linux *iptables*.[16] For simplicity, we did not make the ICMP echo-request probes expire on this machine, but we directly replied to them with ICMP echo-reply messages. Similarly to our large-scale measurements, each single experiment lasted 30 seconds, during which a PlanetLab host probed our machine at a constant rate. We picked 12 exponentially-spaced probing rates in the interval [1, 1000] pps and tested each rate twice. On the machine in our control, we emulated an on-off router by using for a first round of measurements a rate limitation of 20 pps ($BS = 20$ packets, $IBT = 1$ second) and for a second round a value of 50 pps ($BS = 50$ packets, $IBT = 1$ second). In Figure 3.12 we show the CDF of the measured average Burst Size (BS) and Inter-Burst Time (IBT). While for the IBT the correct value is precisely measured, an error of up to 2 units is often encountered in the estimation of the BS parameter. After a manual

[16] `http://www.netfilter.org/projects/iptables`



(a) Burst Size CDF.

(b) Inter-Burst Time CDF.

Figure 3.12: Verification of inferred parameters on a controlled rate-limited machine.

inspection of several experiments, we can only conclude that it is an implementation-dependent issue: the *BS* parameter is simply not always as rigorously enforced as the *IBT* in *iptables*. Nonetheless, the correct inference of the above parameters lets us exclude the existence of any bias coming from the PlanetLab hosts or the networks where they reside.

3.5 Delay

In Section 3.1 we observed that the rate at which we send probes does not influence the resulting round-trip time, whose mean value is stable across all probing rates under consideration. We add here further evidence to support our claim by analyzing the per-experiment RTT variability. For each experiment, we took the coefficient of variation (CoV) of all its round-trip times and plotted in Figure 3.13 all CoVs arranged by probing rate and regardless of the hop distance. Apart from the case of 1 packet per second, for which the CoVs refer to samples of only 30 RTT's (as each experiment lasts 30 seconds), where apparently less variability occurs, for all other probing rates there does not seem to be any noticeable difference across them. The interquartile range appears stable, with a very slight decrease of the median for higher rates. Therefore, we can apparently conclude that the sending rate has no direct impact on the round-trip time of TTL-limited probes: its mean is not altered, and neither is its variability.

3.6 Related Work

To the best of our knowledge, our work is the first of its kind to attempt to precisely characterize ICMP rate-limitation on routers at a large scale. Gunes and Sarac[17] analyzed publicly available traceroute data collected between 1999 and 2008, and noticed that in recent years network operators have configured routers to become increasingly less cooperative to active probing. Then, they conducted large-

[17] Mehmet H Gunes and Kamil Sarac. Analyzing router responsiveness to active measurement probes. *Passive and Active Network Measurement*, pages 23–32, 2009

scale measurements with direct (i.e. destined to a router) and indirect (i.e. traceroute-like) probes and compared the responsiveness of routers according to the chosen protocol: ICMP, TCP or UDP. They concluded that ICMP probes elicit the highest number of responses for both direct and indirect probes, followed by TCP and UDP in the case of direct probes, and by UDP and TCP for indirect ones. It is also based on this finding that we decided to use ICMP probes in our measurements. Govindan and Paxson[18] proposed a technique to estimate the time taken by a router to generate an ICMP time-exceeded message: given a router R located on the path from host A to host B, they compared the one-way delay from A to B experienced by direct probes and by spoofed TTL-limited probes that expire on R but whose ICMP error message is sent to B. They found out that for most routers the slow path time is less than 0.5 ms. Malone and Luckie[19] tackled an issue tightly related to the use of TTL-limited probes: the matching between probes and ICMP time-exceeded messages, based on the quoted contents of the expired probes inside the returned ICMP message. They detailed a variety of packet field modifications applied by routers and middleboxes that might result in discrepancies between the quoted packet and the original probe. Incidentally, they also pointed out that in their measurements there were a few tens of probes that experienced incredibly high RTT's, spanning from 10 to 300 seconds. We encountered exactly the same kind of outliers when analyzing our dataset (as can be seen in Figure 3.2). Layouni et al.,[20] who studied in depth the causes behind undisclosed routers in traceroute, also reported very high round-trip times and attributed them to high activity in the control plane, which at least in our dataset seems more likely to be caused by our own probing rather than by other parallel processes.

[18] Ramesh Govindan and Vern Paxson. Estimating router ICMP generation delays. In *Passive & Active Measurement (PAM)*, 2002

[19] David Malone and Matthew Luckie. Analysis of ICMP quotations. *Passive and Active Network Measurement*, pages 228–232, 2007

[20] Farah Layouni, Brice Augustin, Timur Friedman, and Renata Teixeira. Origine des étoiles dans traceroute. In *Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, 2008

3.7 *Summary*

Although the use of ICMP rate limitation on routers in the Internet has been known for a long time, no previous study had tackled the problem of precisely characterizing how this function is implemented in the wild. We analyzed the RTT distribution obtained when targeting routers with TTL-limited probes and found that it is apparently uncorrelated with the chosen probing rate. We intro-

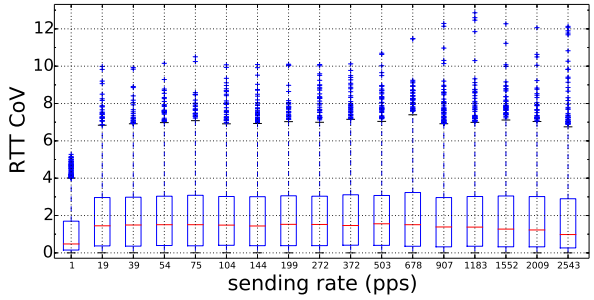


Figure 3.13: RTT CoV box plots for all experiments, arranged by probing rate.

duced a classification of routers based on their responsiveness across different probing rates and observed that rate limitation most often consists of an on-off process, where the router alternates between a state in which it answers to all probes and a state during which it remains silent. We analyzed in details the configuration parameters of on-off routers: burst size, for which we detected a variety of values, and inter-burst time, generally equal to 1s. A future direction for this work could extend our measurements to go deeper into the core of the Internet and group routers with respect to the ISP that manages them. This will allow us, for example, to check if ISPs apply a consistent configuration across their routers.

Now that we have shown that round-trip times obtained with TTL-limited packets can be used for delay measurements, we can introduce in the next chapter the upstream experiment of our tool, ChkDiff, which relies on this type of feedback from intermediate routers to infer differentiation and localize shapers.

4
ChkDiff: the upstream experiment

The neutrality of the Internet has been a hot topic ever since, around a decade ago, bandwidth-hungry applications (e.g., peer-to-peer, video, media streaming) started to gain success among users and a number of ISPs took measures to counter possible detrimental effects on the connectivity they provided. Arbitrary decisions, as for example blocking of BitTorrent traffic in the upstream direction[1] by an operator in the U.S., have since then been reported. As we saw in Chapter 2, other examples include: throttling of YouTube in France[2] and Germany[3] during evening hours, when link utilization reaches its peak; degraded performance over a VPN using OpenVPN default port[4] in the U.S.; most recently, evident decrease in performance for Netflix traffic in early 2014 by two U.S. operators.[5]

But since traffic differentiation is rarely revealed by official sources and certainly does not appear in Service Level Agreements (SLA's), it becomes of utmost importance to be able to detect it from within the network. A number of tools for the detection of traffic differentiation have thus emerged in the literature in the past few years (we reviewed them in Chapter 2). The method we propose here differs from existing work in its attempt to be independent both from the shaping techniques in use by ISPs at layer 3 (IP) of the protocol stack and from the user applications that might be targeted. The idea is that a shaper whose goal is to degrade the performance of selected traffic might do so according to a variety of packet scheduling and buffer management policies, but it will typically result on the user side in larger delays and possibly more losses. Consequently, if we compare the set of delays of a flow to that of the rest of the traffic, and we proceed analogously for losses, we should be able to infer whether any shaping took place. In order for this to be valid for whatever application the user is running, we reuse her own traffic and replay it with minimal changes so that it targets the routers at the first few hops from her. If any shaper is located in proximity to one (or more) of these routers, packets going through it will have degraded performance at that hop and at all subsequent ones, thus allowing us to also pinpoint their relative position. We implemented this technique for upstream traffic in a tool we called ChkDiff,[6] through which we can detect the presence and identify the location of shapers, by using the ICMP feedback provided by

[1] Dslreports: comcast is using sandvine to manage p2p connections. URL: `http://www.dslreports.com/forum/r18323368-Comcast-is-using-Sandvine-to-manage-P2P-Connections.`

[2] Respect my net. URL: `http://respectmynet.eu/view/205`

[3] Respect my net. URL: `http://respectmynet.eu/view/196`

[4] I just doubled my PIA VPN throughput that I am getting on my router by switching from UDP:1194 to TCP:443. URL: `http://www.reddit.com/r/VPN/comments/1xkbca/i_just_doubled_my_pia_vpn_throughput_that_i_am`

[5] Netflix performance on Verizon and Comcast has been dropping for months, 2013. URL: `http://arstechnica.com/information-technology/2014/02/netflix-performance-on-verizon-and-comcast-has-been-dropping-for-months`

[6] The code is available on a dedicated web page: `http://chkdiff.gforge.inria.fr/`

routers. We focus in this chapter on the validation of ChkDiff in the upstream direction: we stress the tool under different shaping scenarios and assess its resilience against sources of error such as traffic variation and ICMP rate limitation on routers.

The chapter is organized as follows: in Section 4.1 we present the functioning of our method in details, along with a discussion of all the technical adjustments needed for the tool to work; in Sections 4.2 and 4.3 we validate ChkDiff in respectively a controlled neutral and non-neutral environment and show that it is able to successfully detect shaping even when a large fraction of the traffic is differentiated; we compare our method to existing work in Section 4.4 and give concluding remarks in Section 4.5.

The results of this chapter were presented in a paper published at ITC 27.[7] An early draft of this work was presented at CoNEXT 2012.[8]

[7] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Towards a general solution for detecting traffic differentiation at the internet access. In *Teletraffic Congress (ITC 27), 2015 27th International*, pages 1–9. IEEE, 2015

[8] Riccardo Ravaioli, Chadi Barakat, and Guillaume Urvoy-Keller. Chkdiff: checking traffic differentiation at Internet access. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pages 57–58. ACM, 2012

4.1 Design of the tool

The strength of ChkDiff lies on its ability to not depend on the kind of shaping technique affecting delays and losses in use by ISPs and on the applications (or rather, traffic flows) that might be targeted. We achieve this by implementing the following design ideas in the core of our tool:

- *Use of real user traffic.* We conduct all experiments with previously dumped user traffic, so that the results yielded by our tool will refer to the exact set of applications run by the user.
- *Trace is left (almost) intact.* This ensures that any shapers traversed by our trace will have the same behaviour they would have if the packets had been generated by their respective applications. As we will see in the following subsections, the only modifications applied to packets are in the TTL field, in order to hit the router(s) at the desired hop, and in the application payload, in which we enforce the same size on all packets so as to avoid different transmission times.
- *Baseline for comparison is the entire traffic.* By the definition of network neutrality, a flow that is not differentiated will be treated in the same way as the rest of the (non-differentiated) traffic by any given router. On the other hand, a flow that is differentiated by a shaper will typically display higher delays or losses, depending on the scheduling and buffer management techniques in use.[9] When compared to the delays and losses of the rest of the trace, this flow will stand out. Our statistical analysis is based on that.

We will show in the validation section that we are able to successfully detect shaping when over half of the traffic is differentiated. The execution of ChkDiff is summarized in Algorithm 1.

A traffic trace is captured during a user's regular Internet activity; it is then processed and arranged into flows. For each hop h

[9] More specifically, a shaper will still be able to classify flows as if they were coming from their respective original applications, when it does so by inspecting IP, transport-layer header fields or application payloads. If it implements stateful TCP flow analysis, our replayed trace would probably bypass it. After this flow identification phase, a shaper will apply a differentiation technique to the selected flows. ChkDiff is able to detect techniques resulting in higher delays and losses; techniques that interfere directly with the transport or application layer, such as TCP RST injection and HTTP redirections, will not be detected.

Algorithm 1 ChkDiff execution

```
1: Capture user traffic
2: for each hop h ∈ {1, 2...k} do
3:     for each run r ∈ {1, 2, 3} do
4:         shuffle trace
5:         replay trace with TTL ← h
6:         collect ICMP time-exceeded replies
7:     end for
8:     detect shaped flows at hop h
9: end for
10: aggregate results and locate shaper(s), if any
```

we intend to test, we shuffle the trace so as to minimize any bias in the network conditions that our flows will experience: we keep the ordering of packets within each flow and modify the global packet ordering to be resilient to side traffic. We set the TTL field of the IP header of each packet to h and we replay the trace. Routers at hop h, if responsive, will reply with ICMP time-exceeded error messages, through which we compute single packet Round-Trip Times (RTT's) to hop h. Any shaper located between the user and hop h must have affected packets belonging to the flows it is configured to differentiate, before the ICMP error messages were elicited. We repeat this operation 3 times for the same hop in order to filter out false positives and we claim that a flow has been shaped when it has been rejected in our statistical analysis across all three runs. Once all the first k hops have been tested, we compare the results and attempt to localize the shapers.

In the rest of this section, we will describe each of the above steps in detail.

4.1.1 *Traffic trace*

The first action taken by ChkDiff is to dump outgoing user traffic while the user runs her usual network applications. This is the trace that will be replayed from the end user host towards routers at the hops nearby in the following steps. Since we focus in this chapter on the upstream direction, we expect the user to execute applications generating some non-trivial outgoing traffic that is not limited to HTTP requests or TCP ACK's: for example media upload, VoIP, file sharing and instant messaging.

4.1.2 *Flows and trace preparation*

Trace classification into flows. The packets in the dumped trace are arranged into 5-tuple flows, that is to say according to source and destination IP addresses, source and destination port numbers and transport protocol.

Fixed-size packets. Next, we need to prepare the trace we have to replay. Packets of different size, if sent along the same path to the same destination, will inevitably have different transmission times. As we will see in Section 4.2, this is a non-negligible source of error if, as it is in our case, we make the assumption that the delays of all packets going along the same path should be comparable. This is especially true if we measure delays to the closest hops, where the delay variability could be low enough to be comparable to or even smaller than the difference in transmission times between small and large packets. In order to overcome this, we force every packet of our trace to be of the same size S (in Bytes), with S being the size of the largest packet in our trace, by adding random padding at the end of packets with shorter payloads. Through this, we preserve all original payloads so that packets can still be intercepted by shapers implementing Deep Packet Inspection (DPI).

Shuffling packets. Before replaying our trace, an additional step is required. Since our analysis will be in terms of flows, we have to ensure that they all see the same network conditions while being injected into the network. It is therefore necessary to shuffle packets so that they exhibit such property. We assign a weight to each flow in our traffic according to its original size in packets and normalize it by the sum of all flows sizes, such that all weights sum to 1. For a trace with f flows, any flow i with size s_i, in number of packets, will have weight $p_i = s_i / \sum_{k=1}^{f} s_k$. A queue is thus created for each flow, where the per-flow packet order is maintained, since it might reveal useful information to a shaper for flow identification. We now pick packets randomly from each queue according to the flow weight and put them aside, ready to be replayed. Whenever a queue becomes empty, its weight is set to 0 and weights of all other flows are updated accordingly, so that they always sum to 1. By popping packets from each queue in the above fashion, we obtain for every flow an ordered sequence of 0's and 1's indicating whether a packet in the resulting shuffled trace came from that flow or not. Given a flow i, such sequence of 0's and 1's can be seen as a Bernoulli process with a probability equal to the weight of flow i, let us say p_i. Now, if we consider the spacing (or inter-packet time) W_i between any two consecutive packets from flow i, we can see that it follows a geometric distribution with parameter p_i: $P(W_i = w) = (1 - p_i)^{w-1} p_i$. The geometric distribution being the discrete version of an exponential distribution, packets of flow i see the real network conditions according to the PASTA property (Poisson Arrivals See Time Averages).[10] As this property applies to all flows, it enables us to reach our initial goal: letting all flows observe the same network conditions, provided that the network offers a stationary service.

Furthermore, shuffling is particularly useful when having to counteract side traffic and ICMP rate limitation, as we will see shortly.

[10] Ronald W Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2):223–231, 1982

4.1.3 Replay

ICMP rate limitation. In Chapter 3 we studied the responsiveness of routers to TTL-limited probes. Through a large measurement campaign, we examined possible bias in the Round-Trip Times of these probes and how ICMP rate-limitation is implemented on routers. We demonstrated that there did not appear to be any correlation between a slow or high probing rate (in the range [1, 4000] packets per second) and the resulting Round-Trip Times. In other words, even at high rates, we were not hitting any capacity limits that might have slowed down the generation of ICMP messages and contributed to the total packet delay. This is good news, since it tells us that the choice of probing rate does not mar the delays we obtain. There will definitely be a delay component due to the generation of the ICMP error message (estimated to be in the order of the submillisecond),[11] since it takes place in the router slow path, which is usually implemented in software instead of hardware and does not have high priority compared to other router operations. But this delay component will have roughly the same weight in all RTT's toward the same router and therefore will not constitute a source of error.

[11] Ramesh Govindan and Vern Paxson. Estimating router ICMP generation delays. In *Passive & Active Measurement (PAM)*, 2002

When using TTL-limited probes as in our case, we must also make sure that we obtain a sufficient number of replies, since it is a fairly widespread practice for manufacturers and network administrators to limit at a fixed maximum rate the responsiveness of routers to these expiring probes. We tested 850 routers from PlanetLab hosts up to hop 5 and demonstrated that ICMP rate limitation is implemented as an on-off process with typical values in $[20, 500]$ packets per second (*pps*). In light of this, the shuffling technique presented above has the undoubted advantage that unanswered probes would be spread fairly evenly across flows, since flow packets themselves are spread evenly across the trace. A non-shuffled trace replayed to an ICMP rate-limiting router would instead incur more variable losses among flows, which would inevitably impair any loss analysis. We will discuss the robustness of our tool to ICMP rate limitation in Section 4.3.

Testing the first k hops. In order to locate the position of a shaper, we need to replay the shuffled trace as many times as the number of hops we want to test, by increasing the IP TTL of all packets at every experiment. For the choice of k, a value of 3 or 4 should suit most cases and provide a large enough picture of what happens at the user's Internet access, including ISP routers and those right after the ISP boundaries. The user trace being made of flows with different IP destination addresses, testing routers that are further away is of increasing complexity due to a reduction in terms of samples per router as we move away from the user.

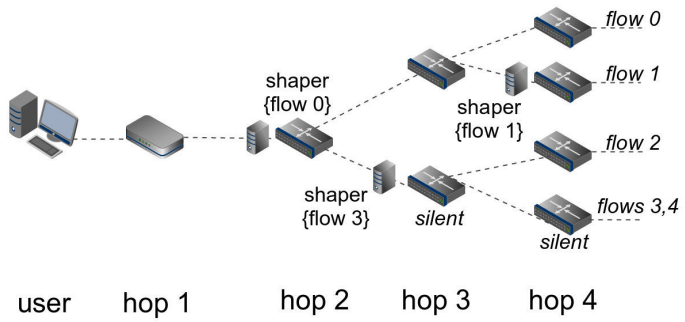


Figure 4.1: An example with shapers at different hops affecting selected flows.

	hop 1	**hop 2**	**hop 3**	**hop 4**	**shapers?**
flow 0	✓	✖	✖	✖	hop 2
flow 1	✓	✓	✓	✖	hop 4
flow 2	✓	✓	unresp	✓	none
flow 3	✓	✓	unresp	✖	hops 3,4
flow 4	✓	✓	unresp	unresp	not until hop 2

Figure 4.2: Expected output of ChkDiff for the network topology in Figure 4.1.

4.1.4 *Results analysis*

We focus our analysis on the study of Round-Trip Times and losses. In both cases, the approach is similar: we consider large flows only, that is those with at least 20 answered packets (a typical minimum sample size in statistical analysis), and we analyze these flows one at a time, comparing them against all the rest of the traffic as a whole (large and small flows indifferently). We reject a flow if it fails at least one between the delay and the loss analysis in all three experiments against the same router.

Delays. We compare the distribution of delays of a flow to the delay distribution of the rest of the trace using a statistical test. Our null hypothesis is that, in an environment without differentiation, if we sample the total set of delays obtained, they will all appear to be drawn from the same distribution as all the other delays of the trace. We conduct our analysis by applying two-sample one-sided Kolmogorov-Smirnov test, which has the benefit of being non-parametric, in that it does not make assumptions on the underlying distribution of the data it is checking. The test takes as statistic the maximum vertical deviation between the Cumulative Distribution Functions (CDF's) of two samples. We chose the one-sided version of this test because, while the two-sided Kolmogorov-Smirnov test looks for the maximum vertical deviation between two curves without including in its result whether this vertical distance was due to the first curve being above the second one or the other way around, the one-sided version looks for this deviation in one given direction. Applied to our scenario, we can test whether a flow experienced worse (i.e. larger) delays than the rest of the trace by checking whether its CDF lies below the CDF of its baseline, and to which extent.

Losses. In order to check if the loss rate of a flow differs significantly from that of the rest of the trace, we proceed by using an argument inspired from the binomial distribution. If we want to examine the losses (i.e. the number of unanswered packets) experienced by flow i, we let p be the loss rate of the rest of the traffic as a whole, and s_i be the original number of packets of flow i. If the loss events of flow i were not caused by a shaper, its number of losses l_i can be modeled as a binomial random variable of parameters $B(s_i, p)$. To test whether this holds true, we can approximate this binomial as a normal random variable of parameters $N(s_i p, s_i p(1-p))$ and verify that the loss events l_i lie within α standard deviations of the normal mean. With α being a function of the significance level we want to achieve, we check that $ps - \alpha\sqrt{p(1-p)s_i} < l_i < ps_i + \alpha\sqrt{p(1-p)s_i}$. The right side of this last condition is the one we are interested in, as it indicates - when it does not hold - that the flow experienced more losses than it should have, and it is what we check in our analysis.

Combined analysis. Since a shaper might cause longer delays or extra losses to affected flows, we combine the delay and the loss analyses described above and reject a flow when it fails at least one of them.

Repetition of experiments. Statistical tests are operated at a certain confidence level, which in our tool we set to 99%. Due to the high number of flows in a user trace, we are bound to have a number of false positives, whatever action we take. To work around this issue, we adopt a simple strategy. We repeat an experiment three times (at the same constant probing rate) to router(s) at the same hop-distance and claim that a flow has been shaped only when it was rejected in all three runs.

4.1.5 Results Aggregation

After collecting traces and analyzing delays and losses for the first k hops, we need to aggregate results in order to attempt to localize shapers, if ever a flow was rejected in any of those hops. A shaper positioned right before hop h, with $h \in \{1, 2, ...k\}$, will cause targeted flows to fail the delay or loss analysis (or both) on all hops $\geq h$. When ChkDiff detects this, it declares the flow as being shaped on the hop segment between h and the previous responsive router. We show an example with routers up to hop 4 in Figure 4.1. We assume that there is a number of non-differentiated flows from the user trace passing through each router besides the four shown in the figure, and that they contribute to the baseline for the statistical analysis. In Figure 4.2 we provide the expected output from ChkDiff based on this scenario. A shaper for flow 0 is deployed right before the router at hop 2: this means that flow 0 will pass the analysis at hop 1, but will not at all successive hops. Flow 1 is a similar case, but at the edge of the tested hops. At hop 3 an unresponsive router, that is to say a router that is configured not to reply to expiring packets,

generates a gap in our assessment, which might be compensated by the results at the next hop. If at the next hop a flow (flow 2) continues to pass the test, we can safely claim that it was not shaped along the whole path under consideration. If instead the flow fails the test, as we show for flow 3 in our example, we can only say that at hop 3 and 4 it encountered a shaper, without being more specific. Finally, if the next hop is also unresponsive, for a flow like flow 4, our conclusion is simply that no shapers were detected up to the last hop where the flow passed the analysis.

4.2 *Validation in a neutral scenario*

Before validating ChkDiff in the presence of traffic differentiation, it is important to justify some measures we take when replaying a user trace: forcing the same size in all packets and aggregating results across 3 experiments. The packet trace we used here and in Section 4.3 was captured in a time-window of 3 minutes of a typical Internet session, in which we performed picture uploading on a social network, browsing on a news site, and sent a few chat messages. The trace is made of 6733 packets, arranged into 275 flows, of which 61 are *large* (i.e. they have more than 20 packets) and comprise 76.8% of the total amount of packets. The exact distribution of flow sizes is shown in Figure 4.3.

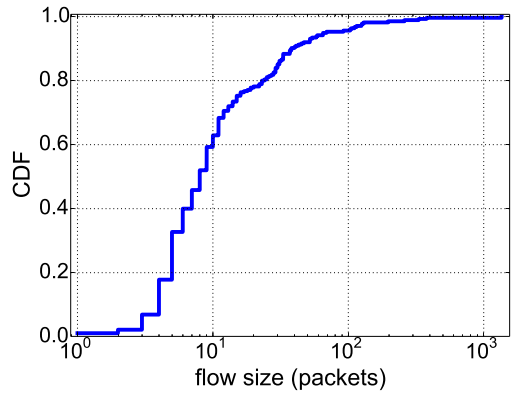


Figure 4.3: Distribution of flow sizes, in number of packets, for the packet trace used for validation.

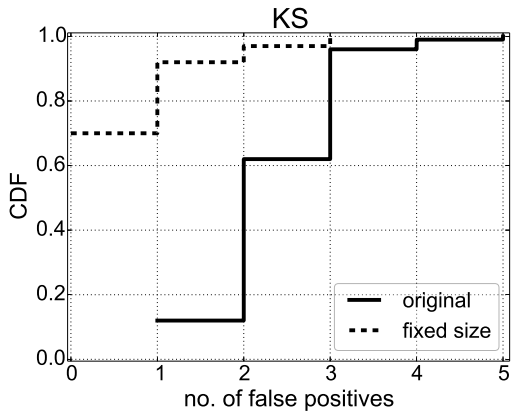


Figure 4.4: Incidence of false positives in the delay analysis when the replayed trace contains unaltered original packets, and when it contains packets of the same size. Results are over 1 run.

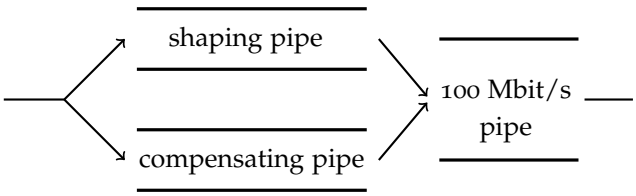


Figure 4.5: Middlebox configuration.

We claimed in Section 4.1.2 that, by fixing the packet size to a constant value for all packets in a trace, we were able to remove a considerable fraction of errors in the delay analysis. We now show the incidence of false positives when packets are the original size and when they are padded to a constant value. For each of the two options, we ran ChkDiff 100 times in a controlled setup with no differentiation towards a router under our control at hop 2. In Figure 4.4 we compare the CDFs of the number of false positives of the delay analysis for each experiment.[12] The improvement is evident: we remove all errors in 70% of experiments and are left with 30% of experiments showing mostly 1 false positive.

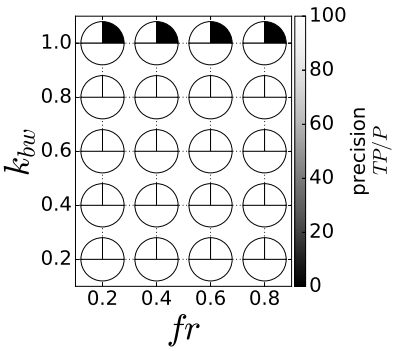
[12] We do not consider the loss analysis, since in our controlled setup the trace did not experience any losses.

The next step is to aggregate the results of multiple runs, as described in Section 4.1.4 and see if these errors disappear. We ran ChkDiff 100 times and observed indeed that no false positives emerged when considering just two runs. In order to have a safe margin of error, we use by default three runs in ChkDiff.

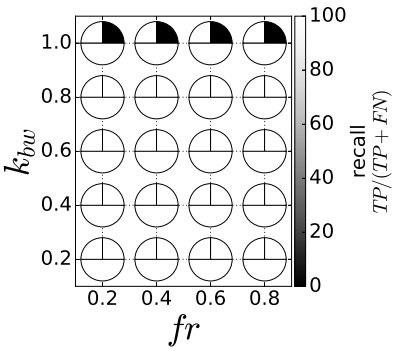
4.3 *Validation in a non-neutral scenario*

We tested how the tool copes with different shaping and network settings in a controlled experimental setup. We focused on two scenarios: Scenario 1, in which we throttle the bandwidth of selected flows, and Scenario 2, where we apply uniform packet drops. In our setup, a user machine is connected through cable to a middlebox, which operates both as a gateway and a shaper, and which is, in turn, connected to a Cisco router under our control, where our probes expire. In the middlebox, we deployed a shaper with Dummynet, a popular and versatile network emulation tool.[13] The configuration we used is depicted in Figure 4.5: incoming packets are directed to either the upper or lower pipe on the left side, depending on whether they belong respectively to the flows to shape or not. The upper pipe is traversed by all flows that we intend to shape; in Scenario 1 it has its own bandwidth *bw* and queue size, and in Scenario 2 it induces uniform losses at rate *lr*. The lower pipe compensates for the transmission delay produced by the upper pipe in Scenario 1: it adds this constant delay component to the packets of all non-differentiated flows, so that only the queueing delay in the upper pipe constitutes the discriminating factor between shaped and non-shaped packets. In scenario 2, it produces no effect. Finally, all packets meet at the pipe on the right-hand side, which emulates a 100 Mbit/s link. In Scenario 2 this pipe induces a uniform loss rate lr_{all} to all flows. All pipes are configured with a buffer size of 100

[13] Marta Carbone and Luigi Rizzo. Dummynet revisited. *SIGCOMM Comput. Commun. Rev.*, 40(2):12–20, April 2010

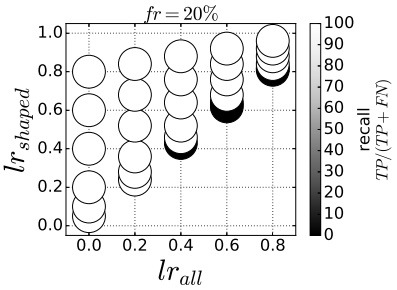


(a) Precision of the combined analysis

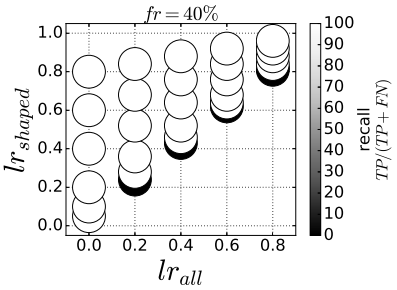


(b) Recall of the combined analysis

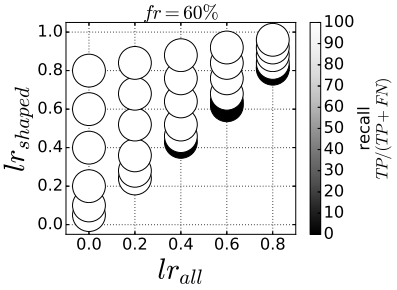
Figure 4.6: Precision and recall of the combined analysis, for the case of one shaped flow in Scenario 1. In each circle, we report in the upper-left quarter the result of only the delay analysis, in the upper-right quarter the result of only the loss analysis, and in the lower half the result of the *combined* analysis, which rejects a flow if either the delay or the loss analysis fails.



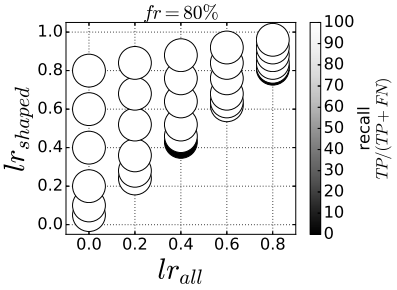
(a) Recall when $fr = 20\%$



(b) Recall when $fr = 40\%$



(c) Recall when $fr = 60\%$



(d) Recall when $fr = 80\%$

Figure 4.7: Recall of the loss analysis as we vary *fr*, for the case of uniform drops on the whole traffic and on *one* selected flow (Scenario 2).

packets and a droptail buffer management policy.

4.3.1 One shaped flow

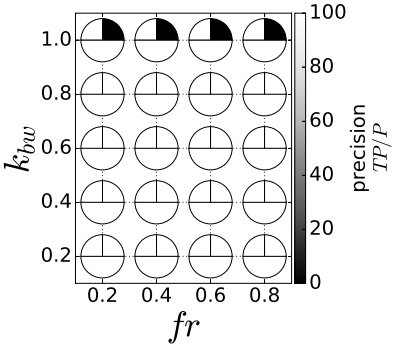
We start by examining a scenario in which only one flow is being differentiated by the shaper. We proceeded by taking the trace previously described and by adding an extra flow of which we varied the number of packets in order for it to be a fraction *fr* of the total amount of packets of the trace. This is the flow that will be targeted by the shaper. Our results are in terms of precision and recall, which show respectively the fraction of detected flows that we know are indeed shaped, and the fraction of shaped flows that are correctly detected.[14] Perfect performance translates into a precision and recall of 100%.

Shaping pipe (Scenario 1). In this configuration the bandwidth *bw* of the upper pipe on the left side is set as a function of the average send-

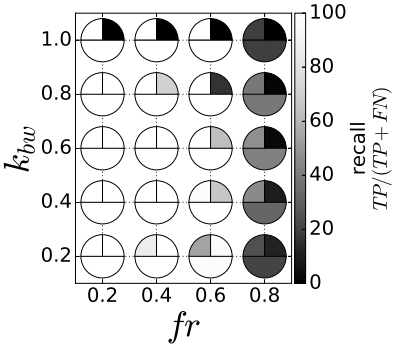
[14] We define *precision* as being the number of true positives (*TP*) over the number of positives (*P*), and *recall* as the number of true positives over the sum of true positives and false negatives (*FN*), that is to say over the number of flows that we know were shaped. More details can be found on `http://en.wikipedia.org/wiki/Precision_and_recall`

ing rate r (in bits per second) of the shaped flow, computed before the experiment begins. We chose $bw = k_{bw}r$, so that a fraction $k_{bw} \in (0, 1]$ of packets of the shaped flow would use all the available pipe bandwidth bw and the rest would queue up. We ran ChkDiff 3 times for each combination of k_{bw} and fr, with $k_{bw} \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $fr \in \{0.2, 0.4, 0.6, 0.8\}$. These are the values we used also in all the following experiments in Scenario 1. The results are provided in Figure 4.6, where we chose a compact representation in which each circle shows in the upper-left quarter the result of the delay analysis, in the upper-right quarter the result of the loss analysis and in the lower half the result of the combined analysis. Since in this scenario a shaping pipe causes queueing delays and, in case its queue fills up, drops packets, we directly evaluate the benefits of combining the two analyses in such setting. In this basic scenario, we see that the combined analysis manages to always identify the shaped flow. At $k_{bw} = 1$ we observe that the flow still experienced some queueing, as a result of the pipe bandwidth being a function of the *average* sending rate of the flow and not of its instantaneous rate.

Uniform drops (Scenario 2). Our goal in Scenario 2 is to verify to which extent ChkDiff manages to identify a shaped flow, when losses affect a selected flow and the entire traffic at different rates. We configured the shaper so that the upper pipe in Figure 4.5 drops a fraction lr of the packets of the flow to shape, and the pipe on the right-hand side, where all traffic goes, has a drop rate lr_{all}. We varied again the size of the targeted flow and ran experiments with $fr \in \{0.2, 0.4, 0.6, 0.8\}$. For this and all the following experiments in Scenario 2, we chose $lr \in \{0.05, 0.1, 0.2, 0.4, 0.6, 0.8\}$ and $lr_{all} \in \{0.2, 0.4, 0.6, 0.8\}$. We do not include the graphs on precision, since all results show a precision of 100%, which means that we never encountered any false positives or, in other words, all the flows detected by ChkDiff as being shaped were indeed shaped. On the other hand, some false negatives (i.e. shaped flows that go undetected) did occur, so our analysis will focus on recall. In Figure 4.7 we present the results for this scenario. We only show the results of the loss analysis, since in this scenario delays are not affects. On the X-axis we plot the loss rate lr_{all} for all packets of the trace, whereas on the Y-axis we show the overall loss-rate lr_{shaped} experienced by the shaped flow: $1 - (1 - lr)(1 - lr_{shaped})$. The tool achieves 100% recall in all cases except those in which, with a low fr and a fairly high ($\geq 40\%$) overall loss rate, the loss rate of the shaped flow is close to the global one. The added loss rates on the lower diagonal of the graphs correspond to $lr = 0.05$, which could be too low to be noticeable on samples of relatively small size. In all other cases, the tool correctly identified the shaped flow.

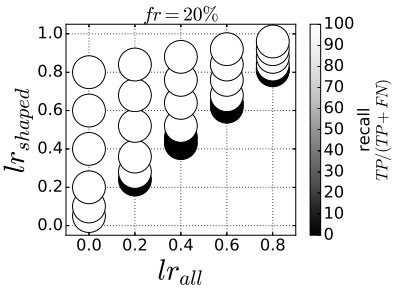


(a) Precision of the combined analysis

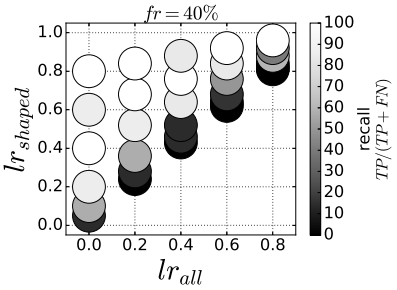


(b) Recall of the combined analysis

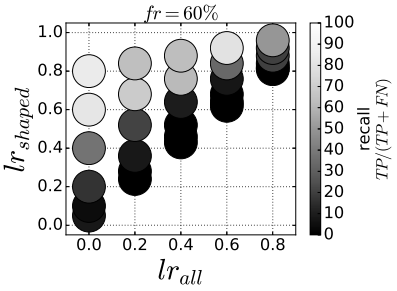
Figure 4.8: Precision and recall of the combined analysis, for the case of *multiple* shaped flows, in Scenario 1.



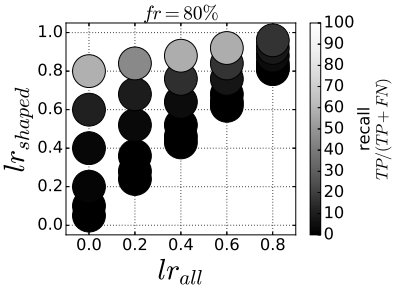
(a) Recall when $fr = 20\%$



(b) Recall when $fr = 40\%$



(c) Recall when $fr = 60\%$



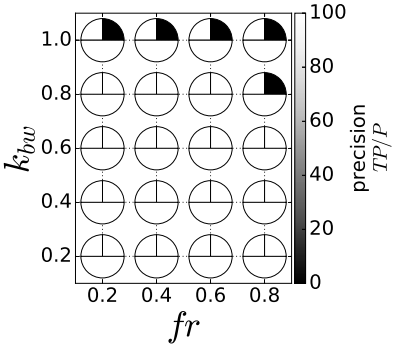
(d) Recall when $fr = 80\%$

Figure 4.9: Recall of the loss analysis as we vary *fr*, for the case of uniform drops on the whole traffic and on *multiple* selected flows (Scenario 2).

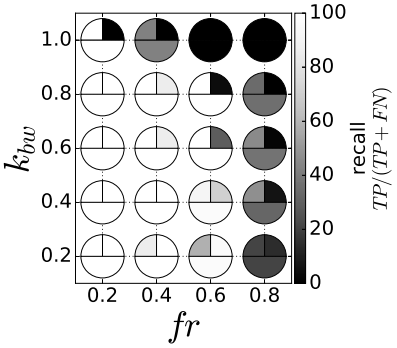
4.3.2 *Multiple shaped flows*

We now move to a more complex scenario, in which multiple flows are being targeted by the shaper. In order to select which flows to shape, given a fraction *fr* of the trace size to differentiate, we iteratively picked the flow whose size (in Bytes) was the closest to the target *fr*, until the total amount was reached.

Shaping pipe (Scenario 1). We set the bandwidth *bw* of the shaping pipe as a function of the average sending rate of all packets belonging to the flows to shape. All shaped flows pass through the same shaping pipe so as to be able to compensate for one transmission time only in the lower pipe. We present the results for this scenario in Figure 4.8. While the precision reached appears to be optimal, the recall plots show that, quite expectedly, when the shaped flows amount to a large fraction of the trace, the baseline for comparison becomes too weak for the test to work.

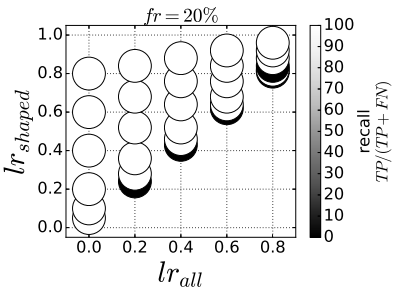


(a) Precision of the combined analysis

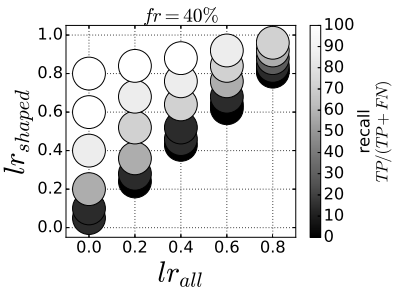


(b) Recall of the combined analysis

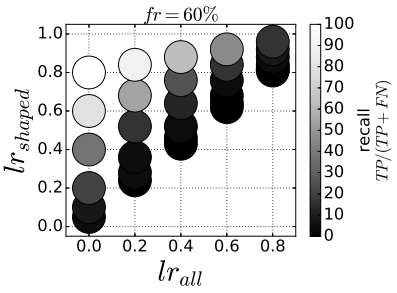
Figure 4.10: Precision and recall of the combined analysis, for the case of *multiple* shaped flows with a WiFi connection, in Scenario 1.



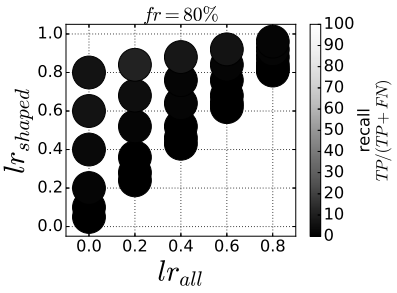
(a) Recall when $fr = 20\%$



(b) Recall when $fr = 40\%$



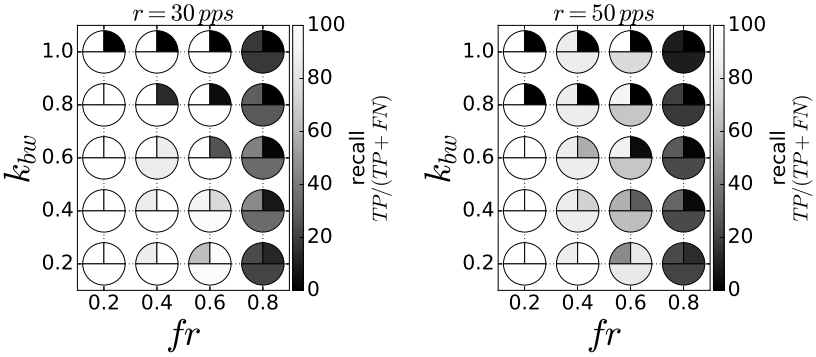
(c) Recall when $fr = 60\%$



(d) Recall when $fr = 80\%$

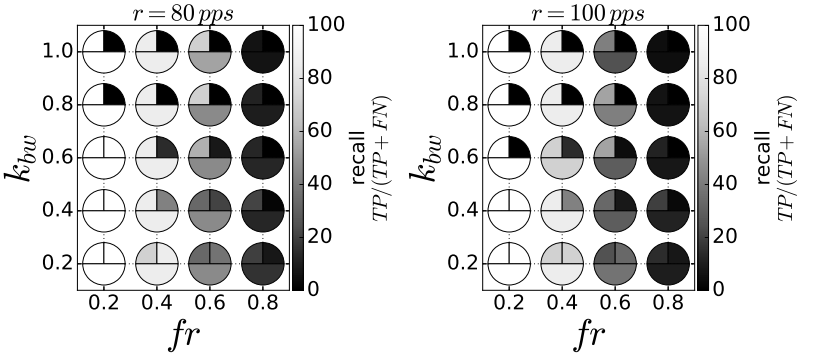
Figure 4.11: Recall of the loss analysis as we vary *fr*, for the case of uniform drops on the whole traffic and on *multiple* selected flows with a WiFi connection (Scenario 2).

Uniform drops (Scenario 2). In the scenario with uniform drops, we used a different shaping pipe for each flow to differentiate, in order to have the same loss rate *lr* for all shaped flows. Results are provided in Figure 4.9, where we omitted the plots on precision, since it reached the optimal value (100%) for all combinations of parameters. In the four subplots, we are mostly interested in the area with $lr_{all} \in [0.0, 0.2]$, as it best represents a realistic setting: in a network with no (0%) or relatively high (20%) packet drops, a shaper causes losses of various degrees to part of the traffic passing through it. For completeness, we also show cases with higher losses and a large fraction of affected traffic. When 20% of the traffic is targeted, we are able to detect all differentiated flows, except when the shaped flows experience just 5% more of losses, on top of the 20% overall loss rate of the trace. We observe that, as we shape an increasing fraction of the trace, the loss analysis significantly degrades. The reason is that, with several flows being differentiated, the baseline will necessarily



(a) Recall of the combined analysis, $r = 30\,pps$

(b) Recall of the combined analysis, $r = 50\,pps$



(c) Recall of the combined analysis, $r = 80\,pps$

(d) Recall of the combined analysis, $r = 100\,pps$

Figure 4.12: Scenario 1 with multiple shaped flows and ICMP rate limitation at 20 *pps*.

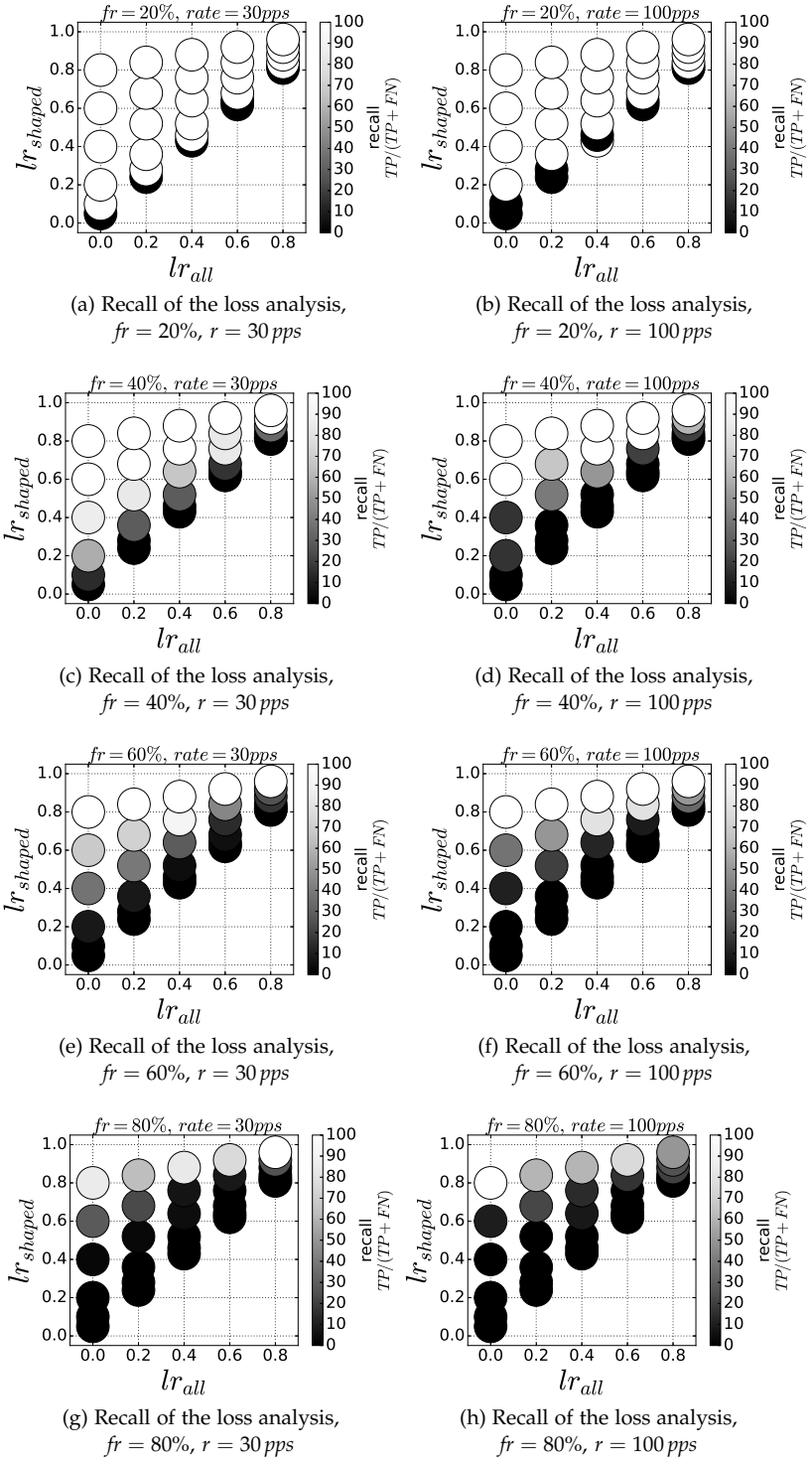
include more and more shaped flows and the statistical analysis will be impacted. However, this constitutes an extreme case for our tool and it is unlikely to be encountered in practice.

4.3.3 *Multiple shaped flows over WiFi*

We repeat the same experiments as in the previous section, with multiple shaped flows, but we use a WiFi connection between the client and the middlebox in order to stress the delay analysis. Figures 4.10 and 4.11 show the results for respectively Scenarios 1 and 2. In both cases, the results are qualitatively similar to the previous setup, where we used a wired connection.

4.3.4 *ICMP rate limitation*

We also wanted to verify how resilient our analysis is when we encounter a router that implements ICMP rate limitation. We tested ChkDiff in the same wired experimental setups as before and configured the router to respond at most at 20 *pps* (with a burst size of 20 packets and a period of 1 second), a common setting we found for Cisco routers, as seen in Chapter 3. We repeated the experiments of Scenarios 1 and 2 at different sending rates (30, 50, 80 and 100 *pps*) higher than the ICMP rate limitation threshold. Our aim is to stress our tool when an additional source of losses is present and see how high our sending rate r can be, with respect to the rate limitation



(a) Recall of the loss analysis, $fr = 20\%$, $r = 30\,pps$

(b) Recall of the loss analysis, $fr = 20\%$, $r = 100\,pps$

(c) Recall of the loss analysis, $fr = 40\%$, $r = 30\,pps$

(d) Recall of the loss analysis, $fr = 40\%$, $r = 100\,pps$

(e) Recall of the loss analysis, $fr = 60\%$, $r = 30\,pps$

(f) Recall of the loss analysis, $fr = 60\%$, $r = 100\,pps$

(g) Recall of the loss analysis, $fr = 80\%$, $r = 30\,pps$

(h) Recall of the loss analysis, $fr = 80\%$, $r = 100\,pps$

Figure 4.13: Scenario 2 with multiple shaped flows and ICMP rate limitation at 20 *pps*.

implemented on the router side, while still minimizing errors.

In Figure 4.12 we show the recall plots of the combined analysis in the case of Scenario 1, where a shaping pipe throttles the bandwidth of multiple selected flows. Since ICMP rate limitation only causes packet drops, it is no surprise that the delays are no more affected than they were in the previous case when the router was fully responsive (Figure 4.8 (b)). We omit here and in the next scenario the results for one shaped flow, since they always showed maximum

precision and recall.

We conducted again the experiments of Scenario 2, where uniform drops are applied to selected flows and to the whole traffic with different probabilities, and we provide the results in Figure 4.13. For constraints of space, we only show the results for the minimum and maximum probing rates we considered: 30 and 100 *pps*. We observe that the loss test experiences considerable degradation only in the extreme case of multiple shaped flows corresponding to 80% of the trace. Most importantly, varying the probing rate from 1.5 (30 *pps*, with a router-induced loss rate of 33%) to 5 times (100 *pps*, with a router-induced loss rate of 80%) the ICMP rate limitation threshold of the router does not appear to alter significantly the results.

4.3.5 A more complex scenario

In a realistic setting, if a user dumps her own traffic while some TCP flows are being targeted by a shaper that throttles their bandwidth, the sending rate of these flows inside the captured trace will already have been reduced by the shaper. If ChkDiff replays this trace at its original sending rate, the TCP flows that were previously throttled will now comply with the shaper's policies and will not of course experience any further degradation. It is therefore important to scale up our probing rate with respect to the original one, in order to be able to trigger and detect the presence of a shaper.

We set up a wired scenario with a shaper, ICMP rate limitation and cross traffic. In the same way as in Section 4.3.1, we created a flow constituting 20% of the total trace size and injected it in our trace so that it would be evenly spread out and have consequently a constant sending rate r_{flow}. The shaper was configured as in the previous case of a shaping pipe affecting one flow only, and its bandwidth was set to r_{flow}. The router activated ICMP rate limitation at 50 *pps*, a common value for Juniper routers, as we showed in Chapter 3. Finally, we added some cross traffic flowing on average at 20% of our sending rate and implemented as a series of bursts of ping packets following a Poisson process. We configured the bandwidth of the pipe on the right-hand side in Figure 4.5 to be equal to the sum of the rate of the trace and of the cross traffic. This way, the whole trace also experiences queueing.

In this setup, we assess whether a shuffled trace replayed at a constant rate is indeed more robust to transient network conditions than the original trace replayed as it is. We increased the probing rate r by a factor of 1.5, 2, 4, 8 and 16 times the original probing rate r_{orig}

		Rate					
		1x	**1.5x**	**2x**	**4x**	**8x**	**16x**
Original trace	True Positives	0	1	1	1	1	1
	False Positives	11	3	3	1	2	1
Shuffled trace	True Positives	0	1	1	1	1	1
	False Positives	0	0	0	0	0	0

Table 4.1: Number of true and false positives when replaying the original and a shuffled trace at different sending rates.

of the trace (with $r_{orig} = 64\,pps$) and counted in Table 4.1 the number of false positives of the delay analysis across 3 runs. We see that, even though in both cases we correctly identify the shaped flow already at $1.5x$, we never encounter any false positives when replaying a shuffled trace. With the original trace, on the other hand, we always obtained some false positives; their number seems to decrease with high probing rates only because the amount of flows with sufficient samples also decreases.

4.4 Assessment with respect to existing methods

A number of tools for the detection of traffic differentiation have been proposed in the literature in the past few years, as detailed in Chapter 2.

A work that has some aspects in common with the upstream experiment of ChkDiff is NetPolice,[15] where the authors were able to detect differentiation in backbone networks with the use of TTL-limited probes. Using synthetic traces made of HTTP, peer-to-peer, BitTorrent and other application flows, they probed ingress and egress routers of backbone ISPs from a large set of PlanetLab nodes in order to notice any difference in loss rates along the same path segment: if any difference was observed, they tried to attribute it to content-based differentiation (with the HTTP flow as baseline) or, when the discrepancy was between different IP sources or destinations, to routing-based differentiation. Our approach does leverage TTL-limited probes, but it is client-oriented (in a traceroute-like manner), it focuses on the user's access ISP, and does not make assumptions on which flows should be considered as non-differentiated in its analysis.

[15] Ying Zhang, Zhuoqing Morley Mao, and Ming Zhang. Detecting traffic differentiation in backbone isps with netpolice. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 103–115. ACM, 2009

One of the first tools presented to the scientific community was BT-Test,[16] which checks for TCP reset packets injected by ISPs during the replaying of a typical BitTorrent packet exchange between a user and controlled server. Its aim was to disclose a practice that had been recently reported by some U.S.-based users. The same authors later proposed a more comprehensive tool, Glasnost,[17] that compares the maximum throughput of an application flow (e.g., BitTorrent, YouTube, etc) to that of a control flow whose packets are the same as in the application flow except for their payload, which is randomized. The packets of the two flows are interleaved so as to experience the same network conditions and are replayed to a server. This technique expects traffic differentiation to happen at the application layer by means of deep packet inspection, and to result in lower throughput for the affected application. ChkDiff is also able to detect such cases, since a lower throughput is the result of higher packet delays, but we do not make the assumption that a shaper targets a specific application and that it discriminates according to packet payloads.

[16] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. Detecting BitTorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC'08)*, Vouliagmeni, Greece, October 2008

[17] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna Gummadi, Ratul Mahajan, and Stefan Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, Apr 2010

A tool that also focuses on a specific application and control flow is DiffProbe,[18] which attentively analyzes the delay and loss distributions of the two flows during a replaying phase at the normal

[18] Partha Kanuparthy and Constantine Dovrolis. Diffprobe: detecting ISP service discrimination. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 1649–1657, Piscataway, NJ, USA, 2010. IEEE Press

application sending rate and during a replaying phase at a higher rate, which attempts to create congestion at possible shapers along the path. The control flow is crafted much in the same way as previously described, with the addition of transport layer fields such as port number being modified in order to bypass shapers. This tool was soon followed by ShaperProbe,[19] which assumes that differentiation happens through a token bucket and tries to infer its parameters (buffer size and processing rate). It sends at the path capacity trains of packets back-to-back to a server and, if they traverse a shaper, it expects to observe a level shift in the received rate at the destination. While both methods undoubtedly provide more insight than ChkDiff on the characteristics of shapers, they analyze the behaviour of one application at a time and, even if in principle they can adapt to any application, they are in practice limited to the packet traces provided with the executables (i.e. Skype, in this case). Packsen[20] has a similar approach to DiffProbe, but it improves on it by using a less computationally expensive statistical analysis in order to infer the shaper type and parameters.

[19] Partha Kanuparthy and Constantine Dovrolis. Shaperprobe: End-to-end detection of isp traffic shaping using active methods. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 473–482. ACM, 2011

[20] Udi Weinsberg, Augustin Soule, and Laurent Massoulié. Inferring traffic shaping and policy parameters using end host measurements. In *INFOCOM*, pages 151–155, 2011

Nano[21] differs from existing solutions in that it carries out passive measurements on user traffic and compares it against a data set of other users in the same geographical area, with comparable machine setups, at the same time of the day, but connected to a different ISP. While this method is undeniably independent of user applications and differentiation techniques, its main disadvantage is that it needs a fairly large number of users for it to be operational.

[21] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. Detecting network neutrality violations with causal inference. *ACM SIGCOMM CoNext*, page 289, 2009

More recently, a theoretical framework for the inference and localization of neutrality violating links has been proposed.[22] After conducting measurements from different vantage points traversing the same links, it builds a linear system of equations in the same fashion as in network performance tomography. When the network is neutral, such system is supposed to be solvable and it infers properties of the links. When instead a link is not neutral, the measurements are inconsistent and the system unsolvable. The deployment of such method would require a large and diverse user base, where several vantage points perform measurements on the same set of paths and send the results to a central server, which would process the data and infer differentiation. Our approach is instead confined to the network performance experienced by the end user who runs the tool: no aggregation of results across users is necessary.

[22] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network neutrality inference. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 63–74, New York, NY, USA, 2014. ACM

4.5 Summary

In this chapter we presented the upstream experiment of ChkDiff, a novel tool for the detection of traffic differentiation at the Internet access. After replaying the user outgoing traffic to the routers at the first few hops, the tool applies a statistical analysis to delays and losses in order to infer whether any of the replayed flows experienced degraded performance. We validated ChkDiff in a controlled environment with different setups and showed its robustness to ICMP

rate limitation.

In the next chapter, we extend ChkDiff so that it includes an experiment for downstream traffic, which we shuffle and replay to the user from a dedicated measurement server in order to detect differentiation.

5
ChkDiff: the downstream experiment

We saw in the previous chapter that ChkDiff directly addresses the problems of scalability to different user applications and of applicability to different shaping techniques affecting delays and losses. We achieve this by performing active measurements with the real user traffic, comparing the performance of a flow against the performance of the rest of the replayed traffic, and by analyzing for each flow its delays and losses, which reflect any alteration introduced by a shaper inside the network.

This chapter complements Chapter 4, in which we focused on the user's upstream traffic and replayed it with low TTL values in a traceroute-like manner against the routers at the first few hops away from the user in order to detect differentiation and localize shapers. We extend this with a new experiment in the downstream direction, where we replay the user's incoming traffic from a server, measure one-way delays and losses, and check for differentiation on a per-flow basis. We describe in details the measures taken by ChkDiff in order to successfully deliver the replayed trace and validate the tool in two differentiation scenarios, with the server located in three different data centers, and over wired, WiFi and 3G connections.

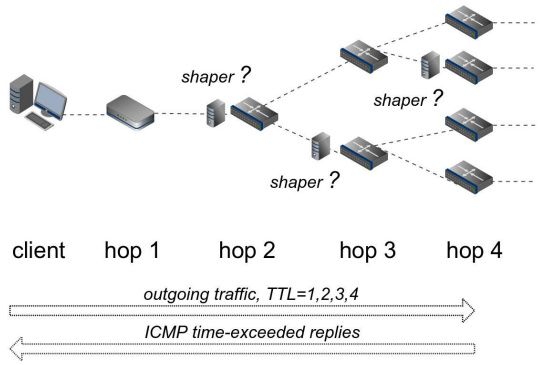
The chapter is organized as follows: in Section 5.1 we provide a detailed description of the methodology we used in ChkDiff; in Section 5.2 we validate the tool; we discuss our method in Section 5.3 and assess it with respect to related work in Section 5.4; we give closing remarks in Section 5.5.

We presented the results of this chapter in a paper published at ITC 28.[1]

[1] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Testing for traffic differentiation with chkdiff: the downstream case. In *Teletraffic Congress (ITC 28), 2016 28th International*. IEEE, 2016

5.1 Methodology

The design of a new tool for the detection of traffic differentiation has to necessarily consider two weak points of existing methods: the difficulty to scale to different applications and the limitation to specific differentiation techniques. As illustrated in Chapter 4, we overcome this in three steps: *a*) we use the real traffic of a user and not a synthetic trace; *b*) we minimize the modifications to the trace needed for the experiment to work and *c*) we analyze the performance of a flow in terms of delays and losses with respect to the rest of the trace in



(a) Upstream experiment

(b) Downstream experiment

Figure 5.1: The two experiments in ChkDiff.

order to infer neutrality violations: these two metrics alone are able to capture the effect of shapers at the IP layer.

A complete run of ChkDiff consists of two experiments, one that replays the user's outgoing traffic (upstream direction) to the routers at the first few hops away from the user and one that replays the user's incoming traffic (downstream direction) from a measurement server to the user. We report in Algorithm 2 an outline of a full execution of ChkDiff.

At first the tool dumps client traffic for a time window of typically 3-5 minutes, while the user is asked to run the applications and services of an Internet session she wishes to test. Next, packets are grouped into 5-tuple flows (source and destination IP addresses, transport protocol, source and destination port numbers), which are further arranged into an outgoing trace, $trace_{out}$, and an incoming one, $trace_{in}$. We now briefly recall the upstream experiment, fully described in Chapter 4, and then go on to illustrate in detail the methodology for the downstream case.

5.1.1 *Upstream experiment, in a nutshell*

Non-trivial outgoing traffic that an access ISP might want to differentiate includes media uploading, P2P file sharing and VoIP; a run of ChkDiff in the upstream direction should ideally test at least one type of such traffic. Before conducting the actual experiment, we shuffle $trace_{out}$ in such a way that the position of the packets of each flow inside the trace follows a Poisson process, so that according to the PASTA property (Poisson Arrivals See Time Averages),[2] each flow will see the same network conditions when the trace is replayed.

[2] Ronald W Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2):223–231, 1982

The order of packets within each flow is preserved. As in the first half of Algorithm 2, we consider the first few hops away from the user, at or in proximity of her access ISP network (up to hop 3 or 4, as in Figure 5.1(a)), and for each hop h we replay $trace_{out}$ at a constant sending rate higher than the original one and with a modified IP TTL set to h, so that the trace will expire on the router(s) at hop h and generate ICMP time-exceeded messages. We showed the validity of this ICMP feedback in Chapter 3, along with its robustness to ICMP rate-limitation. Each flow having its own set of Round-Trip Times (RTT's) and losses, we can now compare its performance up to a given hop to that of the rest of the flows along the same path. We use Kolmogorov-Smirnov test to analyze delays and a binomial-inspired test to analyze losses. By aggregating the results of each flow across consecutive hops we are able to infer the presence of a shaper and localize it in terms of number of hops from the client.

5.1.2 *Downstream experiment*

In this chapter, we present the downstream version of ChkDiff and validate it experimentally. In brief, the experiment in the downstream direction consists in taking all necessary measures to replay the original incoming flows, having the server replay the trace to the client and finally analyzing the results in a way that takes into account the possibility of having multiple paths to the client (Fig-

Algorithm 2 ChkDiff complete execution

```
Capture user traffic, store into trace_out and trace_in
▷ Upstream experiment
for each hop h ∈ {1, 2...k} do
    for each run r ∈ {1, 2, 3} do
        shuffle trace_out
        replay trace_out with TTL ← h
        collect ICMP time-exceeded replies
    end for
    detect shaped flows at hop h
end for
aggregate results and locate shaper(s), if any
▷ Downstream experiment
for each run r ∈ {1, 2, 3} do
    shuffle trace_in
    for each flow f in trace_in do
        find NAT mapping for f
        initiate connection from client
    end for
    replay trace_in from server to client
    compute one-way delays and losses
end for
detect shaped flows
```

ure 5.1(b)).

The second half of Algorithm 2 outlines the main steps of the downstream experiment, which we describe in details in the remainder of this section. First, we need to shuffle $trace_{in}$ in the same way we did in the upstream experiment. This allows us to eliminate cross traffic noise from the effects of possible neutrality violations. Then, for a replayed flow to be able to successfully reach the client, we need to deal with possible Network Address Translation (NAT) and firewall devices a user might be behind and also other possible middleboxes that might be deployed along the path from the server. After all connections are initiated from the client side, the server replays the shuffled $trace_{in}$ to the client at a rate higher than the original one. We compute One-Way Delays (OWD's) for each flow and note the number of losses, if any. In order to infer differentiation, we run a clustering analysis on delays so that we can distinguish when different delay distributions are due to shaping and when they are due to a variety of paths. Lastly, a test on flow losses completes the analysis. We elaborate now on each of the above actions.

Getting ready for replaying. As opposed to the upstream experiment (Chapter 4), we do not truncate or pad packets in $trace_{in}$ to a fixed size. In the upstream experiment the low variability of the total delay along a short wired path is in the same order of magnitude as the variability of the transmission delay, which is proportional to the packet size. That makes it impractical to replay packets exactly as captured, since flows with large packets over a wired connection experience larger delays solely because of their packet size. For the downstream case, we examine a much longer path in terms of hops and delays, and such source of error is canceled out by the inherent delay variability along a larger path. Therefore we are able to replay the packets with their original payloads, as seen by the client upon receiving them.

Replaying incoming traffic from a single source (i.e. our server) means that we cannot keep the original source IP addresses of the user trace, for two main reasons. Firstly, most access networks today are configured to drop outgoing packets with a source address that does not belong to the address space of the access network itself. In other words, they do not allow IP address spoofing.[3] Secondly, as we will see shortly, if a NAT device is present, we can replay a flow only if we find the mapping applied by the NAT to that flow for external endpoints. Since we are not in control of other endpoints (that is to say, all network applications or services run by the user) than our measurement server, we need to overwrite the IP source address of each packet in $trace_{in}$ with the IP address of the server; original port numbers are retained. Conversely, in the upstream experiment the original source and destination IP addresses are preserved. However, even if in the downstream case we lose the ability to reveal shapers based on the source IP address, we can combine upstream and downstream experiments to overcome the limitations of both.

[3] BCP 38 - Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. URL: `https://tools.ietf.org/html/bcp38`

Furthermore, the commercial shapers studied in a recent work[4] do not use this piece of information to classify flows.

After being shuffled, $trace_{in}$ is sent via FTP to the server. While we deploy a server with a public IP address, listening on a known port, a client is likely less easy to reach: a few network elements need to be considered before we can replay the trace to the client.

- **NAT's.** In today's networks, where IPv4 addresses are running out, deploying a NAT device has become a widespread practice. For our purposes, this means that the client's view of a flow might not be the same as the server's. Since the trace to replay is originally as seen by the client, we might need to modify the destination IP address and port number in order to reach the client from an external endpoint (Figure 5.1(b)). NAT devices are usually defined according to how they map the same source pair $X{:}\,x$ of IP address X and port number x of an internal network when different external destination IP addresses and port numbers are reached (for instance $Y_1{:}\,y_1$ and $Y_2{:}\,y_2$).[5] We distinguish four cases: (i) in the simplest scenario the mapping is independent of the external destination endpoint and will not change for the same source pair $X{:}\,x$; (ii) some NAT's generate the same mapping only with same external destination IP address ($Y_1 \equiv Y_2$) or (iii) with the same external destination IP address *and* port number ($Y_1 \equiv Y_2$ and $y_1 \equiv y_2$)) and (iv) the mapping might be connection-dependent and vary each time a new connection to the same external destination pair is initiated.
- **Firewall.** We assume that any user is protected by a firewall from the outer network. Consequently, all flows need to be initiated from the user side (a technique called hole punching) before the server can replay the trace. For TCP flows, we reproduce the whole 3-way handshake without the interaction of the kernel on both endpoints. We assign an initial sequence number for each side of a TCP flow and modify it accordingly in the data packets. For UDP flows, we only send one probe from the client.
- **Initial sequence numbers.** It has been observed that some middleboxes overwrite the initial sequence number of TCP flows.[6] Since firewalls can easily keep track of the sequence numbers and reject inconsistent packets, it is important to intercept the overwritten sequence number from the SYN packet of the TCP handshake and modify all subsequent sequence and acknowledgement numbers individually for each TCP flow.
- **Timeouts.** In NetFilter,[7] the standard firewall provided in Linux, the TCP timeout for established connections is 5 days, while for UDP it is only 30 seconds whether packets have been seen in one or both directions. This means that we need to make sure that the time elapsing between two consecutive packets of each UDP flow, initial probes included, is less than 30 seconds. NAT mappings expire too in order to remove stale entries from the NAT table. Given the large amount of commercial NAT devices, we refer in this case

[4] Arash Molavi Kakhki, Abbas Razaghpanah, Anke Li, Hyungjoon Koo, Rajesh Golani, David Choffnes, Phillipa Gill, and Alan Mislove. Identifying traffic differentiation in mobile networks. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 239–251. ACM, 2015

[5] RFC 4787 - Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. URL: `http://tools.ietf.org/html/rfc4787`

RFC 5382 - NAT Behavioral Requirements for TCP. URL: `http://tools.ietf.org/html/rfc5382`

[6] Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Is it still possible to extend TCP? In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 181–194, New York, NY, USA, 2011. ACM

Gregory Detal, Benjamin Hesmans, Olivier Bonaventure, Yves Vanaubel, and Benoit Donnet. Revealing middlebox interference with tracebox. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 1–8, New York, NY, USA, 2013. ACM

[7] NetFilter: Firewalling, NAT and packet mangling for Linux. URL: `www.netfilter.org`

to the requirements and guidelines found in the RFC's. For UDP flows the timeout should not be less than 2 minutes, while 5 minutes is recommended;[8] for TCP if the connection is not yet in the established state, the timeout should not be less than 4 minutes, and if it is already in the established state it should be no less than 2 hours and 4 minutes.[9] These values are all large enough for the purpose of our experiment and do not interfere with ChkDiff. Finally, to avoid unnecessary synchronization between client and server, we do not actually send any acknowledgments from the client side for the TCP flows we replay. In the Linux kernel, the socket parameter TCP_USER_TIMEOUT sets the maximum amount of time that data can be transmitted without being acknowledged. Its default value is 20 minutes,[10] which is roughly one order of magnitude larger than a single run of ChkDiff.

[8] RFC 4787 - Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. URL: http://tools.ietf.org/html/rfc4787

[9] RFC 5382 - NAT Behavioral Requirements for TCP. URL: http://tools.ietf.org/html/rfc5382

[10] Linux Programmer's Manual - TCP protocol . URL: http://man7.org/linux/man-pages/man7/tcp.7.html

Given that our goal is not to classify all possible devices along a path but to rapidly deliver the trace to the client, we take a conservative approach and assume that the most stringent restrictions among the above ones are in place. We expect the NAT to be connection-dependent and perform a per-flow mapping discovery already during the hole punching initiated by the client against her firewall. For each flow, we encrypt the client's view of its source IP address and port number and add it to the payload of its SYN packet or UDP probe, as NAT devices are expected to overwrite every occurrence of the client's own IP address in a packet. We set the client-side initial sequence number of TCP flows in accordance to the acknowledgement numbers used in the trace (since no packets are sent from the client during the replaying phase, the acknowledgement number of a TCP flow sent by the server is constant across packets of the same flow). From the server side, for TCP flows, we keep track of incoming SYN packets along with the observed sequence numbers and client's source pair, and mimic the TCP handshake; for UDP flows we just keep track of client's and server's views of the client-side source pair. When these two views differ, we modify the client-side IP address and port number of the corresponding flows in $trace_{in}$ with the pair as seen by the server. The server also overwrites the acknowledgement number of a TCP flow, when the number in the trace does not match the sequence number seen in the received SYN packet. Additionally, in order to overcome the relatively short timeout on UDP flows, we enforce a maximum interval of 30 seconds between any two UDP packets of the same flow when shuffling $trace_{in}$ and start a timeout on the client side to make sure that we do not exceed 30 seconds between the initial UDP probe of a flow and its first occurrence in the trace. We discard the current run of the experiment and start a new one if ever this timeout expires. In any case, during an ordinary execution of ChkDiff only a few seconds elapse between hole punching and the start of the replaying phase.

We avoid the overhead of opening a socket for each flow and replaying the trace from the application layer by injecting packets with

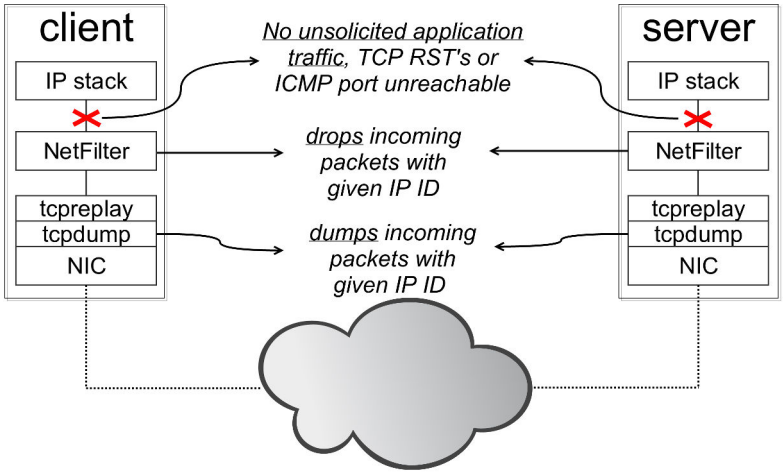


Figure 5.2: Configuration of client and server.

tcpreplay[11] directly between the IP layer and the Network Interface Card. Since we are emulating TCP and UDP flows below the IP layer, we need to prevent our packets from reaching their respective client applications, which could cause unsolicited traffic or unexpected application behaviour. Also, our hole-punching probes target ports on the server on which no process should be listening and will trigger TCP reset packets for TCP SYN probes, ICMP port-unreachable messages for UDP probes and real application packets if ever a process is indeed listening. As error messages cross a firewall on their way to the user, the corresponding newly-created connection entries are removed. Therefore, we need a way to distinguish between experiment packets and regular traffic, so that we can drop the former right before they reach the IP stack and we can allow the latter to pass through. We achieve this by assigning a unique number to each user session and overwrite with this value the 2-Byte IP ID field of each experiment packet between a given user and the server. Through a combined use of *tcpdump* and *iptables*, as shown in Figure 5.2, we dump the experiment packets right at the Network Interface Card and drop them before they reach the kernel, so that error messages are never generated at either endpoint.

[11] Tcpreplay. URL: `http://tcpreplay.appneta.com/`

Replay. We are now ready to replay the trace from the server at a constant packet rate higher than the original one (by default, we replay it at twice the original rate).

We dump the replayed trace on both the server and the client and then for each flow we measure One-Way Delays and note the number of lost packets. For simplicity, we avoid any clock synchronization between user and server: any effect due to clock skewing is not expected to disrupt the measured delays for the short time window of one experiment (a few tens of seconds) and in any case will affect all flows equally, as they are evenly spread across the trace.

Once the server has completed the replaying phase, the client closes the emulated TCP connections by sending an RST packet for each TCP flow in $trace_{in}$. This has the added benefit of clearing space

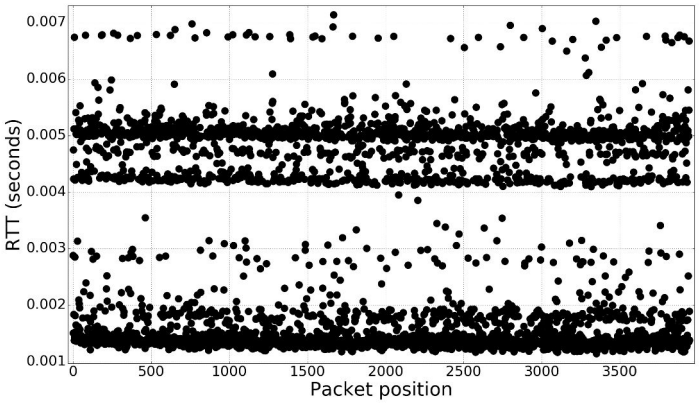


Figure 5.3: Timeseries of an experiment with packets following multiple paths. Each packet is represented by a black dot.

in the open connection table of the firewall, if ever a per-user restriction is active.

Results analysis. The study of delays between two endpoints across a path of several hops has to necessarily take into account the possibility of multiple paths. Discovering the paths taken by each flow would be cumbersome: first of all, we do not know the exact hash function applied in a load balancing decision and we observed that some data centers, where the measurement server could be deployed, make a massive use of load balancers; second of all, it would take some extra time, as we would need to probe at low rates (i.e. 1 packet per second) to bypass ICMP rate limitation - as seen in Chapter 3 - and have a complete view of each path. An example is provided in the timeseries of Figure 5.3, where we can visually identify at least five different paths; a direct comparison between any two flows becomes harder in this case. In the absence of the ground truth, we can rely on the fact that non-differentiated flows following the same path will have similar delay distributions, which a clustering algorithm can group together. A differentiated flow going on any of the available paths will show a distribution significantly different from that of all other flows and should not belong to any of the discovered groups. We combine this with a loss analysis in order to capture the behaviour of shapers.

- **Delays.** Our choice of clustering algorithm for delays is dbscan,[12] which groups together points that are in the same high-density area and labels as outliers those that do not belong to any found cluster. As a representative point for each flow we take its 25th percentile: it is close enough to the real path delay, it discards possible queueing delays and it is robust to delay variations due for example to WiFi. The algorithm then takes two parameters: the minimum number n of core samples to form a cluster and the maximum distance ϵ between any two samples for them to be included as core points in a cluster. Since we expect shaped flows to stand out from non-shaped flows, we set n to 2. As for ϵ, we need a value that reflects the delay variations of a path: we take the core values (2nd quartile range, i.e. the 25th-50th percentile

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996

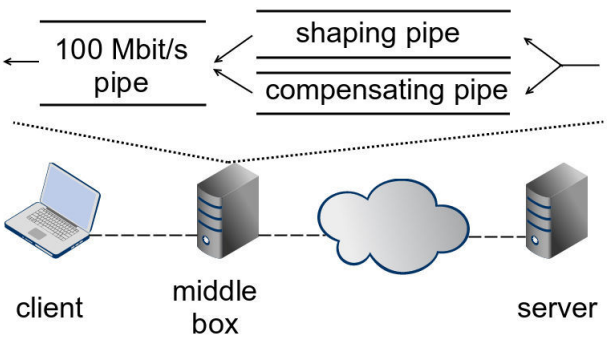


Figure 5.4: Setup used in the validation of ChkDiff, along with the shaper configuration.

range) of the delay distribution of each flow, we aggregate them and then pick for ϵ a large value in this set, the 75th percentile.

The output of dbscan will be a set of clusters of flows and a set of outliers. We label the latter as having failed the delay analysis.

- **Losses.** We compare the losses of a flow to the loss rate of the rest of the trace as a whole, in the way illustrated in Chapter 4. The reasoning is the following: if a flow i with s_i packets has not been differentiated, its number of lost packets can be modeled as a binomial random variable of parameters $B(s_i, p)$, where p is the loss rate of the rest of the trace. If we approximate this binomial to a normal random variable of parameters $N(s_i p, s_i p(1-p))$, we can verify whether the number of lost packets l_i of flow i lies within α standard deviations of the normal mean, where α is approximated to 2.58 for our chosen significance level of 99%. Since we are interested to know whether a flow experienced *more* losses than it should have with the global loss rate p, we check that $l_i < ps_i + \alpha\sqrt{p(1-p)s_i}$. If the condition does not hold, the flow is rejected by our loss analysis.

Since a shaper affects the delays or the losses of a flow, or both, we reject a flow if it fails either analysis.

We repeat the whole experiment three times in order to remove transient errors and claim that a flow was differentiated if in *all* three runs it failed the combined analysis of delays and losses.

5.2 *Validation*

We validate the downstream experiment of ChkDiff in wired, WiFi and 3G setups (Figure 5.4) with a client located in France and the server located in three different Amazon data centers: Germany, Ireland and Oregon (USA). The client is directly connected to a middlebox, where the shaper is deployed and which serves also as the client's gateway. In the WiFi setup, the client is connected to the gateway through a dedicated WiFi network operating on the same channel as the local University WiFi network to cause more link-level collisions. In the 3G setup, the client is connected to the middlebox via a wired connection and the middlebox is connected via WiFi to a

mobile phone functioning as hotspot. We test ChkDiff in the two differentiation scenarios seen in Chapter 4: given a set of flows we want to differentiate, in Scenario 1 we throttle their bandwidth and in Scenario 2 we apply a uniform packet drop rate to them. We configure dummynet[13] on the middlebox to shape incoming traffic, as shown in the upper part of Figure 5.4. Flows to shape are forwarded to the upper pipe, which applies the desired differentiation technique according to the scenario we test; flows that we do not intend to differentiate go through the lower pipe, which only adds a constant delay equal to the transmission delay of the upper pipe in Scenario 1 and has no effect in Scenario 2. This way, in Scenario 1 the difference in delays between shaped and non-shaped flows is due only to the queueing delay at the upper pipe. A final pipe, where all flows eventually go, emulates a 100 Mbit/s link. In Scenario 2 this last pipe causes uniform drops on the whole trace.

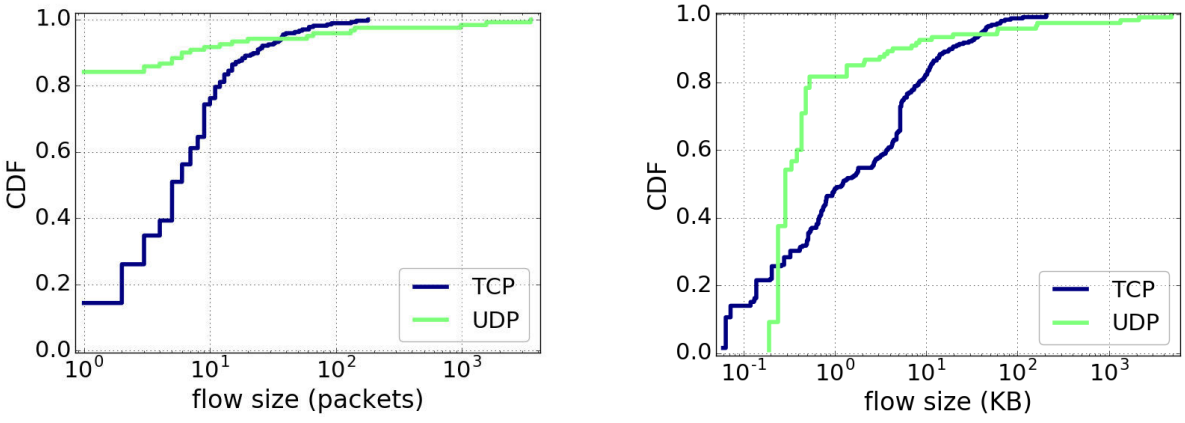
[13] Marta Carbone and Luigi Rizzo. Dummynet revisited. *SIGCOMM Comput. Commun. Rev.*, 40(2):12–20, April 2010

In all experiments we replay a trace of approximately 9000 packets captured during an Internet session of 3 minutes that included watching a short streaming video, browsing a news website and making a call on Skype. In dummynet, our pipes have a buffer length of 100 packets and use droptail buffer management policy.

5.2.1 *Shaping pipe (Scenario 1)*

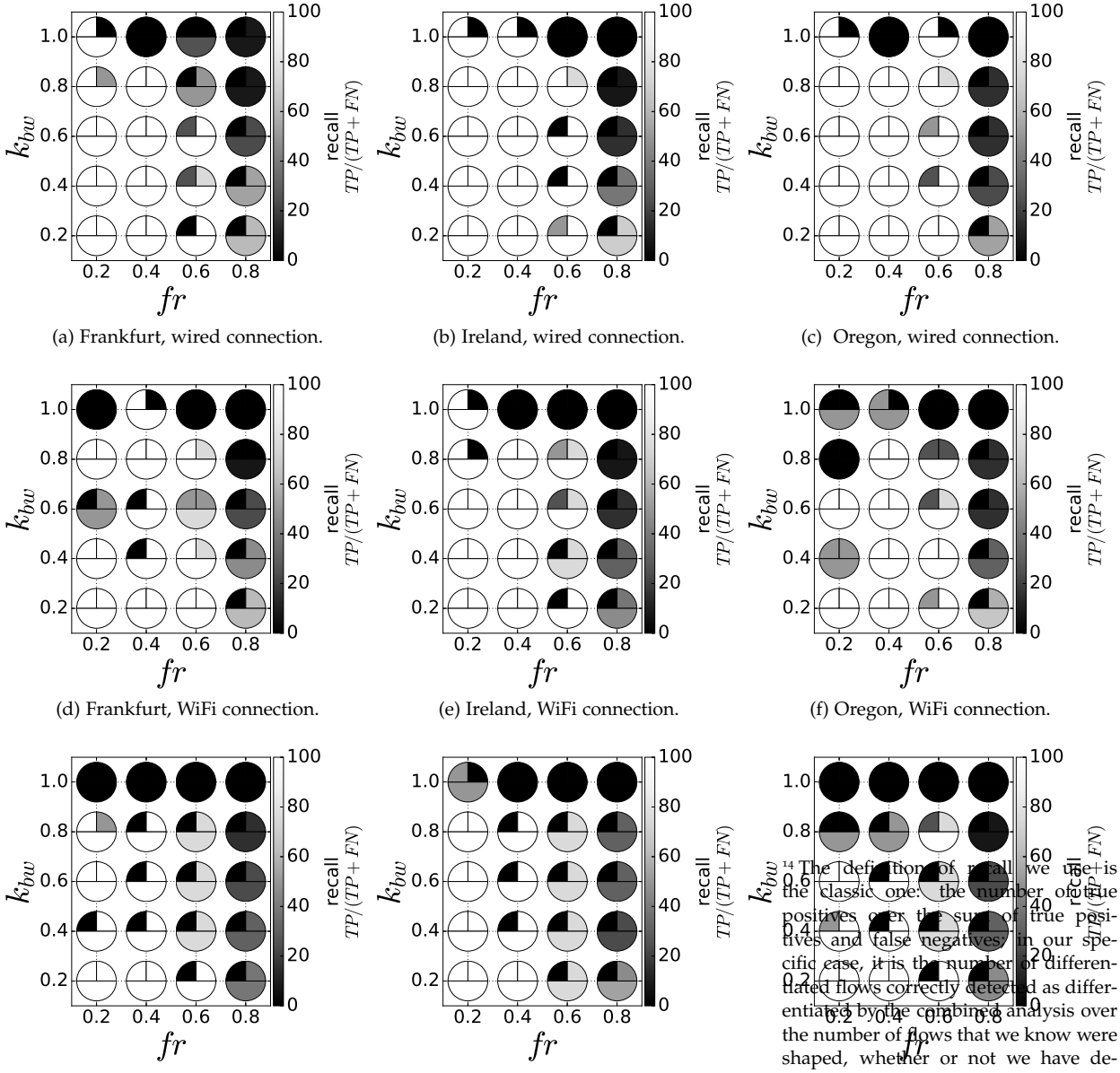
In this scenario, we compute in $trace_{in}$ the overall sending rate of the flows to shape and set the shaping pipe to a fraction $k_{bw} < 1$ of this value. The second parameter we vary is the fraction fr of packets we shape. When picking which flows to differentiate, we choose iteratively the flow closest to the target fr, until the desired size is reached. All flows we differentiate go through the same shaping pipe. By combining delay and loss analysis, we show that we can effectively identify all shaped flows as long as the fraction fr does not constitute most of $trace_{in}$, which is what we expect when we take the whole trace as baseline for comparison.

Figure 5.5: Distribution of flow sizes for the trace we used in the validation of the tool.



(a) CDF of flow sizes in # of packets.

(b) CDF of flow sizes in KB.



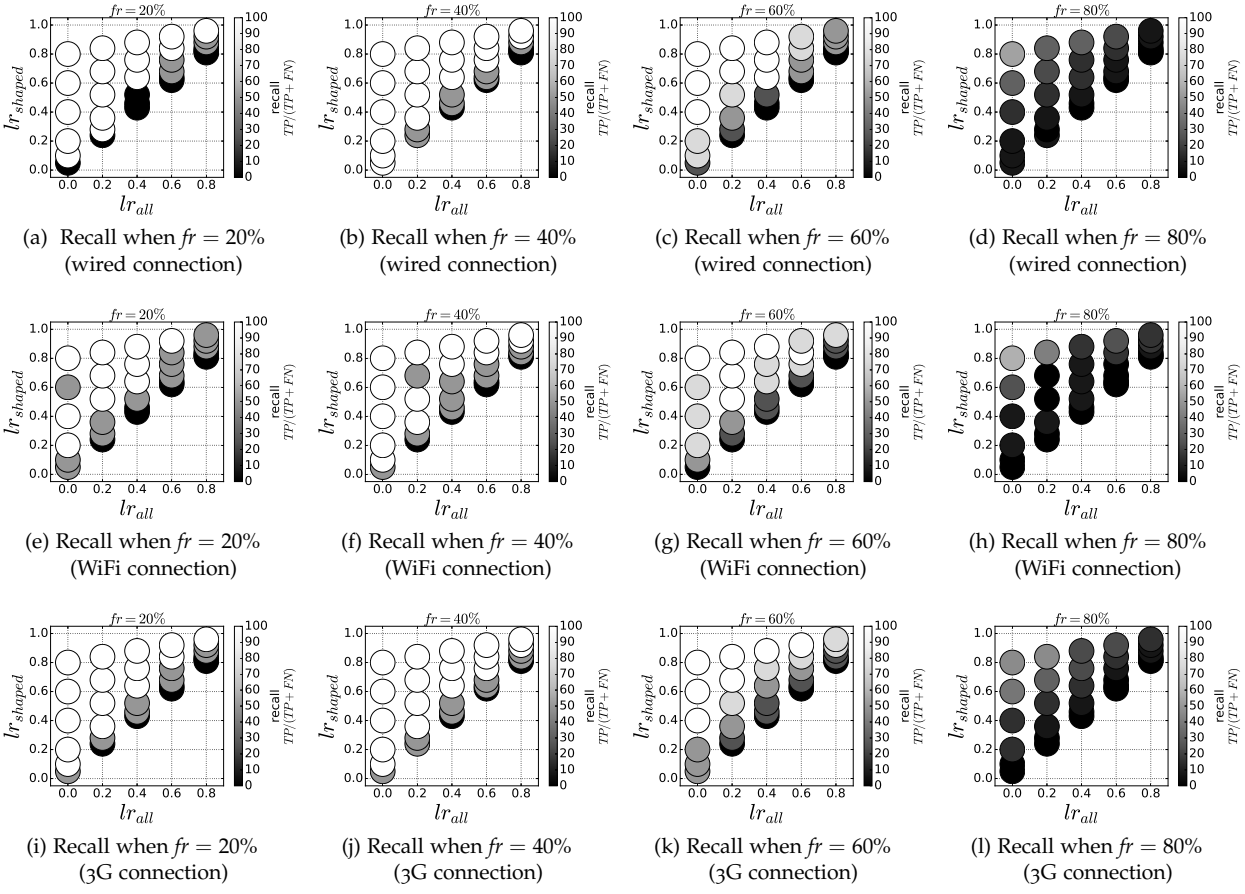
(a) Frankfurt, wired connection.
(b) Ireland, wired connection.
(c) Oregon, wired connection.
(d) Frankfurt, WiFi connection.
(e) Ireland, WiFi connection.
(f) Oregon, WiFi connection.
(g) Frankfurt, 3G connection.
(h) Ireland, 3G connection.
(i) Oregon, 3G connection.

Figure 5.6: Recall of the combined analysis (delay and losses) for Scenario 1 over wired, WiFi and 3G connections.

We present in Figure 5.6 the results in terms of recall[14] for each of the three server locations over wired, WiFi and 3G connections. For each pair of fr and k_{bw} we show a pie where the colour of the upper-left quarter represents the result of the delay analysis, the colour of the upper-right quarter represents the result of the loss analysis, and the colour of the lower half is the outcome of the combined analysis. If for $fr \leq 40\%$ in almost all cases the delay analysis suffices to detect all shaped flows, for $fr = 60\%$ we see that combining the delay and loss analysis is essential for a correct output. When $k_{bw} = 1.0$ the shaping pipe is configured with the average bit rate of incoming packets, so the flows that are supposedly being differentiated might not be shaped at all, hence the uncertain outcome on the top row of each graph.

In this scenario, ChkDiff appears to cope just as well with a wired

[14] The definition of recall we use is the classic one: the number of true positives over the sum of true positives and false negatives; in our specific case, it is the number of differentiated flows correctly detected as differentiated by the combined analysis over the number of flows that we know were shaped, whether or not we have detected them. A recall of 100% indicates that we have correctly identified all shaped flows, while a recall of 0% indicates that we have missed them all. We do not include the results in terms of precision, commonly defined as the number of true positives over the number of positives, since we found it for all experiments to be the optimal one, at 100%; in short, we never flagged a non-differentiated flow erroneously.



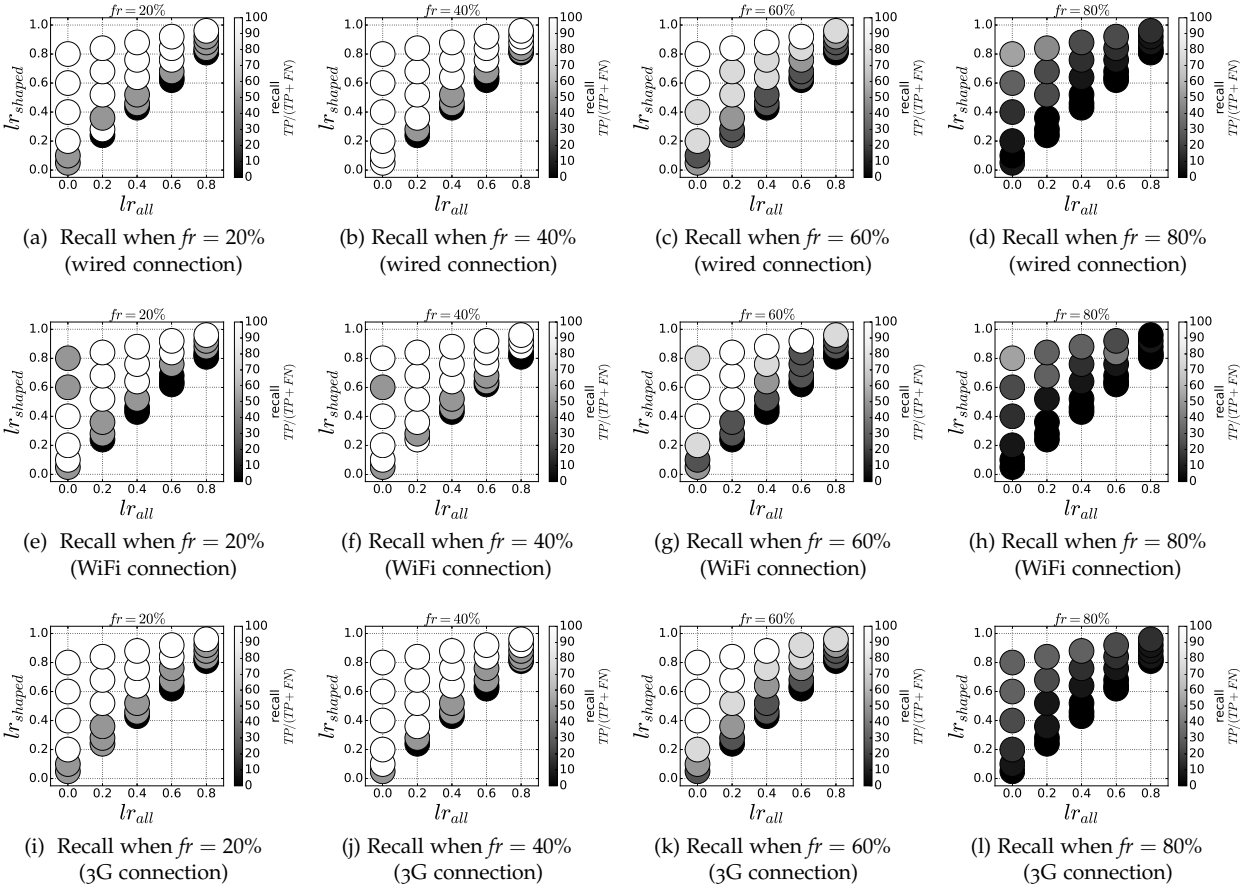
(a) Recall when $fr = 20\%$ (wired connection)
(b) Recall when $fr = 40\%$ (wired connection)
(c) Recall when $fr = 60\%$ (wired connection)
(d) Recall when $fr = 80\%$ (wired connection)
(e) Recall when $fr = 20\%$ (WiFi connection)
(f) Recall when $fr = 40\%$ (WiFi connection)
(g) Recall when $fr = 60\%$ (WiFi connection)
(h) Recall when $fr = 80\%$ (WiFi connection)
(i) Recall when $fr = 20\%$ (3G connection)
(j) Recall when $fr = 40\%$ (3G connection)
(k) Recall when $fr = 60\%$ (3G connection)
(l) Recall when $fr = 80\%$ (3G connection)

Figure 5.7: Recall of the loss analysis as we vary *fr* in Scenario 2 from the server located in Germany, over wired, WiFi and 3G connections.

connection as it does with WiFi and 3G. Over the two wireless connections, there seems to be more noise that is in any case neutralized for the most part when combining the two analysis.

5.2.2 *Uniform drops (Scenario 2)*

We consider now a shaper that uniformly drops packets of selected flows at a loss rate *lr* (upper-right pipe in Figure 5.4) and of the whole trace at a loss rate lr_{all} (left pipe in the same figure). As opposed to the previous scenario, we deploy here a dedicated shaping pipe for each flow we want to differentiate. We vary *lr* and lr_{all}, as well as the fraction *fr* of traffic impacted by *lr*. Shaped flows will thus have an overall loss rate lr_{shaped} equal to $1 - (1 - lr)(1 - lr_{all})$. We show the results for each of the three server locations also used in Scenario 1, over the three types of connection previously considered (Figures 5.7, 5.8 and 5.9). Since in this scenario the differentiation we apply does not affect delays, we focus only on the outcome of the loss analysis. For completeness, results are shown also for high values of lr_{all}, even if in practice a global loss rate of 20% is already able to disrupt TCP connections. Results appear qualitatively similar regardless of where the server is located. We observe for both wired and wireless setups that when *fr* is less than half of the trace, ChkDiff is able to detect all differentiated flows except for the cases on



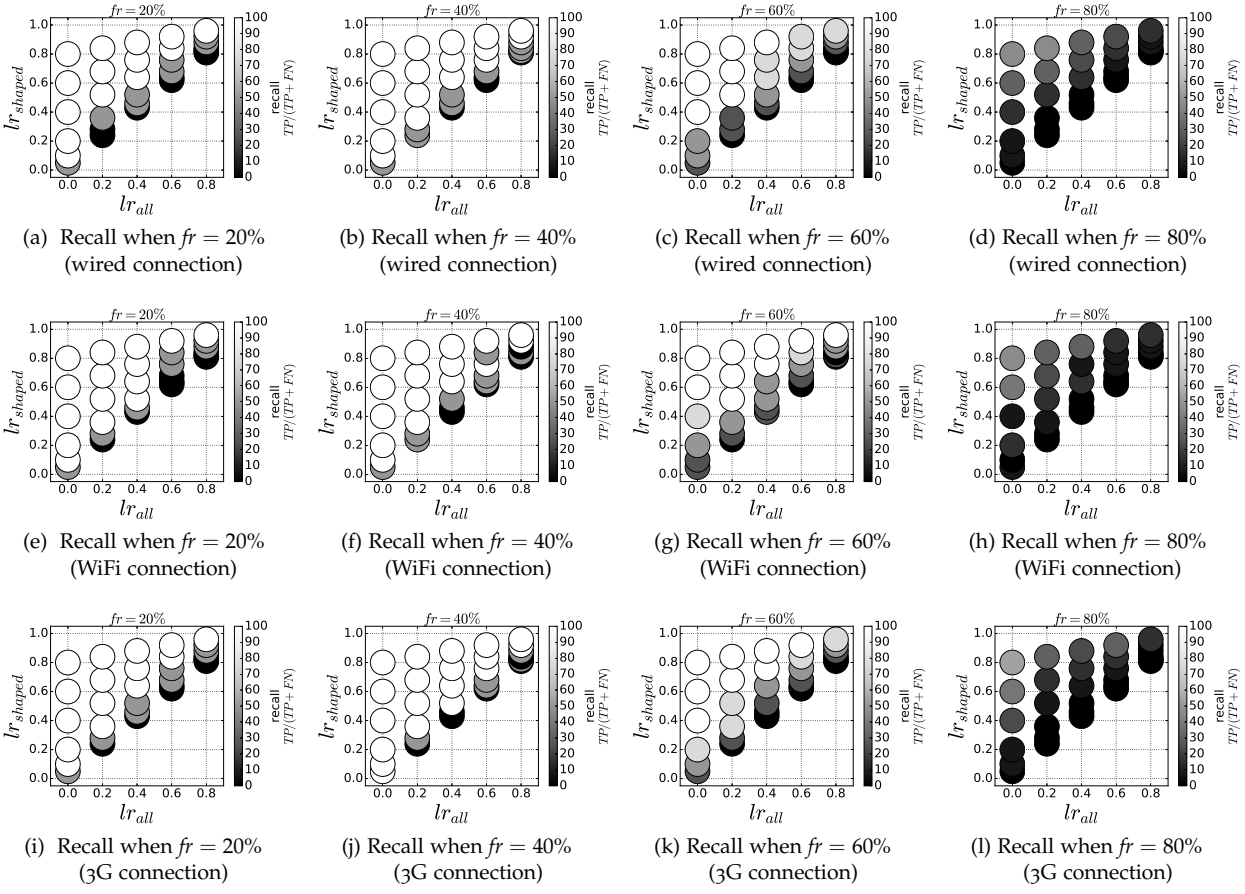
(a) Recall when $fr = 20\%$ (wired connection)
(b) Recall when $fr = 40\%$ (wired connection)
(c) Recall when $fr = 60\%$ (wired connection)
(d) Recall when $fr = 80\%$ (wired connection)
(e) Recall when $fr = 20\%$ (WiFi connection)
(f) Recall when $fr = 40\%$ (WiFi connection)
(g) Recall when $fr = 60\%$ (WiFi connection)
(h) Recall when $fr = 80\%$ (WiFi connection)
(i) Recall when $fr = 20\%$ (3G connection)
(j) Recall when $fr = 40\%$ (3G connection)
(k) Recall when $fr = 60\%$ (3G connection)
(l) Recall when $fr = 80\%$ (3G connection)

Figure 5.8: Recall of the loss analysis as we vary *fr* in Scenario 2 from the server located in Ireland, over wired, WiFi and 3G connections.

the bottom diagonal, where the difference in loss rate between differentiated and non-differentiated flows is the lowest (5%, 10% and sometimes 20%). Experiments over WiFi and 3G seem to be only slightly worse than over wired for the lowest values of lr_{all}.

5.3 *Discussion*

A full run of ChkDiff in upstream and downstream directions is able to detect differentiation when, regardless of its implementation, it directly worsens the throughput, packet delay and losses of user applications. This is the typical effect introduced by a shaper. Even though throughout the thesis we used the terms shaping and differentiation interchangeably, the former is a subset of the latter and shaping is what we aim at detecting with our current tool. As we saw in the validation section, we cannot precisely reveal shaping when most of the user traffic is affected, as our baseline in the analysis would mainly be made of differentiated flows. To counteract this, a user should be running different applications during the capturing phase, so as to have a variety of flows to check against. In the extreme case, if really an ISP throttles the bandwidth of all traffic for a given user, it would not be possible to discern it from severe network congestion from the view point of this particular user.



(a) Recall when $fr = 20\%$ (wired connection)
(b) Recall when $fr = 40\%$ (wired connection)
(c) Recall when $fr = 60\%$ (wired connection)
(d) Recall when $fr = 80\%$ (wired connection)
(e) Recall when $fr = 20\%$ (WiFi connection)
(f) Recall when $fr = 40\%$ (WiFi connection)
(g) Recall when $fr = 60\%$ (WiFi connection)
(h) Recall when $fr = 80\%$ (WiFi connection)
(i) Recall when $fr = 20\%$ (3G connection)
(j) Recall when $fr = 40\%$ (3G connection)
(k) Recall when $fr = 60\%$ (3G connection)
(l) Recall when $fr = 80\%$ (3G connection)

Figure 5.9: Recall of the loss analysis as we vary *fr* in Scenario 2 from the server located in Oregon, over wired, WiFi and 3G connections.

In this work, we assume that ISPs select flows to differentiate based on packet fields (IP addresses, ports and payload). We do not directly address classification based on flow bandwidth, but we make sure that we replay the trace at a higher global rate than the original one. An alternative for shuffling that we have not yet fully explored would be to rearrange the packets of a flow inside the trace in a way that would also take into account the original flow rate, so that it would be straightforward to scale up the flow rates when replaying the trace. If instead an ISP shapes certain traffic at specific times of the day (e.g. peak hours), ChkDiff will simply report it as differentiated if executed during those hours.

There exist other techniques for differentiation other than shaping, as we saw in the reported cases of traffic differentiation in Chapter 2. When an ISP performs port blocking[15] or, more in general, blocking of entire applications, the traffic generated by a user connected through that ISP would only consist of a few outgoing requests, certainly not enough for ChkDiff to work. In case users suspect their ISP to apply such policies, they could for example capture their own traffic in a different network (e.g., using a cell phone as a hotspot) and then replay it against the suspected ISP. Alternatively, we could envisage to provide along with our tool, or let advanced users upload, a set of flows from well-known applications that a user could

[15] Robert Beverly, Steven Bauer, and Arthur Berger. The Internet is not a big truck: toward quantifying network neutrality. In *Passive and Active Network Measurement*, pages 135–144. Springer, 2007

inject into their own traces. When a flow experiences worse performance due to interconnection provisioning between an ISP and a content provider, we can detect differentiation with the upstream experiment if the problem affects upstream traffic and is localized within 3-4 hops from the user. The downstream experiment, as in all other measurement tools making use of a server, would probably bypass any suspected links directly connected to the content provider in question. Another possible differentiation technique is RST packet injection, through which an ISP could terminate selected TCP connections, as for example did Comcast in 2007 for BitTorrent users sharing their files.[16] We do not currently implement detection of injected RST packets, but it would be an easy addition: we would just need to dump RST packets on the user side and verify if they belong to any flow in the replayed trace. We will soon complement ChkDiff with this extra feature in future work.

[16] Packet forgery by ISPs: A report on the comcast affair. URL: `https://www.eff.org/wp/packet-forgery-isps-report-comcast-affair`

Also, we do not detect differentiation when it aims at providing better treatment to selected traffic. Such behaviour could be the result of an agreement between a content provider and an ISP and it does not necessarily imply any worse conditions for the rest of the traffic than in normal network conditions. It would be interesting to redefine the delay analysis in both upstream and downstream experiments to account for this, as it could reveal what services and applications are favoured in a given network. We plan to address this in future work.

The tool is available for Linux machines on the web page of the project.[17] We currently provide a server located in our lab, where no traffic differentiation is taking place.

[17] `http://chkdiff.gforge.inria.fr/`

5.4 *Assessment with respect to existing methods*

Many tools for the detection of traffic differentiation have appeared in the literature in recent years, as detailed in Chapter 2. Among the first ones, BT-test[18] checks for injected RST packets while emulating a BitTorrent packet exchange between a user and a server. Other tools compare the performance of a synthetic application flow to the performance of a similar flow with some modified or randomized packet fields (port numbers or payloads), so that a shaper targeting such application would affect the former and not the latter. Glasnost[19] looks for differences in throughput between these two flows, while DiffProbe[20] attempts to create congestion in the ISP network by scaling up the replaying rate of application and control flows, and then analyzes their delay and loss distributions. ShaperProbe[21] expands on DiffProbe by considering the case of a shaper implemented as a token bucket and tries to infer its parameters as the received rate at the destination shows a level shift. Packsen[22] also tries to identify the shaper type and its parameters, but claims to use a more efficient statistical analysis. As opposed to ChkDiff, these tools are limited to the set of application traces made available by their authors (e.g., Skype, BitTorrent, YouTube, etc), which would make it hard to main-

[18] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. Detecting BitTorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC'08)*, Vouliagmeni, Greece, October 2008

[19] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna Gummadi, Ratul Mahajan, and Stefan Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, Apr 2010

[20] Partha Kanuparthy and Constantine Dovrolis. Diffprobe: detecting ISP service discrimination. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 1649–1657, Piscataway, NJ, USA, 2010. IEEE Press

[21] Partha Kanuparthy and Constantine Dovrolis. Shaperprobe: End-to-end detection of isp traffic shaping using active methods. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 473–482. ACM, 2011

[22] Udi Weinsberg, Augustin Soule, and Laurent Massoulié. Inferring traffic shaping and policy parameters using end host measurements. In *INFOCOM*, pages 151–155, 2011

tain them in the long run.

A recent work, Differentiator Detector,[23] aims at solving this by replaying between user and server a captured user trace and reproducing the same original application behaviour (ports, payloads, inter-packet times) *at* the application layer, first through a direct path between the two endpoints (application flow) and then through a VPN tunnel to a middlebox (control flow). It measures throughput, RTT distribution and losses in order to detect shaping. Even though our tool replays separately upstream and downstream traffic, in the upstream direction it tests the real hops where the original packets went instead of testing the path between client and server, and in the downstream direction it deals in a more robust and scalable way with NAT devices.

[23] Arash Molavi Kakhki, Abbas Razaghpanah, Anke Li, Hyungjoon Koo, Rajesh Golani, David Choffnes, Phillipa Gill, and Alan Mislove. Identifying traffic differentiation in mobile networks. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 239–251. ACM, 2015

Nano[24] differs from existing solutions in that it carries out passive measurements on user traffic and compares it against a data set of other users in the same geographical area, with comparable machine setups, at the same time of the day, but connected to a different ISP. While this method is undeniably independent of user applications and differentiation techniques, its main disadvantage is that it needs a fairly large number of users for it to be operational. ChkDiff on the other hand depends solely on the end user running it.

[24] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. Detecting network neutrality violations with causal inference. *ACM SIGCOMM CoNext*, page 289, 2009

5.5 Summary

We extended with a downstream experiment ChkDiff, a tool which enables users to detect differentiation on their own traffic. After first checking for degraded traffic performance on upstream traffic, the tool replays user incoming flows from a measurement server to the user and analyzes delays and losses to verify whether each flow experienced the same network conditions as the rest of the trace. While in the upstream direction our tool proved to be robust to rate limitation in the ICMP feedback generated by routers, in the downstream case we successfully cope with NAT's and middleboxes in front of the client and with end-to-end measurements possibly comprising a diversity of paths between server and client. We validated ChkDiff in the wild, with two differentiation scenarios over three types of connections: wired, WiFi and 3G. We showed that it correctly identifies shaped flows when up to half of a user trace is affected.

In future work, we envisage to include in the tool tests that check for differentiation techniques that do not necessarily alter delays and losses, as for instance TCP RST injection. We also intend to run a study with volunteers in a variety of wired and mobile setups in order to have a mapping of the current practices of ISPs.

6
Conclusion

In this thesis, we presented and validated ChkDiff, a novel tool for the detection of traffic differentiation at the access ISP, and conducted a study on the responsiveness routers to TTL-limited probes. We conclude with a summary of our contributions in Section 6.1 and a discussion about future directions in Section 6.2.

6.1 Summary

Compared to existing work, the strength of ChkDiff lies in its ability to test any application run by the user and to apply to any shaping technique deployed by an ISP that results in degraded flow performance. Our tool runs measurements directly on previously-captured user traffic and comprises two experiments, according to the direction of the traffic. In both cases we shuffle the user trace in such a way that, when replayed, all flows will experience the same network conditions.

In the upstream experiment, we replay outgoing user traffic against the routers at the first few hops away from the user by forging the TTL field of each packet. With the ICMP time-exceeded replies returned by intermediate routers we compute per-flow delays and losses and infer differentiation by comparing the performance of each flow against that of the rest of the trace. By repeating this experiment across successive hops, we are also able to localize the position of possible shapers in terms of number of hops from the user. We validated this in wired and WiFi setups in two different shaping scenarios, where we respectively throttled the bandwidth of selected flows and applied a uniform loss rate to selected flows and to the whole trace. We showed that ChkDiff is able to correctly detect shaped flows when up to 60% of traffic undergoes bandwidth throttling and when up to 40% of traffic is subject to a higher loss rate than the whole trace. We also validated our tool in the presence of ICMP rate limitation and showed that it does not alter significantly our results, even when only 20% of probes receive a reply.

In order to corroborate our choice of measurements in the upstream experiment, we first studied the responsiveness of routers to TTL-limited probes with a large-scale campaign from 180 hosts in PlanetLab. Among the 850 routers we tested, we found that almost

a third of them were fully responsive in the range of probing rates we considered (up to 2500 *pps*), around 4% were unresponsive and around 60% implemented ICMP rate limitation. Among these, the most common form of ICMP limitation displays an on-off pattern, defined by the burst size of the generated packets and the inter-burst time, which we characterized also according to the router vendors. Even at relatively high probing rates, we did not hit any capacity limits that resulted in larger delays. This means that in the upstream experiment of ChkDiff we can probe at rates at least within the range we tested without impairing the measurements of round-trip times.

In the downstream experiment, we replay incoming user traffic from a measurement server and detect differentiation by measuring per-flow one-way delays and losses. We take the necessary actions to handle NATs, firewalls and middleboxes and, unlike other methods, we consider the possibility of having multiple paths between server and client when analyzing flow delays. We validated this in wired, WiFi and 3G setups using a measurement server deployed in three different data center locations, under the two shaping scenarios already used for the upstream experiment. Similarly to the upstream case, results showed that ChkDiff successfully detects differentiation when up to 60% of traffic has its bandwidth throttled and up to 40% of traffic experiences a higher loss rate than the overall one. The results were qualitatively similar across the three types of connection, thus proving the robustness of our analysis.

6.2 *Future directions*

A natural next step for our project would be to distribute the tool to a wider audience. First, we could ask undergraduate students at our University to download and run it, so as to have a global view of traffic differentiation practices by French access ISPs. Then, we could envisage a collaboration with M-Lab,[1] which already hosts servers for or offers visibility to tools for differentiation detection (Glasnost, Neubot) and network monitoring and troubleshooting (e.g., NDT, Pathload2, BISmark, MobiPerf, etc.). M-Lab could ideally host our measurement server for the downstream experiment in a number of locations, in order to be able to test different paths to the user. Along with this, we could also collect anonymized results from users and publish per-country statistics, similarly to what is done for Glasnost.[2]

[1] www.measurementlab.net

[2] http://broadband.mpi-sws.org/transparency/results/

A mobile version of ChkDiff could also be implemented, perhaps in a lightweight version that reduces the measurement overhead. Porting to Android and iOS presents several challenges, especially in the capture and replay of traffic, since in a mobile environment we should not expect to have administrator privileges on the device. Alternatives to the usage of tcpdump and tcpreplay should be explored. Currently, as shown in the validation section of Chapter 5, a user can check if her mobile operator applies shaping on her 3G connection simply by turning her mobile phone into a WiFi hotspot and connecting a computer to it. With this approach, we can easily

test desktop applications, but not mobile ones.

As we saw in Chapter 2, both the United States and the European Union have adopted regulations that enforce network neutrality and restrict the number of cases in which an ISP is allowed to apply traffic differentiation. Nevertheless, no tool is currently widely adopted by National Regulatory Authorities (NRAs) and even then NRAs can only check for differentiation on a limited set of applications, for instance by running Glasnost. ChkDiff could be a solution for this, with users sending the output of ChkDiff to their national NRA in case of detection of traffic differentiation.

In our tool, we check for differentiation on the exact paths covered by the original trace only in the upstream experiment. In the downstream direction, since IP spoofing is not a viable solution in today's networks,[3] we needed to resort to replaying from a measurement server using the server IP address as source IP address in all packets of the trace. This is the same technique used by all other tools that perform active measurements: they test for differentiation along the path between the server and the user, which in the vicinity of the user might not cover exactly the hops traversed by the original flows. This is not necessarily an issue, but it is certainly a limitation of this type of measurements. As a workaround, as hinted above, we could offer the option to replay the downstream trace from different server locations in order to cover more paths to the user.

[3] BCP 38 - Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. URL: https://tools.ietf.org/html/bcp38

Finally, our methodology infers neutrality violations from Quality of Service (QoS) parameters, without taking into account Quality of Experience (QoE). In other words, we only consider the results of network measurements (delays and losses) and not the effects on the usability of applications. This could be a direction to explore. ACQUA,[4] a parallel project in our research team, tries to predict with machine-learning techniques the QoE of popular applications (e.g. Skype, YouTube) based on the measured QoS. We could integrate ACQUA into our project in order to also infer if a detected differentiation mechanism significantly degrades the usability of an application. Moreover, since an equal treatment of traffic does not necessarily translate into an equally good QoE for applications with different requirements (e.g., file sharing, video streaming, VoIP, IPTV, etc), we could also envisage to explore which shaping mechanisms do not degrade but *improve* the QoE of different types of applications.

[4] htttp://project.inria.fr/acqua

Bibliography

[1] AT&T blocking iPhone's FaceTime app would harm consumers and break net neutrality rules. URL: http://www.freepress.net/press-release/99480/att-blocking-iphones-facetime-app-would-harm-consumers-and-break-net-neutrality.

[2] BCP 38 - Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. URL: https://tools.ietf.org/html/bcp38.

[3] Bouygues télécom filtre malhonnêtement son réseau 3G et inspecte vos données. URL: http://grapsus.net/blog/post/Bouygues-Telecom-filtre-malhonnetement-son-reseau-3G-et-inspecte-vos-donnees.

[4] Chile, primer país en incorporar la neutralidad en la red. URL: http://www.elmundo.es/elmundo/2010/07/16/navegante/1279272468.html.

[5] Dslreports: comcast is using sandvine to manage p2p connections. URL: http://www.dslreports.com/forum/r18323368-Comcast-is-using-Sandvine-to-manage-P2P-Connections.

[6] Group asks FCC to probe iPhone Skype restrictions. URL: http://fortune.com/2009/04/03/group-asks-fcc-to-probe-iphone-skype-restrictions/.

[7] I just doubled my PIA VPN throughput that I am getting on my router by switching from UDP:1194 to TCP:443. URL: http://www.reddit.com/r/VPN/comments/1xkbca/i_just_doubled_my_pia_vpn_throughput_that_i_am.

[8] Linux Programmer's Manual - TCP protocol . URL: http://man7.org/linux/man-pages/man7/tcp.7.html.

[9] MetroPCS 4G Data-Blocking Plans May Violate Net Neutrality. URL: http://www.wired.com/2011/01/metropcs-net-neutrality/.

[10] Net neutrality enshrined in dutch law. URL: http://www.theguardian.com/technology/2011/jun/23/netherlands-enshrines-net-neutrality-law.

[11] NetFilter: Firewalling, NAT and packet mangling for Linux. URL: www.netfilter.org.

[12] Packet forgery by ISPs: A report on the comcast affair. URL: https://www.eff.org/wp/packet-forgery-isps-report-comcast-affair.

[13] Phone Company Helps Make the Case for Net Neutrality. URL: http://www.savetheinternet.com/blog/10/04/05/phone-company-helps-make-case-net-neutrality.

[14] Respect my net. URL: http://respectmynet.eu/view/205.

[15] Respect my net. URL: http://respectmynet.eu/view/196.

[16] RFC 4787 - Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. URL: http://tools.ietf.org/html/rfc4787.

[17] RFC 5382 - NAT Behavioral Requirements for TCP. URL: http://tools.ietf.org/html/rfc5382.

[18] Service Name and Transport Protocol Port Number Registry. *IANA*. URL: www.iana.org/assignments/port-numbers.

[19] Tcpreplay. URL: http://tcpreplay.appneta.com/.

[20] Vint Cerf speaks out on net neutrality. URL: https://googleblog.blogspot.fr/2005/11/vint-cerf-speaks-out-on-net-neutrality.html.

[21] Vonage says broadband provider blocks its calls. URL: http://www.cnet.com/news/vonage-says-broadband-provider-blocks-its-calls/.

[22] Widespread Hijacking of Search Traffic in the United States. URL: https://www.eff.org/deeplinks/2011/07/widespread-search-hijacking-in-the-us.

[23] Netflix performance on Verizon and Comcast has been dropping for months, 2013. URL: http://arstechnica.com/information-technology/2014/02/netflix-performance-on-verizon-and-comcast-has-been-dropping-for-months.

[24] FCC. Protecting and promoting the open internet. April 2015. URL: https://www.federalregister.gov/articles/2015/04/13/2015-07841/protecting-and-promoting-theopen-internet.

[25] WhatsApp Voice Calling Already Banned by UAE's Etisalat: Report. 2015. URL: http://gadgets.ndtv.com/apps/news/whatsapp-voice-calling-already-banned-by-uaes-etisalat-report-672283.

[26] P. Almquist. RFC 1349: Type of Service in the Internet Protocol Suite. 1992. URL: https://tools.ietf.org/html/rfc1349.

[27] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 153–158. ACM, 2006.

[28] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring load-balanced paths in the internet. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 149–160. ACM, 2007.

[29] Simone Basso, Antonio Servetti, and Juan Carlos De Martin. The network neutrality bot architecture: a preliminary approach for self-monitoring of Internet access QoS. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 1131–1136. IEEE, 2011.

[30] BEREC. A framework for Quality of Service in the scope of Net Neutrality. *BEREC Report BoR (11) 53*, December 2011. URL: `http://berec.europa.eu/eng/document_register/subject_matter/berec/download/0/117-a-framework-for-quality-of-service-in-th_0.pdf`.

[31] BEREC. Guidelines for quality of service in the scope of net neutrality. *BEREC Report BoR (12) 131*, November 2012. URL: `http://berec.europa.eu/eng/document_register/subject_matter/berec/download/0/1101-berec-guidelines-for-quality-of-service-_0.pdf`.

[32] BEREC. Annex of monitoring quality of internet access services in the context of net neutrality. *BEREC Report BoR (14) 117*, September 2014. URL: `http://berec.europa.eu/eng/document_register/subject_matter/berec/download/1/4602-monitoring-quality-of-internet-access-se_1.pdf`.

[33] BEREC. Monitoring quality of internet access services in the context of net neutrality. *BEREC Report BoR (14) 117*, September 2014. URL: `http://berec.europa.eu/eng/document_register/subject_matter/berec/download/0/4602-monitoring-quality-of-internet-access-se_0.pdf`.

[34] Robert Beverly, Steven Bauer, and Arthur Berger. The Internet is not a big truck: toward quantifying network neutrality. In *Passive and Active Network Measurement*, pages 135–144. Springer, 2007.

[35] S. Blake, F. Baker, and D. Black. RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. 1998. URL: `https://tools.ietf.org/html/rfc2474`.

[36] Marta Carbone and Luigi Rizzo. Dummynet revisited. *SIGCOMM Comput. Commun. Rev.*, 40(2):12–20, April 2010.

[37] Cisco. TTL expiry attack identification and mitigation. URL: http://www.cisco.com/web/about/security/intelligence/ttl-expiry.html.

[38] K Claffy, Tracie E Monk, and Daniel McRobb. Internet tomography. *Nature*, 7(11), 1999.

[39] Jon Crowcroft. Talk on Net Neutrality, December 2006. URL: "http://www.cl.cam.ac.uk/~jac22/talks/neut.ppt.gz".

[40] Jon Crowcroft. Net neutrality: the technical side of the debate: a white paper. *SIGCOMM Comput. Commun. Rev.*, 37:49–56, January 2007.

[41] Jon Crowcroft. Talk on Newt or Notrality, July 2011. URL: "https://www.cl.cam.ac.uk/~jac22/talks/notrality.ppt".

[42] Gregory Detal, Benjamin Hesmans, Olivier Bonaventure, Yves Vanaubel, and Benoit Donnet. Revealing middlebox interference with tracebox. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 1–8, New York, NY, USA, 2013. ACM.

[43] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna Gummadi, Ratul Mahajan, and Stefan Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, Apr 2010.

[44] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. Detecting BitTorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC'08)*, Vouliagmeni, Greece, October 2008.

[45] Constantine Dovrolis, Krishna Gummadi, Aleksandar Kuzmanovic, and Sascha D. Meinrath. Measurement lab: overview and an invitation to the research community. *SIGCOMM Comput. Commun. Rev.*, 40:53–56, June 2010.

[46] Allen B Downey. Using pathchar to estimate internet link characteristics. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 241–250. ACM, 1999.

[47] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[48] Ramesh Govindan and Vern Paxson. Estimating router ICMP generation delays. In *Passive & Active Measurement (PAM)*, 2002.

[49] BITAG Technical Working Group. Differentiated Treatment of Internet Traffic. 2015.

[50] Mehmet H Gunes and Kamil Sarac. Analyzing router responsiveness to active measurement probes. *Passive and Active Network Measurement*, pages 23–32, 2009.

[51] Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Is it still possible to extend TCP? In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 181–194, New York, NY, USA, 2011. ACM.

[52] Ningning Hu, Li Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. A measurement study of internet bottlenecks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1689–1700. IEEE, 2005.

[53] Van Jacobson. traceroute. URL: `ftp://ftp.ee.lbl.gov/traceroute.tar.gz`.

[54] Partha Kanuparthy and Constantine Dovrolis. Diffprobe: detecting ISP service discrimination. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 1649–1657, Piscataway, NJ, USA, 2010. IEEE Press.

[55] Partha Kanuparthy and Constantine Dovrolis. Shaperprobe: End-to-end detection of isp traffic shaping using active methods. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 473–482. ACM, 2011.

[56] Ioannis Koukoutsidis. Public QoS and net neutrality measurements. *Journal of Information*, 5, 2015.

[57] Farah Layouni, Brice Augustin, Timur Friedman, and Renata Teixeira. Origine des étoiles dans traceroute. In *Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, 2008.

[58] Lawrence Lessig and Robert W. McChesney. No Tolls on The Internet. June 2006. URL: `http://www.washingtonpost.com/wp-dyn/content/article/2006/06/07/AR2006060702108.html`.

[59] David Malone and Matthew Luckie. Analysis of ICMP quotations. *Passive and Active Network Measurement*, pages 228–232, 2007.

[60] Arash Molavi Kakhki, Abbas Razaghpanah, Anke Li, Hyungjoon Koo, Rajesh Golani, David Choffnes, Phillipa Gill, and Alan Mislove. Identifying traffic differentiation in mobile networks. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 239–251. ACM, 2015.

[61] Christos Pappas, Katerina Argyraki, Stefan Bechtold, and Adrian Perrig. Transparency Instead of Neutrality. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, pages 22:1–22:7. ACM, 2015.

[62] Alina Quereilhac, Mathieu Lacage, Claudio Freire, Thierry Turletti, and Walid Dabbous. Nepi: An integration framework for network experimentation. In *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, pages 1–5. IEEE, 2011.

[63] Ashwin Rao, Justine Sherry, Arnaud Legout, Arvind Krishnamurthy, Walid Dabbous, and David Choffnes. Meddle: middleboxes for increased transparency and control of mobile traffic. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pages 65–66. ACM, 2012.

[64] Riccardo Ravaioli, Chadi Barakat, and Guillaume Urvoy-Keller. Chkdiff: checking traffic differentiation at Internet access. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pages 57–58. ACM, 2012.

[65] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Characterizing ICMP Rate Limitation on Routers. In *IEEE International Conference on Communications (ICC)*, 2015.

[66] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Towards a general solution for detecting traffic differentiation at the internet access. In *Teletraffic Congress (ITC 27), 2015 27th International*, pages 1–9. IEEE, 2015.

[67] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Testing for traffic differentiation with chkdiff: the downstream case. In *Teletraffic Congress (ITC 28), 2016 28th International*. IEEE, 2016.

[68] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end Arguments in System Design. *ACM Trans. Comput. Syst.*, 2(4):277–288, November 1984.

[69] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[70] RA Steenbergen. A practical guide to (correctly) troubleshooting with traceroute. *North American Network Operators Group*, pages 1–49, 2009.

[71] Srikanth Sundaresan, Walter De Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Measuring home broadband performance. *Communications of the ACM*, 55(11):100–109, 2012.

[72] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. Detecting network neutrality violations with causal inference. *ACM SIGCOMM CoNext*, page 289, 2009.

[73] Yves Vanaubel, Jean-Jacques Pansiot, Pascal Mérindol, and Benoit Donnet. Network fingerprinting: TTL-based router signatures. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 369–376. ACM, 2013.

[74] Udi Weinsberg, Augustin Soule, and Laurent Massoulié. Inferring traffic shaping and policy parameters using end host measurements. In *INFOCOM*, pages 151–155, 2011.

[75] Ronald W Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2):223–231, 1982.

[76] Tim Wu. Network neutrality, broadband discrimination. *Journal of Telecommunications and high Technology law*, 2:141, 2003.

[77] Ying Zhang, Zhuoqing Morley Mao, and Ming Zhang. Detecting traffic differentiation in backbone isps with netpolice. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 103–115. ACM, 2009.

[78] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network neutrality inference. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 63–74, New York, NY, USA, 2014. ACM.

List of Publications

[79] Riccardo Ravaioli, Chadi Barakat, and Guillaume Urvoy-Keller. Chkdiff: checking traffic differentiation at internet access. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pages 57–58. ACM, 2012.

[80] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Characterizing ICMP rate limitation on routers. In *IEEE International Conference on Communications (ICC)*, 2015.

[81] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Towards a general solution for detecting traffic differentiation at the internet access. In *Teletraffic Congress (ITC 27), 2015 27th International*, pages 1–9. IEEE, 2015.

[82] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Testing for traffic differentiation with chkdiff: the downstream case. In *Teletraffic Congress (ITC 28), 2016 28th International*, pages 1–9. IEEE, 2016.