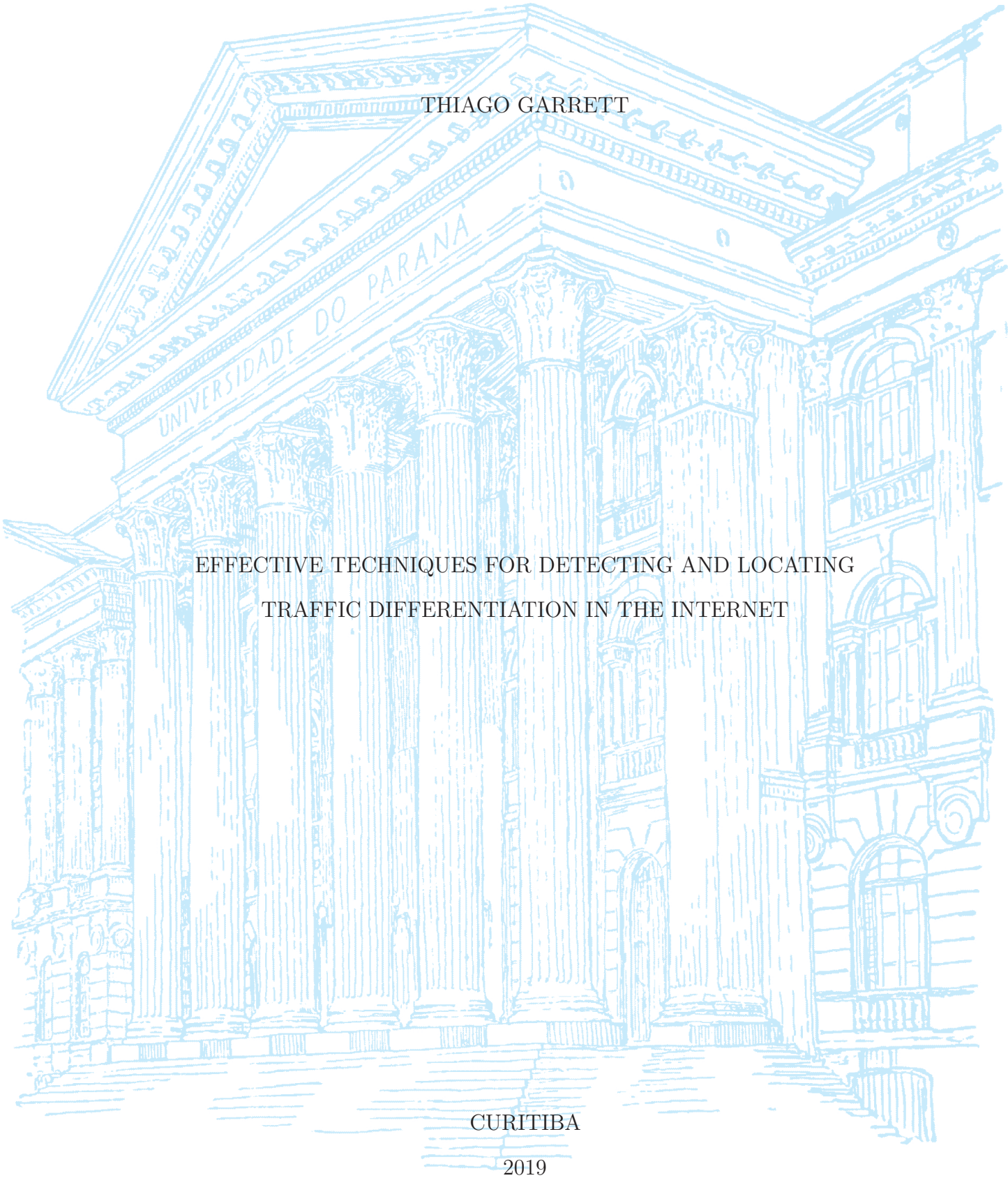
UNIVERSIDADE FEDERAL DO PARANÁ

THIAGO GARRETT



EFFECTIVE TECHNIQUES FOR DETECTING AND LOCATING TRAFFIC DIFFERENTIATION IN THE INTERNET

CURITIBA

2019

THIAGO GARRETT

EFFECTIVE TECHNIQUES FOR DETECTING AND LOCATING TRAFFIC DIFFERENTIATION IN THE INTERNET

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação.*

Orientador: Elias P. Duarte Jr.

Coorientador: Luis C. E. Bona.

CURITIBA

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

G239e Garrett, Thiago
Effective techniques for detecting and locating traffic differentiation in the internet [recurso eletrônico] / Thiago Garrett. – Curitiba, 2019.

Tese - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós- Graduação em Informática, 2019.

Orientador: Elias Procópio Duarte Junior. Coorientador: Luis Carlos Erpen Bona.

1. Internet das coisas. 2. Internet - Controle de acesso. 3. Redes locais de computadores. I. Universidade Federal do Paraná. II. Duarte Junior, Elias Procópio. III. Bona, Luis Carlos Erpen. IV. Título.

CDD: 004.678

Bibliotecária: Vanusa Maciel CRB- 9/1928

MINISTÉRIO DA EDUCAÇÃO
SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **THIAGO GARRETT** intitulada: **Effective Techniques for Detecting and Locating Traffic Differentiation in the Internet**, sob orientação do Prof. Dr. ELIAS PROCÓPIO DUARTE JUNIOR, que após após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 24 de Setembro de 2019.

ELIAS PROCÓPIO DUARTE JUNIOR
Presidente da Banca Examinadora

SCHAHRAM DUSTDAR
Avaliador Externo (TECHNISCHE UNIVERSITÄT WIEN)

LUIS CARLOS ERPEN DE BONA
Coorientador (UNIVERSIDADE FEDERAL DO PARANÁ)

ARTUR ZIVIANI
Avaliador Externo (LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIÊNTÍFICA)

WAGNER MACHADO NUNAN ZOLA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

Science knows no country, because knowledge belongs to humanity, and is the torch which illuminates the world. – **Louis Pasteur**

ACKNOWLEDGEMENTS

First and foremost, I sincerely thank my wife Erica for supporting me during all these years. I can not put into words how important it was being able to focus on my research without any other worries. I also apologize for all the bad nights of sleep, due to the noise from the computer running simulations. But she never complained, and I love her even more for that.

I also thank my advisors, Professors Elias and Luis, who taught me nearly everything I know about scientific research. I would not have been able to come this far without their advice, insights, and kind words.

I would like to extend my sincere gratitude to my parents, who always unconditionally supported me in anything I decided to do in my life.

I would also like to thank Professor Schahram Dustdar for receiving me in Vienna. My experience working with him was invaluable. I hope we can work together again in the future. In this context, I thank the Erasmus Mundus SMART^2 project (Project Reference: 552042-EM-1-2014-1-FR-ERA MUNDUS-EMA2), coordinated by CENTRALESUPELEC, which made my stay in Vienna possible.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. I would like to deeply thank everybody in CAPES that work hard to keep the Brazilian science running, despite all the attacks it suffers on the brink of any political or economic crisis. Absolutely no nation in the world prospered without investing in scientific research.

RESUMO

A Neutralidade da Rede torna-se cada vez mais relevante conforme se intensifica o debate global e diversos governos implementam regulações. Este princípio diz que todo tráfego deve ser processado sem diferenciação, independentemente da origem, destino e/ou conteúdo. Práticas de diferenciação de tráfego (DT) devem ser transparentes, independentemente de regulações, pois afetam significativamente usuários finais. Assim, é essencial monitorar DT na Internet. Várias soluções já foram propostas para detectar DT. Essas soluções baseiam-se em medições de rede e inferência estatística. Porém, existem desafios em aberto. Esta tese tem três objetivos principais: (i) consolidar o estado da arte referente ao problema de detectar DT; (ii) investigar a DT em contextos ainda não explorados, especificamente a Internet das Coisas (IoT); e (iii) propor novas soluções para detecção de DT que solucionem alguns dos desafios em aberto, em particular localizar a fonte de DT. Primeiramente descrevemos o atual estado da arte, incluindo várias soluções de detecção de DT. Também propomos uma taxonomia para os diferentes tipos de DT e de detecção, e identificamos desafios em aberto. Em seguida, avaliamos o impacto da DT na IoT, simulando DT de diferentes padrões de tráfego IoT. Resultados mostram que mesmo uma priorização pequena pode ter um impacto significativo no desempenho de dispositivos de IoT. Propomos então uma solução para detectar DT na Internet, que baseia-se em uma nova estratégia que combina diversas métricas para detectar tipos diferente de DT. Resultados de simulação mostram que esta estratégia é capaz de detectar DT em diversas situações. Em seguida, propomos um modelo geral para monitoramento contínuo de DT na Internet, que se propõe a unificar as soluções atuais e futuras de detecção de DT, ao mesmo tempo que tira proveito de tecnologias atuais e emergentes. Neste contexto, uma nova solução para identificar a fonte de DT na Internet é proposta. O objetivo desta proposta é tanto viabilizar a implementação do nosso modelo geral quanto solucionar o problema de localizar DT. A proposta tira proveito de propriedades de roteamento da Internet para identificar em qual Sistema Autônomo (AS) DT acontece. Medições de vários pontos de vista são combinadas, e a fonte de DT é inferida com base nos caminhos em nível de AS entre os pontos de medição. Para avaliar esta proposta, primeiramente executamos experimentos para confirmar que rotas na Internet realmente apresentam as propriedades requeridas. Diversas simulações foram então executadas para avaliar a eficiência da proposta de localização de DT. Resultados mostram que em diversas situações, efetuar medições a partir de poucos nodos no núcleo da Internet obtém resultados similares a efetuar medições a partir de muitos nodos na borda.

Palavras-chave: Neutralidade da Rede, Diferenciação de Tráfego, Medição de Rede

ABSTRACT

Network Neutrality is becoming increasingly important as the global debate intensifies and governments worldwide implement and withdraw regulations. According to this principle, all traffic must be processed without differentiation, regardless of origin, destination and/or content. Traffic Differentiation (TD) practices should be transparent, regardless of regulations, since they can significantly affect end-users. It is thus essential to monitor TD in the Internet. Several solutions have been proposed to detect TD. These solutions are based on network measurements and statistical inference. However, there are still open challenges. This thesis has three main objectives: (i) to consolidate the state of the art regarding the problem of detecting TD; (ii) to investigate TD on contexts not yet explored, in particular the Internet of Things (IoT); and (iii) to propose new solutions regarding TD detection that address open challenges, in particular locating the source of TD. We first describe the current state of the art, including a description of multiple solutions for detecting TD. We also propose a taxonomy for the different types of TD and the different types of detection, and identify open challenges. Then, we evaluate the impact of TD on IoT, by simulating TD on different IoT traffic patterns. Results show that even a small prioritization may have a significant impact on the performance of IoT devices. Next, we propose a solution for detecting TD in the Internet. This solution relies on a new strategy of combining several metrics to detect different types of TD. Simulation results show that this strategy is capable of detecting TD under several conditions. We then propose a general model for continuously monitoring TD on the Internet, which aims at unifying current and future TD detection solutions, while taking advantage of current and emerging technologies. In this context, a new solution for locating the source of TD in the Internet is proposed. The goal of this proposal is to both enable the implementation of our general model and address the problem of locating TD. The proposal takes advantage of properties of Internet peering to identify in which Autonomous System (AS) TD occurs. Probes from multiple vantage points are combined, and the source of TD is inferred based on the AS-level routes between the measurement points. To evaluate this proposal, we first ran several experiments to confirm that indeed Internet routes do present the required properties. Then, several simulations were performed to assess the efficiency of the proposal for locating TD. The results show that for several different scenarios issuing probes from a few end-hosts in core Internet ASes achieves similar results than from numerous end-hosts on the edge.

Keywords: Network Neutrality, Traffic Differentiation, Network Measurement

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AQM	Active Queue Management
AS	Autonomous System
c2p	Customer-to-provider
CAIDA	Center for Applied Internet Data Analysis
CDN	Content Delivery Network
CPU	Central Processing Unit
DNS	Domain Name System
DPI	Deep Packet Inspection
DT	Drop-Tail
ECMP	Equal-Cost Multi-Path
ED	Event-Driven
FCFS	First Come First Served
FIFO	First In First Out
FTP	File Transfer Protocol
HTC	Human-Type Communication
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
KB	Kilobyte
Kbps	Kilobit per second
KL	Kullback-Leibler
KS	Kolmogorov-Smirnov
M2M	Machine-to-machine
MB	Megabyte
Mbps	Megabit per second
MLab	Measurement Lab
MTC	Machine-Type Communication
MWU	Mann-Whitney U-test
NAT	Network Address Translation
NFV	Network Function Virtualization
NN	Network Neutrality
NS-2	Network Simulator version 2
p2c	Peer-to-customer

p2p	Peer-to-peer
P2P	Peer-to-Peer
PE	Payload Exchange
PU	Periodic Update
QoE	Quality of Experience
QoS	Quality of Service
RTO	Retransmission Timeout
RTP	Real-Time Transport Protocol
s2s	Sibling-to-sibling
SDN	Software-Defined Networking
SLA	Service-Level Agreement
SMTP	Simple Mail Transfer Protocol
SP	Strict Priority
SSH	Secure Shell
TCP	Transmission Control Protocol
TD	Traffic Differentiation
ToS	Type of Service
TPZ	Two-Proportion Z-test
TTL	Time To Live
UDP	User Datagram Protocol
VoIP	Voice over IP
VNF	Virtualized Network Function
VPN	Virtual Private Network
WFQ	Weighted Fair Queuing
WRED	Weighted Random Early Detection

LIST OF SYMBOLS

$f(l)$	Function $f : L \to R$ that maps a link $l \in L$ to the corresponding relationship $r \in R$
$P_{u,v}$	Set of all paths between ASes u and v
L_p	Sequence of links of a path $p \in P_{u,v}$
R_p	Sequence of relationships of the corresponding sequence of links, i.e. $\{f(l) : l \in L_p\}$
$V_{u,v}$	Set of all valley-free paths between ASes u and v
I_u	Inferred behavior of AS u
$I_{u,v}$	Inferred behavior of the pair of ASes (u, v)
σ	Maximum difference in size, relative to the shortest paths
$V^{\sigma}_{u,v}$	Set of valley-free paths between ASes u and v with size not larger than the size of the shortest path plus σ, i.e. $\{p' : p' \in V_{u,v}, \lvert p'\rvert \leq \lvert p\rvert + \sigma\}$, being $p \in V_{u,v}$ the shortest valley-free path
δ	Maximum valley-free distance from suspect AS i up to which measurement ASes are checked when searching for measurement pairs
X_E	Set of ASes present in the possible valley-free paths $V^{\sigma}_{u,v}$ between the ASes in the pair $E = (u, v)$

CONTENTS

1 INTRODUCTION

The Internet enables the interconnection of billions of individuals worldwide through providers operated by government, industry, academia, and private parties. Supporting the fast growth of the Internet is a challenge [1], not only due to technical issues, but also because of economical factors. For instance, in order to decrease and/or postpone investments in the network infrastructure, an Internet Service Provider (ISP) may employ discriminatory traffic management techniques [2]. Actually the motivation for the adoption of these practices can be manifold: ISPs may seek to obtain competitive advantages, increase the number of customers, or charge higher fees from users and content/service providers.

Discriminatory traffic management practices are applied to prioritize or degrade specific types of traffic over others [3] – e.g., based on content, protocol, origin or destination. These practices are often called Traffic Differentiation (TD). Note that TD can be applied for several reasons. It can be used to control congestion by throttling bandwidth-hungry applications, such as P2P file sharing and video streaming [4, 5]. TD can also be adopted because of commercial agreements, by which content/service providers pay extra fees to get their traffic prioritized, the so-called fast-lanes [6]. Other reasons include obtaining competitive advantage by which an ISP prioritizes the traffic of its own services and degrades (or even blocks) traffic from competitors [7, 8].

TD is part of the long and controversial debate regarding Network Neutrality (NN) [9]. A large number of countries worldwide have enforced NN with regulations [6]. Examples include Japan [10], Norway [11], Canada [12], Chile [13], Colombia [14, 15], South Korea [16], Brazil [17, 18], Mexico [19], USA [20], India [21], and the European Union [22]. A definition of NN that appears in these regulations states that, in a neutral network, every type of traffic must be treated equally, regardless of its origin, destination and/or content, i.e., TD is not allowed [23].

One of the central topics in the global NN debate is how to ensure that the Internet continues to be an environment that fosters innovation for all interested parties [24]. TD might threaten three concepts that have been considered essential for the success of the Internet [25]: innovation, fair competition, and consumer's freedom of choice. For example, TD would allow ISPs to control which services would have a better chance of succeeding, by prioritizing their traffic [2]. In such scenarios, new services and innovative solutions might struggle, since they would not be able to fairly compete against the already well-established services [26, 27]. On the other hand, less restrictions to the ISPs might result in a more competitive market [28, 29]. Some even argue that consumers should be able to decide which portion of their traffic is to be prioritized [30].

Furthermore, according to existing NN regulations [31], some traffic management practices are considered to be "reasonable" and are allowed, even if they prioritize or degrade different types of traffic [32, 33]. Usually, a traffic management practice is considered reasonable if it is beneficial to the network and its users as a whole. Examples include addressing illegal content (e.g., piracy, spam, or viruses), or prioritizing DNS queries. Another reasonable practice specified in some regulations is to prioritize the so-called *specialized services* [31], e.g., real-time health services, in order to meet their Quality of Service (QoS) requirements.

Regardless of regulations and the outcome of the global NN debate, we argue that the adoption of TD practices should be transparent, since they can significantly affect end-users and content/service providers. Regulations alone cannot guarantee ISP compliance, and even in a non-regulated environment, transparency should be a basic demand of ISP users as well as service/content providers. Note that there exist TD practices that are not covered by regulations [34, 35]. It is thus essential to monitor the presence of TD on the Internet [36]. The purpose is not only to increase transparency but also to check whether ISPs are complying with regulations.

However, detecting TD is not a trivial task [37]. ISPs may implement TD in a myriad of different ways. Traffic may be discriminated based on protocol, origin, destination, and payload, among other ways [38]. Furthermore, several techniques may be employed, such as traffic shaping [39], traffic policing [40] and even discriminatory internal routing [41]. Another challenge is to discover where within the network TD is taking place; this can be truly hard as there can be no prior knowledge of the internal structure of the network [42]. To make things worse, there are several other factors besides TD that can affect traffic performance and be misinterpreted as TD [37]. Examples include congestion, cross-traffic and load balancing.

A large number of strategies and tools have been proposed to detect TD [3, 37, 38, 41–48]. These solutions are based on network measurements and statistical inference. Since there is no way to determine the properties of an arbitrary network in a state that cannot be precisely described, existing solutions rely on end-to-end measurements to infer possible discriminatory behaviors. In general, they take measurements from one or several end-hosts, employing different types of traffic and probes. The measurements obtained are then analyzed to determine whether there was a significant difference over different sets of samples. Robust statistical models are necessary to distinguish between TD and performance variations caused by other phenomena.

Current solutions for TD detection are based on different assumptions, leading to different capabilities and limitations. Different solutions may detect different types of TD, using different techniques. For instance, several of these solutions generate synthetic traffic between two end-hosts that allow the comparison of the end-to-end performance of different applications. Some assume the existence of neutral traffic, which establishes a baseline that allows detection based on comparison results. Other solutions take measurements for individual hops along the route between two end-hosts, in an attempt to identify exactly where TD occurred. There are also solutions that passively capture the traffic from different applications, instead of generating traffic or issuing probes.

This thesis has three main objectives: (i) to consolidate the current state of the art regarding the problem of detecting TD in the Internet; (ii) to investigate TD on contexts/environments not yet explored by the current state of the art, in particular the Internet of Things (IoT); and (iii) to propose new solutions for monitoring NN in the Internet that address open challenges identified in the current state of the art, while taking advantage of the already exiting proposals; in particular, locating the source of TD.

The first objective resulted in a survey on TD detection [49], which presents a comprehensive view of the current state of the art. We present neutrality definitions, identify ways that ISPs may implement TD, define the problem of TD detection, describe multiple existing solutions for the problem, and compare the solutions. We also propose a taxonomy for the different types of TD and the different types of detection, and identify open challenges.

Our second objective refers to emerging technologies not yet explored by current solutions. Existing works do not address several of such new contexts, which may present different characteristics and requirements. One of these contexts is the IoT, which is expected to constitute a significant portion of the Internet in the future, both in terms of traffic and market share. For IoT to achieve its full potential, innovative solutions are necessary to address several open challenges. Since TD might hinder innovation, it can threaten IoT success. We thus investigate the impact of TD on common IoT traffic patterns [50], such as periodic updates and real-time notifications. We present simulation results, and discuss the vulnerabilities of IoT applications under TD. Results show that even a little prioritization may introduce a significant difference between different traffic priorities. This difference might greatly influence the Quality of Experience (QoE) perceived by end-users.

Our third objective is to propose a new solution for monitoring NN in the Internet, that address several of the open challenges identified. In order to pursue this objective, we first evaluate a solution for detecting TD between two end-hosts in the Internet, based on the techniques employed by the several existing solutions. This proposal consists of combining several different metrics to enable the detection of more types of TD. Simulation results show that this strategy was capable of correctly detecting TD under several conditions.

Next, we propose a general model for continuously monitoring TD [51]. The main goal of this model is to unify the existing solutions for detecting TD into a single framework, while taking advantage of both current and emerging technologies. In this context, we propose a strategy for identifying which Autonomous System (AS) is employing TD. This proposal combines TD detection results and/or measurements from several different sources in order to infer the location of TD in the Internet, at the AS-level. With this proposal, we aim at both enabling the implementation of the proposed general model, as well as addressing the problem of locating TD in the Internet. We further describe this proposal next.

There are still few proposals for locating the source of TD in the Internet. Some proposals [3, 42, 52] rely on path discovery techniques, such as the *traceroute* tool [53]. Unfortunately, these techniques may not succeed in obtaining the exact path between any given pair of end-hosts in the Internet, which may turn those proposals to locate TD ineffective. An alternative approach [46] assumes prior knowledge of the exact topology for the whole network, which may not be feasible in the Internet.

Our proposal for locating TD in the Internet does not rely on *traceroute*-like techniques, and does not require the exact host-level topology of the Internet, as previous proposals do. We assume prior knowledge of the AS-level topology of the Internet, instead of the host-level topology, which is a more realistic assumption. There are several datasets available that infer the AS-level topology of the Internet [54].

The proposal takes advantage of Internet peering to identify which Autonomous System (AS) is practicing TD. The main idea is to investigate ASes suspected of being the sources of TD until only the AS that is actually discriminating traffic is filtered out. This is done by combining TD detection results regarding different pairs of end-hosts, and making inferences based on the possible AS-level paths between the pairs. The rationale is that if a particular AS is in all possible paths between two end-hosts, then the TD detection measurements are guaranteed to have traversed that AS, and thus its behavior can be assessed.

AS-level paths in the Internet follow a set of routing policies based on the relationship between ASes [55], i.e., how they exchange traffic. Therefore, we examine all possible AS-level paths between end-hosts according to inter-domain routing policies, instead of running measurements with *traceroute*-like techniques. Furthermore, we also present metrics for defining where the measurement points should be located, since the effectiveness of our proposal depends on how well positioned measurement points are.

We evaluate our proposals for locating TD using two different sets of experiments. The first set consists of experiments executed on the PlanetLab global testbed [56]. The goal of these experiments is twofold: (i) to check our assumptions regarding the properties of AS-level paths; and (ii) to confirm the limitations of *traceroute*-like techniques for obtaining such paths. In the second set of experiments, we describe several simulations for assessing the efficiency of our proposal for locating TD under different scenarios.

The results obtained for the PlanetLab experiments confirm that the majority of successfully observed AS-level paths complied with the properties assumed. Furthermore, the experiments show that path discovery techniques such as *traceroute* may not be reliable, which further motivates our approach. The simulations show that our proposal is capable of locating TD between several different pairs of ASes, under different conditions. Furthermore, on a variety of scenarios issuing measurements from a few core Internet ASes achieves similar results to issuing measurements from a large number of ASes on the edge. Results also show which metrics are better for selecting measurement points under different scenarios.

The main contributions of this thesis are the following:

- A comprehensive view of the current state of the art regarding the problem of detecting TD is presented, describing multiple solutions, comparing them, and identifying open challenges.
- A taxonomy for the different types of TD and the different types of detection is proposed.
- We evaluate the impact TD may have on IoT traffic.
- A general model for continuously monitoring NN, which aims at unifying the current solutions for TD detection is proposed.
- We present evidence that *traceroute*-like techniques may not be reliable.
- A new strategy for locating TD in the Internet that addresses the shortcomings of previous solutions is proposed.
- We make an innovative use of AS-level routing properties.
- Metrics for choosing good measurement points are defined and evaluated.

The rest of this thesis is organized as follows. We first present concepts that serve as a basis for the rest of this work in Chapter 2. Then, we describe the state of the art in Chapter 3, presenting several existing solutions for the problem of detecting TD, as well as a taxonomy, a comparison of these solutions, and open challenges. An investigation of the impact of TD on IoT is then presented in Chapter 4. Next, we describe a proposal for detecting TD in the Internet in Chapter 5. We then describe our general model for continuously monitoring NN, as well as the proposal for locating TD in Chapter 6. Finally, we conclude the thesis in Chapter 7.

2 TRAFFIC DIFFERENTIATION IN THE INTERNET

In this chapter we present the main definitions and concepts that are used in the rest of this thesis. In Section 2.1, we first give a brief overview of Traffic Differentiation (TD) in the Internet, in the context of Network Neutrality (NN). Next, we define NN and the meaning of a "neutral network" in Section 2.2. Different ways ISPs may implement TD is then described in Section 2.3. Finally we define the problems of detecting and locating TD in the Internet in Section 2.4.

2.1 OVERVIEW

The Internet is a global network that consists of several interconnected Autonomous Systems (ASes) [57]. Each AS comprises a collection of Internet routing prefixes and is controlled by an administrative entity called Internet Service Provider (ISP). ISPs are hierarchically organized in three tiers. Tier 1 ISPs correspond to the core Internet backbone, which consists of high performance networks which interconnect the tier 2 ISPs on a global scale. Tier 2 ISPs provide global connectivity to the tier 3 ISPs, which provide Internet access to end-hosts. An end-host is any computer or device connected directly to a tier 3 ISP or a gateway providing Internet connectivity to a local network. End-hosts form the so-called edge of the Internet, while Tiers 1 and 2 ISPs form the so-called core of the Internet.

The Internet was originally designed following two principles that are essential in the context of NN [58]: the *end-to-end* principle and the *best-effort* principle. The end-to-end principle states that messages exchanged between two end-hosts are sent in packets that are forwarded by autonomous routers. A router simply forwards a packet to the next hop so that the packet will reach the destination through the shortest path. In particular, a single router cannot define or control the complete route that a packet traverses from the origin to the destination. The best-effort principle states that every packet must traverse the network as fast as possible. A router employs a queue to manage the incoming packets. If the queue grows and uses all the space available, the router should drop the next incoming packets, regardless of their content, origin, destination or any other feature.

Actually, there exist several different scheduling algorithms both for dropping arriving packets and for determining which packets should be forwarded and removed from the buffer. Some of the most common types of schedulers are [38, 45]: (i) First Come First Served (FCFS), in which the packets that arrived first are forwarded first; (ii) Strict Priority (SP), in which the scheduler always give priority to a specific type of traffic; (iii) Leaky Bucket, in which maximum rates are defined for each type of traffic; (iv) Token Bucket, in which a limit is defined for the average rate of each type of traffic; (v) Weighted Fair Queuing (WFQ), in which the maximum rates for the different types of traffic are determined using weights; (vi) Drop-Tail (DT), which drops all new incoming packets when the buffer is full; and (vii) Weighted Random Early Detection (WRED), in which low priority packets have a higher probability of being dropped.

We call *neutral schedulers* those that do not differentiate traffic. FCFS and DT, for example, are neutral schedulers. Non-neutral schedulers are those that may be employed to discriminate between different types of traffic, either by dropping or delaying packets that

are classified as low priority. For example, the Leaky Bucket scheduler might be employed to enforce a maximum rate of some specific type of traffic. Active Queue Management techniques (AQM) [59] may also be employed to differentiate traffic.

As mentioned in the Introduction, it is important to make it clear that according to existing NN regulations [31], some traffic management practices are considered "reasonable" even if they prioritize or degrade different types of traffic. These are thus exceptions in which TD is allowed [32, 33]. Mostly these fall into two categories: either the traffic management practice is considered reasonable because it is beneficial to the network and its users as a whole (e.g., addressing illegal content, viruses and spam) or TD is employed to prioritize *specialized services* [31] in order to meet their QoS requirements (e.g. real-time services). The main focus of this thesis is on detecting and locating TD, thus determining whether TD is legal/beneficial is out of the scope of this work.

2.2 NETWORK NEUTRALITY DEFINITIONS

There is no unique definition for NN, several different definitions can be found in the literature [23, 60–62]. However, it is possible to state that most definitions, including those employed by regulations worldwide take into account whether TD is going on in the network. Thus, a network is defined as neutral if all data packets are treated equally in that network, i.e., unreasonable TD practices are not allowed. Therefore, an Internet Service Provider (ISP) cannot slow down, prioritize or block any type of specific traffic, regardless of its origin, destination and/or content. In addition, a NN *violation* corresponds to any unreasonable practice that causes some particular traffic to be treated differently from others.

The Internet end-to-end and best-effort principles are behind the NN definition, and they imply that all routers should forward every packet in a neutral fashion, without prioritizing any subset of packets over others [58]. Every type of traffic is then subject to the same conditions. Therefore, in a neutral network, all routers must employ neutral schedulers. For instance, if all routers in an AS employ only the FCFS and DT schedulers, the network is neutral [38], since packets will always be forwarded and dropped (if the buffer is full) in the order they arrive, regardless of other features.

2.3 IMPLEMENTING TRAFFIC DIFFERENTIATION

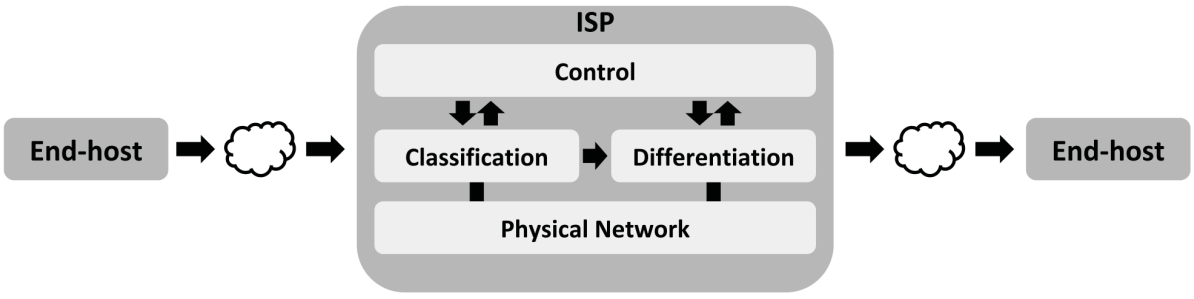


Figure 2.1: High-level description of TD on an ISP network.

There are several mechanisms for implementing TD in a network. Each ISP might employ different mechanisms that better suit its own interests and the characteristics of

its network. However, regardless of the specific mechanisms, TD can be employed by an ISP in one or several routers, at the ingress or egress points, or by internal routers.

Figure 2.1 is a high-level illustration of how TD can affect traffic between two end-hosts. The figure is agnostic to the actual characteristics of the network and specific TD mechanisms employed. The ISP network employing TD is divided into different logical components. Any traffic traversing the AS is classified and treated accordingly to the assigned class. A control component defines how traffic is classified and differentiated. These logical components run on top of the actual physical network and may be implemented in several different ways.

Traffic classification may be based on several criteria [63, 64], such as TCP/UDP port, source address, destination address, application protocol, information obtained from deep packet inspection (DPI), the previous-AS from which the traffic came or the next-AS to which the traffic will be forwarded, traffic performance or behavior, a combination of these or any other more complex criteria. Furthermore, these criteria may change over time. Based on the classification, traffic is treated differently according to the assigned class.

There are several ways for implementing traffic classification on a real network. For instance, classification can be run at the ingress point of an AS and class information can be inserted in the packet header (of some AS-internal protocol), informing the following routers how that packet/traffic should be treated. Another strategy is to configure all routers to both classify and discriminate traffic.

TD may also be employed using several different mechanisms and deployed in several different configurations. Furthermore, these mechanisms may change over time. For instance, as with classification, TD may be performed only at the ingress point of an AS, by all or by some routers in the network. Another possibility is to have specialized devices deployed on middleboxes [65] to perform TD.

The most common TD mechanisms are traffic shaping [39] and traffic policing [40]. These mechanisms differ in the way routers process incoming packets given their classes. Traffic policing employs non-neutral schedulers to limit the rate of "low-class" traffic by dropping packets more often. Traffic shaping limits the traffic rate by delaying "low-class" packets, employing schedulers that prioritize "high-class" traffic when forwarding or dropping packets. Other examples of TD mechanisms include: forged TCP reset (RST) packet injection forcing TCP connections to abruptly end; and forwarding traffic through separate routes depending on their classes, some of them presenting higher performance, the so-called fast-lanes [6].

Different TD mechanisms may affect traffic in different ways. For instance, traffic policing may result in larger loss rates, while traffic shaping may result in longer delays. If packets of two different classes are forwarded along two different routes, and just one of them is congested, the packets will experience significantly different delays and loss rates.

An ISP may dynamically control how traffic is classified and differentiated in its network. There are different possible approaches for implementing the control module of Figure 2.1. For instance, it can be a person or a system that automatically reconfigures routers and other devices in the network according to some predefined criteria. We note that *Software-Defined Networking* (SDN) is a technology that allows sophisticated classification and differentiation mechanisms to be easily deployed and managed [66].

2.4 PROBLEM DEFINITIONS

In this thesis we address the problem of detecting and locating TD in the context of NN. We assume an external observer that does not have access to network configurations and internals. The problem of detecting TD consists of inferring whether some given network traffic is being treated differently from other traffic. In other words, the TD detection problem consists in determining whether different types of traffic are experiencing different performance levels due to some type of prioritization in the network only because of their different features (e.g., source, destination, port, content, etc.). A related problem is to identify exactly which features are triggering TD. The problem of locating TD consists of identifying where (in which AS or ASes, router or routers) TD occurred.

TD may not always impact the traffic traversing a network, such as when there is not much traffic and the discriminatory practices do not result in any extra delay or loss. In such cases, we say that TD is *non-observable*, since it is not feasible to infer whether TD is being employed, at least based only on external observations. Similarly, TD is *observable* when it effectively impacts the network traffic, for example by increasing the delay or loss rate.

3 STATE OF THE ART

In this chapter, we present the state of the art for the problems defined in Chapter 2. First, we describe several existing solutions for detecting TD in Section 3.1. We then define a taxonomy for the different types of TD and the different types of detection in Section 3.2. Then, in Section 3.3, we consolidate the state of the art by identifying the most common techniques employed by current solutions, and the main challenges they face for detecting TD. We also compare the solutions according to our taxonomy and common features. Finally, we identify open challenges in Section 3.4. An earlier version of this chapter was published in [49].

3.1 A SURVEY OF SOLUTIONS FOR TD DETECTION

In this section, we present a survey of solutions for the problem of detecting TD on the Internet. These solutions include both tools and strategies. They were designed with different assumptions and goals, and thus employ different techniques to achieve their goals. All solutions rely on network measurements and infer TD. This is often done by checking whether different types of traffic are treated differently while traversing the network.

The rest of this section is organized as follows. In Subsections 3.1.1 to 3.1.12, we describe several existing solutions for the problem of detecting TD on the Internet, in order of publication date. Finally, in Subsection 3.1.13 we present an overview of several other works related to monitoring NN.

3.1.1 Gnutella Rogue SuperPeer

The *Rogue SuperPeer* (RSP) [47] is a strategy to measure port blocking in the Internet. This strategy detects whether traffic on specific ports (corresponding to specific applications or classes of applications) is being blocked between end-hosts and a measurement host. Port blocking is an important and straightforward strategy that can be used by network operators to control which types of traffic are allowed on their networks. Although it can be used for fair reasons, such as to block worms, it can also be used for anti-competitive or economic purposes, for example an operator can block services with which it is competing.

The main principles behind the design of the Rogue SuperPeer are: generality, range, quantity, and minimal participation, described as follows. By generality the authors mean that any arbitrary TCP or UDP port number (from 0 to $2^{16} - 1$) can be tested. Range means that a large range of networks across the Internet are tested. Quantity means that a large number of hosts are tested. Finally, minimal participation means that the participation is not active, coordinated, or cooperative, users are engaged in the process of testing without even noticing it.

The RSP infrastructure consists of two parts: the Rogue SuperPeer itself and a measurement host. The Rogue SuperPeer itself is a superpeer of a P2P network, in this case the authors used Gnutella[1]. This superpeer joins the network and is advertised as any other superpeer. When a new peer connects to the RSP, it issues queries and responses according to the normal protocol. However, the process is slightly modified so that these new peers will trigger port blocking measurements. The main idea is to induce a large

[1] `http://www.gnutellaforums.com`

number of globally distributed hosts to attempt connecting to a specific IP address using the TCP ports being evaluated.

In order to understand RSP it is necessary to first understand the basics of the underlying network. Gnutella is a P2P network comprised of two types of processes: superpeers and peers (also called clients or leaves). Each superpeer is connected to other superpeers and to a set of peers. In order for a new peer to join the overlay network, it first contacts a superpeer. The superpeer may then accept or reject the new peer connection. If the peer request is accepted, it stays connected to that superpeer. However, if the connection attempt is rejected, the superpeer replies with a "busy" message. This response includes an indication of other superpeers (IP/port) that might be contacted by the new peer to join the network.

The RSP strategy works like this: after a new peer sends a connection request to the Rogue superpeer, the superpeer sends back a "busy" reply, and refers the new peer to the measurement host, using a particular port to be evaluated. Figure 3.1 illustrates the RSP strategy. A new peer sends a connection request to the RSP (1). The RSP then refuses the new peer, replying with a busy message referring the new peer to the IP address of the measurement host and the port to be evaluated. The new peer then initiates a connection with the measurement host (2). The port number referred by the RSP changes every 5 minutes, in order to evaluate a large number of ports. The measurement host and the superpeer both register incoming connections from the new peers.

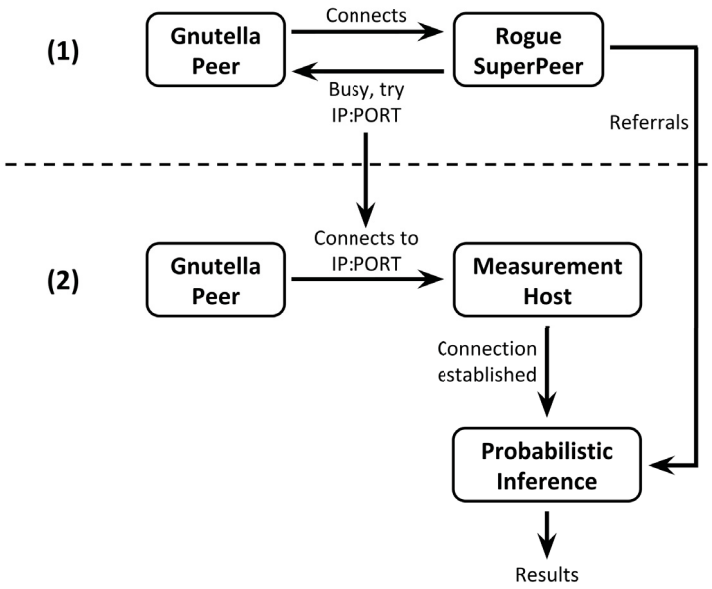


Figure 3.1: Gnutella RSP strategy.

Determining whether a port is *not* blocked is done as follows. If at least one peer, redirected by the RSP, successfully connects to the measurement host, then the port used for this connection is not blocked. However, if no peer connects to the measurement host on a given port referred by the Rogue superpeer, there are two possibilities: either all peers ignored the referral, or the port is blocked. The authors empirically concluded that the probability of a new peer ignoring the RSP referral is about 80%. The authors then determined that at least 50 referrals are necessary to infer that a port was blocked, with a confidence level of 99.5%.

Experiments with the RSP strategy were executed for 2 months. During this period, approximately 150,000 referrals were generated for about 72,000 distinct Gnutella peers, which were distributed in approximately 31,000 different prefixes, which can be considered a significant sample from the Internet. The results show that of the 31,000 prefixes, in at least 256 a port was blocked. The most frequently blocked port was 136 and those blocked less frequently were 80 (HTTP), 6346 (Gnutella), and 6969 (which was used for comparisons). Some email ports (25, 110 and 143) were blocked twice as often as the comparison port 6969. The other most frequently blocked services were: FTP, SSH, BitTorrent and VPNs. The authors also report that some universities and ISPs blocked ports often used by P2P networks (1214, 4662, 6346, 6881). Furthermore, some ISPs in Canada, the USA and Poland blocked Skype ports.

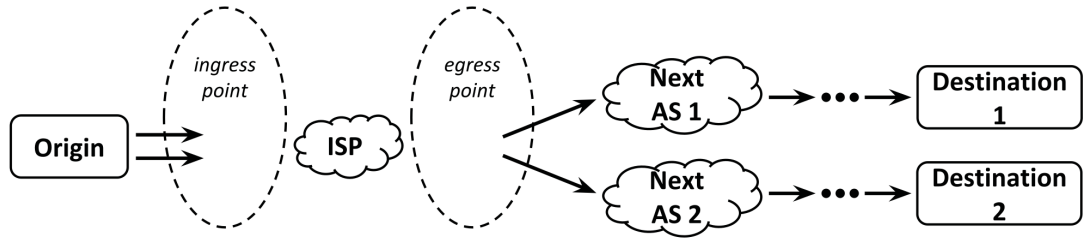
The RSP strategy addresses a specific case of TD, which is port blocking. It also addresses the problem of locating which ISP is performing TD, by aggregating several measurements from a given prefix. The strategy is based on hybrid active/passive measurements and must be executed on a P2P network. However, some issues are not addressed by the authors. For instance, the Gnutella RSP strategy cannot always tell whether a port blocking is being performed by the peer ISP or by the ISP of the measurement host. Furthermore, an ISP may be blocking all Gnutella traffic based on the application protocol, regardless of port numbers.

3.1.2 NetPolice

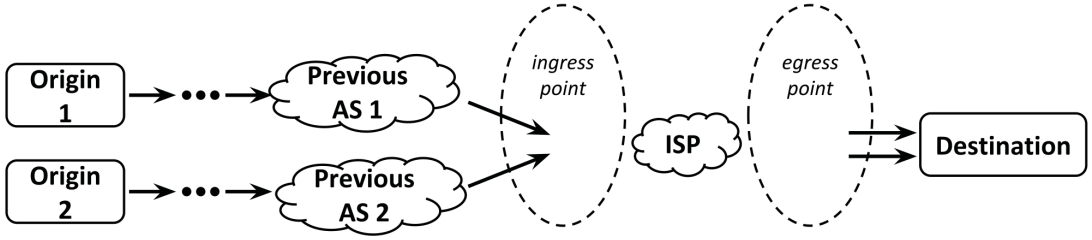
NetPolice [42] (a previous version of which was named NVLens [67]) is a tool for detecting TD in the backbone of the Internet (Tier 1). The authors argue that when TD is executed in the backbone the impact is stronger than when it is executed by ISPs that are closer to the border, since TD in the backbone potentially affects more traffic. NetPolice is able to locate which ISP is performing TD. The tool measures the loss rate experienced by different types of traffic, sent from multiple sources, as they traverse a target ISP. TTL-based probes are employed in order to discover paths traversed by packets in the network, including the internal path of the target ISP.

NetPolice detects TD triggered by content and routing, assuming that traffic is classified based on header, payload, or using routing policies. Figure 3.2 shows how NetPolice detects different types of TD. In Figure 3.2(a), measurements are made using the same source to multiple destinations. These destinations are chosen so that after the packets leave the target ISP they enter different ASes. This strategy allows NetPolice to check if the target ISP is employing TD based on the next AS packets are headed to. In Figure 3.2(b), measurements are made using a single destination and multiple sources, selected in such a way that the packets entering the target ISP come from different ASes. This allows the tool to check if the target ISP is employing TD based on the previous AS from which the packet came. Figure 3.2(c), shows yet another case in which measurements are made using the same source and the same destination, but traffic is generated for multiple applications (changing destination port or payload). This allows the tool to detect TD triggered by application/content.

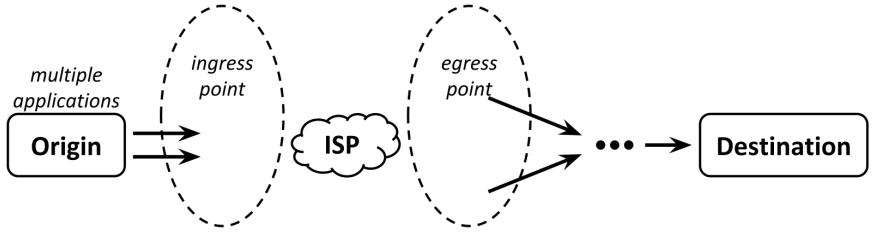
The strategy employed by NetPolice to detect TD is based on 4 steps and is shown in Figure 3.3. The first step consists of discovering paths that traverse the target ISP, from multiple origins. A large number of route traces (using for example the *traceroute* tool [53]) are issued from multiple sources to a large number of Internet destinations (prefixes). This process allows NetPolice to estimate the distances between ingress and egress points of the target ISP, as well as the previous AS from which packets come to each ingress point,



(a) TD based on next AS.



(b) TD based on previous AS.



(c) TD based on destination port or payload.

Figure 3.2: NetPolice detects different types of TD.

and the next ASes from the egress points. With this information NetPolice pre-computes the TTL values between each pair of ingress/egress points with respect to the target ISP, from all sources. The set of paths discovered and other information obtained in this step is called "path view".

The second step consists of the selection of a set of paths on which measurements are going to be executed. The paths are a subset selected from the "path view", as it is unfeasible to run measurements on all paths. This set of paths should give a good coverage of the internal network of the target ISP. In order to avoid unnecessary work, the choice of on which paths measurements will be executed must be done carefully, in particular to avoid source and destination pairs that pass through the same ISP internal paths or paths that do not traverse the target ISP. This selection is modeled as an optimization problem, with the following constraints: each tuple (origin, input, output) must be traversed at least R times by paths to different destinations; each tuple (input, output, destination) must be traversed at least R times by paths from different sources; finally, there can be no more than m paths from the same source. The set of paths on which measurements are executed is called "tasks" and is sent to the next step of NetPolice.

Measurements are executed in the third step, using traffic generated by different applications: HTTP, BitTorrent, SMTP, PPLive and VoIP. Measurements are executed as follows. Periodically, each 200 seconds, and for each application, two measurement probes are sent: one with the TTL field set so that its reaches the ingress point (in), and another with the TTL field set to reach the egress point (eg). The loss rate for the internal path of

the target ISP is then obtained by subtracting the loss rate measured for the egress point from the loss rate measured for the ingress point.

Finally, in the fourth step, NetPolice uses the obtained measurements to infer whether the target ISP is employing TD based on content or routing. The inference employs the KS (Kolmogorov-Smirnov) test in order to compare the distributions of the measured data. The detection of TD by content is then done by comparing the data distributions of each application with the distribution of HTTP application data. NetPolice assumes that HTTP traffic is a baseline for detecting TD, i.e., HTTP traffic is assumed not to be discriminated. KS tests are applied to determine if the measurement data obtained for a given application is significantly different from the data measured for the HTTP-based application, thus characterizing TD. TD based on routing is detected in a similar way, but comparing the distributions for data obtained for different paths but carrying the same application.

The authors discuss several strategies for reducing noise. The inaccuracy of loss rate measurements can be caused by an overloaded prober, especially due to high CPU utilization. The authors mention that a reasonable limit is 65% for CPU utilization. Another factor to consider is whether ICMP rate limiting practices are being used, which is often done to prevent router overload; NetPolice avoids this problem by keeping a large probing interval, on the order of hundreds of seconds. Another noise reduction factor considered is the loss on the reverse path. As NetPolice uses single-ended probes to measure loss rates, they can be inflated due to reverse path losses. The authors report an experimental result that the loss rate increases with the packet size; they thus use the loss rate measured by 40-byte probe packets as the upper bound of the loss rate on the reverse path. Finally, although some ISPs perform load balancing using Equal-Cost Multi-Path (ECMP) between a pair of ingress/egress points to improve the performance, this can be a problem given the measurement strategy of NetPolice, it was not detected in any ISP evaluated.

Experimental results reported for NetPolice were obtained in the PlanetLab. 18 ISPs distributed across 3 continents were evaluated over a period of 10 weeks. The results show that 4 ISPs performed TD on 4 applications and 10 ISPs performed TD based on the previous AS of the packets. The packet loss rates measured in these cases were up to 5% different. The authors also observed, from the results obtained, that TD can depend on the load of the network. For some ISPs, NetPolice detected that the values assigned to the ToS field of the IP packet header were strongly related to TD, and the values assigned were usually related to the destination port not on content (thus DPI was not done). Another observation was that different routers do not apply TD in the same way.

NetPolice addresses both the problem of detecting TD and the problem of locating which ISP is performing TD. It is also one of the few solutions that detects TD triggered by the path traversed by the traffic. Note however that the TTL-based probing techniques employed may result in false-negatives, e.g., when ICMP is not supported. Another possible limitation of the NetPolice strategy is the set of paths on which measurements are executed. The paths traversed between the same origin and destination, by different types of traffic, may not be the same. Thus, the paths obtained in the path discovery step may not be the same path traversed when the measurements are made for different applications. For instance, the packets may be forwarded to an egress point that is different from the one that is expected.

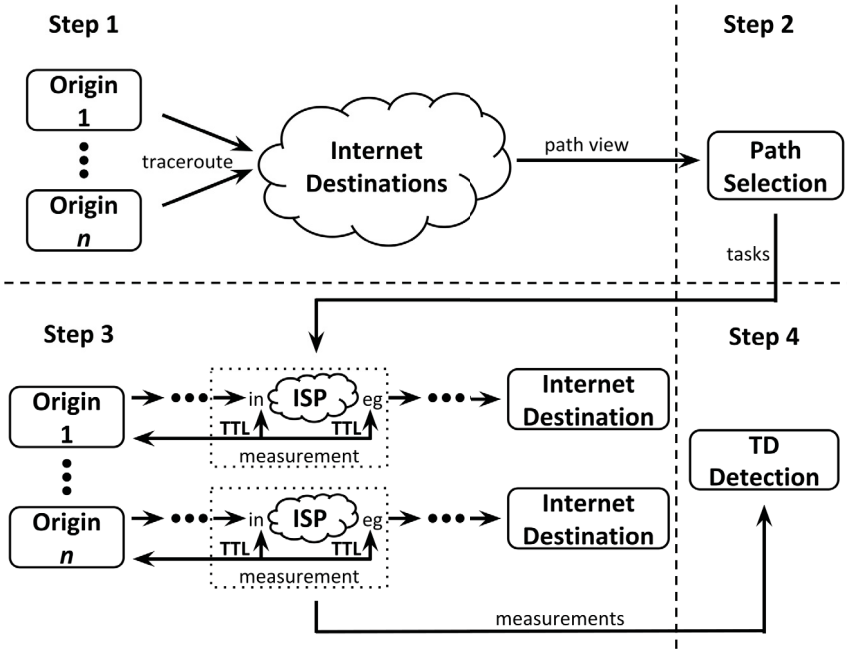


Figure 3.3: NetPolice TD detection.

3.1.3 NANO

NANO (Network Access Neutrality Observatory) [37, 68] is the first system that detects neutrality violations which does not test specific applications/ports nor specific discrimination mechanisms. NANO infers whether an ISP is discriminating traffic based on performance data obtained passively. If the performance of an application measured in an ISP network is statistically significantly lower than the performance of the same application measured in the networks of other ISPs, it is possible that TD is being employed. NANO uses a causal inference model to establish a relationship between the observed performance degradation and the ISP policies. NANO employs passive monitoring, i.e., it is based on measurements of the real traffic of the observed applications while they are running.

Some of the main features of NANO for TD detection are, according to the authors: (i) several other strategies detect discrimination based on specific traffic characteristics such as port or content, whereas NANO has a more general approach, measuring the performance of the applications regardless of the specific TD mechanisms employed by the ISPs; (ii) the fact that NANO passively monitors traffic makes it more difficult for ISPs to detect that NANO is being used and employ techniques to deceive the TD detection system; and (iii) while NANO compares metrics from the same application executed on different ISPs, several other solutions compare different application metrics in the same ISP.

NANO's TD detection strategy presents three major challenges: (i) the TD mechanism employed by the ISP is not known in advance, so the detection strategy must be generic; (ii) the standard performance of an application at a particular ISP is not known beforehand, making it difficult to detect possible degradations, since there is no baseline for comparison; and (iii) many factors other than TD can cause application performance degradation, such as overhead, geographic location, the particular software and hardware being used, and other network features.

The different factors, besides TD, that can cause degradation of the performance of an application, are represented in the statistical model used by NANO as confounding

factors [69]. It is necessary to identify the confounding factors and to collect data not only from the applications, but also from these confounds. The TD detection strategy used by NANO is therefore based on the comparison of the performance of the same application executed on different ISPs using measurements with similar confounds. An example of a confounding factor is the time of day: one should not compare measurements taken at different times, since application performance usually varies depending on the time of the day (due to a higher/lower system load, for example).

NANO uses a stratification technique [70] to group performance measurements according to the confounds. This technique places measurements in strata, so that the confounds for the measurements in each stratum have similar values. Three types of confounding factors are defined: (i) client-related confounds (examples include software that may affect the performance of the measured application, such as the operating system or a specific Web browser); (ii) network-based confounds which are related to the network (such as properties of the network path, or geographic location, for example); and (iii) time-based confounds (such as time of the day, for example, that can affect the performance of the application being measured).

After the stratification, NANO estimates for each stratum how much the performance of the application changes when accessed through a specific ISP, in comparison with the performance observed when not using that ISP, which is called the baseline performance. The average performance is computed as the average performance of the application executed on all other ISPs within the same stratum, except the ISP being evaluated. These estimates represent a quantification of the causal relationship between each ISP and the possibility that TD is being employed.

In the last step of its TD detection strategy, NANO aggregates the estimates of all strata and verifies if the values obtained are statistically significant. The central idea is that if, on average, the performance of an application was significantly degraded on a specific ISP, then there is a causal relationship between the ISP and the practice of TD.

NANO is implemented in two parts: the agents and a server. An agent runs on each client host and is responsible for monitoring application performance by measuring real traffic generated by that client host. The metrics employed are specific to each application, whichever is most appropriate. In addition to application performance data, agents also collect data about the confounding factors. All data acquired by agents is periodically sent to the server. Agents are implemented as network sniffers, analyzing all packets received and sent by the host. The NANO server receives all data collected by the agents and is responsible for performing TD detection based on this data.

Figure 3.4 shows how NANO works. Agents are deployed on several end-hosts, each host executes an agent (1), being responsible for passively monitoring the performance of running applications being evaluated. Data obtained by all agents is sent to a server, which classifies them in strata according to the confounds (2). Finally, the server infers (3), according to a causal model, which ISPs employed TD for each application being evaluated.

To evaluate NANO, the authors executed experiments using the PlanetLab and Emulab testbeds. Nodes geographically distributed across PlanetLab were employed. These nodes executed servers running the applications being evaluated. A set of ISPs was created in Emulab, each with a different set of clients. Each ISP provided Internet connectivity to its clients. Thus, clients could only access the applications hosted on PlanetLab nodes through the ISPs, allowing the emulation of different TD practices and different confounding factors.

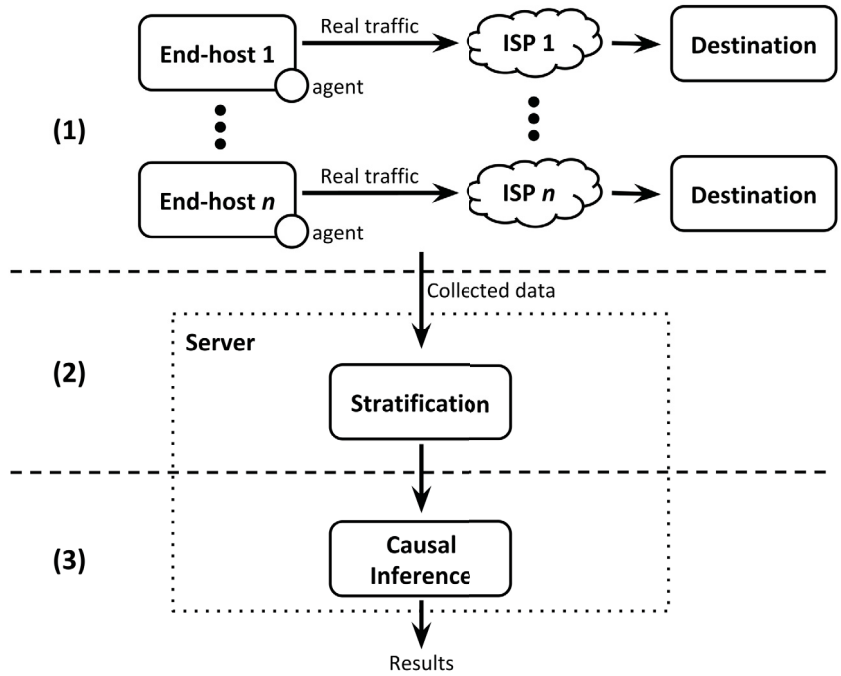


Figure 3.4: NANO TD detection.

Experimental results show that NANO is able to detect TD in different ways and for different types of applications, provided that all confounding factors are known and measured. The NANO detection strategy proved to be generic enough, detecting traffic discrimination even without prior knowledge of which TD policies were employed by the ISPs. However, if NANO does not take into account all confounds, the causal relationship between the ISP and TD can lead to mistakes, either false-negatives and false-positives. As it is impossible to identify *all* relevant confounding factors or even to decide whether a given set of confounding factors is complete enough, it is not straightforward to apply NANO to detect TD in complex real networks.

A solution similar to NANO was proposed in [71] by different authors. In this more recent work, the same stratification technique was employed for grouping measurements from several different users. However, in addition to TD detection, the authors also propose a strategy for ensuring the privacy of the measurement data acquired by the system. The proposal guarantees privacy under the differential privacy framework [72]. The authors demonstrate that their approach is able to ensure privacy while still achieving a good TD detection accuracy.

3.1.4 POPI

POPI [41] is a tool based on end-to-end measurements to detect whether non-neutral schedulers are being employed by an ISP. In particular, POPI detects whether packets of different types are being forwarded with different priorities, i.e., POPI detects packet forwarding prioritization. The authors assume that TD is performed by using non-neutral schedulers, which were presented in Chapter 2. If only neutral schedulers are employed, then all packets are forwarded according to the arrival order. On the other hand, if non-neutral schedulers are being used, the loss rates will be different for different types of traffic under congestion. POPI measures the loss rate for different types of traffic to infer whether different priorities were assigned for the types of traffic measured.

POPI works in three steps, shown in Figure 3.5. In the first step (1) measurements are obtained after a series of packet bursts are injected. In the second step (2) the measurements are used to order the types of traffic with larger loss rates for each burst. In the third step (3), a statistical analysis is made to detect the prioritization of specific types of traffic during the bursts. Each step is described in more details below.

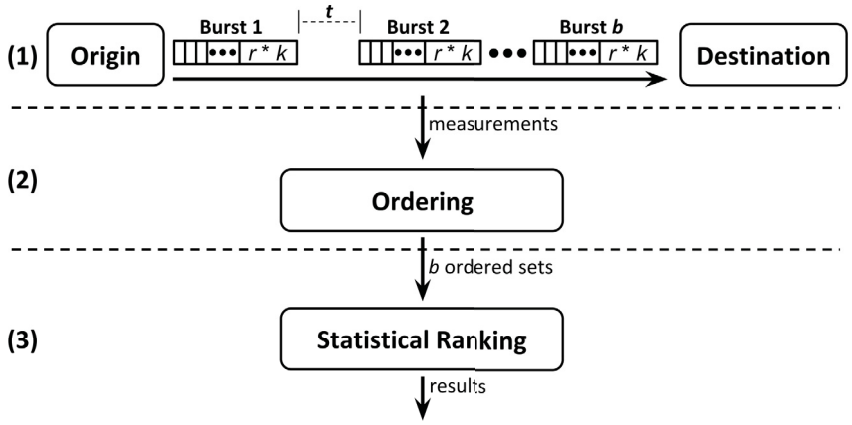


Figure 3.5: POPI TD detection.

In the first step (1), POPI measures the loss rate for k different types of traffic, generated with different destination ports and/or payload. These measurements are made on b bursts of packets, between a pair of end-hosts. The bursts are triggered in intervals of t seconds. Each burst is composed of r rounds, and in each round k packets are sent – one for each type of traffic, in random order. Thus in each burst $r * k$ packets are sent in sequence. According to the authors, the value of t should not be too low, so that a burst does not interfere with the next, but should also not be too high, so that the whole measurement process ends too early, making it less susceptible to cross-traffic variations. The authors also claim that it is necessary to send a large number of packets to saturate the path between the end-hosts causing congestion and thus causing packets to be dropped.

In the second step (2), the loss rates for all types of traffic in each burst are computed and ordered. The traffic type with the largest loss rate of any given burst is placed first, the traffic type with the second largest loss rate is placed next, and so on. According to the authors, if all types of traffic are treated equally, the positions of different types will be random for the different bursts, since packets of different types are sent randomly in each round. However, if some traffic types have low priority, they will always be in the first positions after the ordering is done because they suffer higher losses. At the end of this step, there will be b sets of types of traffic, ordered according to the loss rate observed for each burst.

In the third step (3), a statistical analysis is made to verify if there was prioritization of any specific type of traffic along the bursts. According to the authors, if POPI compares only two different types of traffic, it would be enough to just compare the measurements obtained for each type to determine if one had a different priority of the other. However, to compare more than two types of traffic it is necessary to group them according to the assigned priorities. In order to check whether the relative positions of k measurements are consistently repeated over b observations, the statistical *Problem of N Rankings* [73] can be applied. The solution adopted by POPI consists of computing the average position for each type of traffic over all bursts (*Average Normalized Ranks*) and group, employing a hierarchical divisive method, the traffic types whose averages

are similar. This process results in groups of types of traffic ordered according to their priorities. In a neutral network, this would result in a single group, since all types would have a similar average, i.e., the same priority.

To evaluate POPI, the authors first executed experiments using the NS-2 network simulator. In these simulations two pairs of end-hosts were used. One of these pairs was responsible for simulating background traffic, while the other pair simulated the execution of POPI. In the topology used in the simulations, the communication packets between the two pairs crosses the same two routers, which were responsible for simulating the prioritization of certain types of traffic, with a maximum bandwidth of 100 Mbps. The simulations were executed for $k = 32$ traffic types and $b = 32$ bursts. Incremental values were used for r – the number of rounds per burst. The upload rate for background traffic ranged from 10 to 90 Mbps. The results obtained in these simulations show that POPI was effective even in the presence of a large amount of background traffic: low priority packets were always dropped before those with higher priority. Another result was for the value of r: when $r < 18$ the measurement traffic was not able to cause congestion, thus no losses were observed making it impossible to infer anything. As the value of r increases, losses are observed more frequently for low priority traffic. Based on the results, the authors state that $r > 30$ is sufficient to obtain reliable results. Thus, $r = 40$ was used in the experiments executed in the PlanetLab, described below.

Experiments were conducted in the PlanetLab to evaluate POPI in a real environment and to find possible real cases of prioritization. In these experiments 162 nodes of the testbed were employed, spread around the world. POPI was executed on all pairs of nodes and in both directions for each pair. The values used for the variables were: $k = 26$, $b = 32$, $r = 40$, and $t = 10s$. The size of the packets generated was always equal to 1500 bytes, which generated an average bandwidth consumption of 1.04 Mbps. The results detected traffic prioritization for 15 node pairs. The authors also ran experiments using other metrics besides the loss rate. Unfortunately although these other metrics present lower overhead, they were not able to detect most of the prioritization cases that had been detected in the experiments based on the packet loss rate alone.

3.1.5 DiffProbe

The Differential Probing (DiffProbe) method [38] detects delay and loss differentiation. By using statistical methods, DiffProbe is able to detect that a non-neutral scheduler is being used, results are reported for both SP (Strict Priority) and Weighted Fair Queueing (WFQ) schedulers. Furthermore DiffProbe is also able to detect packet dropping policies, results are presented for WRED (Weighted Random Early Detection). According to the authors, the proposed method is a new class of network tomography [74]. DiffProbe assumes that an ISP classifies each packet as either high (H) or low (L) priority; low priority traffic suffers longer delays and higher losses according the scheduler and packet dropping policy adopted by the ISP. DiffProbe compares an application flow with a probing flow, both sent simultaneously. The main idea is that if one of the flows is treated differently, a difference of the performance will be observable if they are sent at the same time. DiffProbe measures the loss rate and the delays of two different flows sent simultaneously between a client and a measurement host.

The application flow is generated based on recorded real traffic, results are presented for two applications: Skype and Vonage. The application flow employs the same transport protocol, packet sizes, ports, payload and transmission intervals as the original traffic. The probing flow is used as a baseline for comparison. The authors claim that this

baseline traffic should be different enough from the application traffic, so they are not classified in the same way. However, the probing flow must have features in common with the application flow, such as packet sizes, so that performance results for both flows can be compared. The probing flow is generated as the application flow is sent through the network. In order to detect port or application differentiation, the probing flow packets have the same size of the last packet sent in the application flow, while the payload is random and the destination port is different. It is assumed that this destination port is not likely to be discriminated, i.e., it is classified as high priority by the ISP under test. If the discrimination is based on other features such as packet sizes, packet inter-arrival times, etc. then the rate of the probing flow must also be randomized.

At first, the two flows are sent at the same rate. After an interval, the sending rate of the probing flow is increased. The idea is to saturate the link with a larger amount of packets from the probing flow. DiffProbe never alters the sending rate of the application flow, since that might change the classification of that particular type of traffic. TD detection is made by comparing the measurement distributions of the two flows. DiffProbe employs the Kullback-Leibler (KL) divergence for delay distributions, and the two-proportion z-test (TPZ) for loss rate distributions.

Figure 3.6 shows how DiffProbe works in three steps. The first step (1) consists in generating the two flows, which are sent in the second step (2) simultaneously from a client to a measurement host, and afterwards in the opposite direction from the measurement host to the client. In the third step (3), the measured delays and loss rates from both flows are statistically analyzed in order to infer whether TD was employed or not.

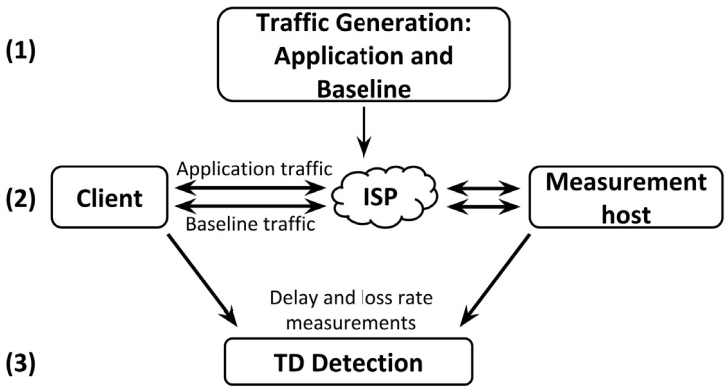


Figure 3.6: DiffProbe TD detection.

The authors evaluated DiffProbe using the NS-2 simulator and also emulation, with a client connected to a residential ISP and a server hosted at an university. TD was emulated by a router between the end-points. Both simulations and experiments in the emulated environment show that, whenever TD was observable and the amount of generated traffic saturated the link employed, the detection was accurate.

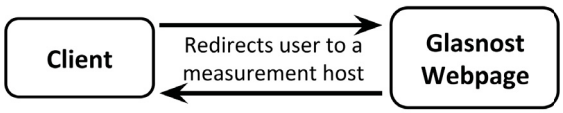
DiffProbe detects if a specific application traffic is being discriminated between two end-hosts. It assumes that the baseline traffic is not discriminated, which may lead to false-negatives if this is not true. Furthermore, DiffProbe requires path saturation, which always represents significant network overhead.

3.1.6 Glasnost

Glasnost [43] is a tool that allows Internet end-users to check if their ISPs are employing TD. It was designed as an easy-to-use tool that can be accessed via Web and requires no

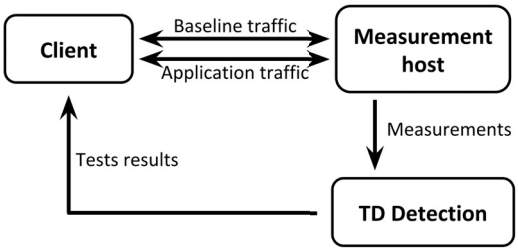
technical knowledge. Glasnost has already been used by thousands of Internet end-users around the world. It was initially designed for detecting TD on BitTorrent traffic, but can also be used to detect differentiation on any traffic of any application. The webpage was shut down in May 2017.

Figure 3.7 illustrates the usage of Glasnost. Initially, the end-user accesses the Glasnost webpage[2] and is redirected to a measurement server (Figure 3.7(a)). Users may be redirected to one of several different measurement servers, making it harder for ISPs to employ techniques against a specific server. The user then downloads the client application (Figure 3.7(b)) which is a Java applet that is executed at the end-user Web browser. The client connects to the measurement server and executes a series of tests (Figure 3.7(c)), after which the results are shown to the user.



(a) The Glasnost webpage redirects the user to a measurement host.

(b) The user downloads the client application (applet) from the measurement host.



(c) A series of tests are executed and results are returned to the user.

Figure 3.7: The usage of the Glasnost tool by an end-user.

Glasnost detects that the traffic from a given application is suffering differentiation by sending two flows in sequence, between a client and a measurement server. One flow corresponds to the target application, and the other is the baseline traffic generated for comparison purposes. The application traffic consists of messages of the real application. Glasnost assumes that an ISP identifies applications based on destination port or application protocol. The baseline flow is identical to the application flow in terms of the number of messages and message sizes, however the payload is different being generated randomly.

The measurement server computes the throughput for each flow. Each flow between the client application and the measurement server lasts several seconds, long

[2] `http://broadband.mpi-sws.org/transparency/glasnost.php`

enough for TCP to reach a stable state. The tests are repeated multiple times in order to reduce the noise in the measurements obtained. In the end, the measurement server processes the obtained data and displays the results on a webpage to the user. The computed metrics are the minimum, maximum, and median of the measured throughput.

The maximum throughputs observed for each flow are then compared to infer if the flows were treated differently. If the difference is higher than a threshold T, then Glasnost concludes that TD occurred. The authors claim that this threshold represents a trade-off between the ability of the system to detect TD and the generation of false-positives. For instance, if T is 50%, Glasnost will detect TD only if the maximum throughput achieved by one of the flows was half the maximum throughput of the other, possibly leading to false-negatives. On the other hand, if T is small, e.g., 5%, Glasnost can make mistakes and indicate TD when in reality the difference might have been caused by cross-traffic. The authors claim that 20% is a good value for T.

The authors report that in 2010 Glasnost detected that 10% of BitTorrent users suffered TD. Among the detected cases, differentiation occurred mostly on the upstream flow, with a few cases of TD on the downstream flow. One surprising result is that, after it was concluded that an ISP was practicing TD, only 21% of the ISP users were effectively affected (median). The authors list 3 possible explanations: (i) only users generating a large amount of traffic have been affected; (ii) only some parts of the ISP were affected; and (iii) TD was applied only during specific periods of the day, such as during peak times, for example. The authors also report that about 6% of users claimed that the tool did not detect TD that they believed to be suffering. One possible explanation for this is that the decision to minimize the number of false-positives increases the number of false-negatives.

Some of the Glasnost authors had developed earlier the BTTest tool [75] which clearly served as inspiration for Glasnost. BTTest detects if an ISP is blocking BitTorrent traffic. The operation of BTTest is very similar to that of Glasnost, except that BTTest only detects traffic blocking and only for BitTorrent. BTTest was available for a period of 17 weeks in 2008 and 2009, in which more than 47,300 end-users employed the tool around the world. The results obtained in this period show that in about 8% of the tests BitTorrent was blocked, mainly in the USA. In addition, the vast majority of blockings, about 99%, occurred on upstream data rather than downstream. Another tool, BonaFide [76] is based on Glasnost but focused on detecting TD in mobile networks. The tool was developed for the Android operating system and works in a way that is very similar to Glasnost, but with some modifications related to restrictions of mobile devices. A BonaFide client application running on a mobile device communicates with a measurement server that runs the tests. Each test consists of flows, as in Glasnost. BonaFide supports several application protocols, such as VoIP and BitTorrent, for example.

In synthesis, Glasnost is able to detect whether the throughput of certain types of traffic between two end-hosts are being limited when compared with baseline traffic. The measurement technique employed may result in false-positives, depending on the network load and cross-traffic, or in false-negatives, if the baseline traffic is classified in the same way as the differentiated traffic. Furthermore, it is important to note that the throughput cannot be used to assess the performance of applications that do not produce large amounts of traffic.

3.1.7 Packsen

Packsen [45] is a system that detects if an ISP is employing a traffic shaper to assign different priorities to different types of traffic. In addition to detecting the presence of TD,

the solution also infers which scheduler is being employed and its properties. The main idea is to compare the probability distributions of traffic features at the source and at the destination. If a significant difference is detected it may indicate the presence of traffic shaper between source and destination. Packsen is thus able to detect discrimination based on application protocol, port, time of the day, source, destination, among others.

The solution generates two different flows between two end-hosts, one flow is employed as a baseline for comparison, the other flow is from a specific application under test. A basic assumption is that the baseline flow is not suffering any type of discrimination. Packsen generates the two flows interleaved and with exactly the same bandwidth requirements. Packsen then makes measurements to infer the presence of a traffic shaper. Packsen measures the inter-arrival times of packets from both flows, as well as the bandwidth consumption. The main idea is that if there is a non-neutral traffic shaper in the path between the two end-hosts, the arrival of packets of a discriminated flow will present substantial differences from the way they were sent.

Three different statistical methods are employed. The methods are increasingly expensive in terms of computational power required. The first method employs short flows and presents low computational overhead and detects the presence of a shaper. This method compares the inter-arrival time distributions of the two flows, using the Mann-Whitney U-test [77] (MWU). In case TD is detected, the second method is used which infers which scheduler was employed, and with which parameters, such as the weight assigned to each flow for instance. This method compares the bandwidth required at the source with the bandwidth consumption measured at the destination. The second method is not robust in the presence of cross-traffic, especially when other applications are generating a large amount of traffic simultaneously with Packsen. The situation is particularly complex if the cross-traffic has an influence on the classification of the application flow and not on the baseline flow. A third method is then proposed, which is computationally more expensive than the others and consists of repeating the measurements several times until the results are reliable.

Packsen consists of three main types of components shown in Figure 3.8: a client, an experiment server and measurement hosts. The client connects to the experiment server and requests an experiment to be executed (1). The experiment server chooses one experiment and returns to the client. The client then chooses one available measurement host, informing the experiment which should be executed (2). The measurement host and the client run the experiment and collect data from the traffic generated. The data is then sent to the experiment server where it is stored for further analyses (3).

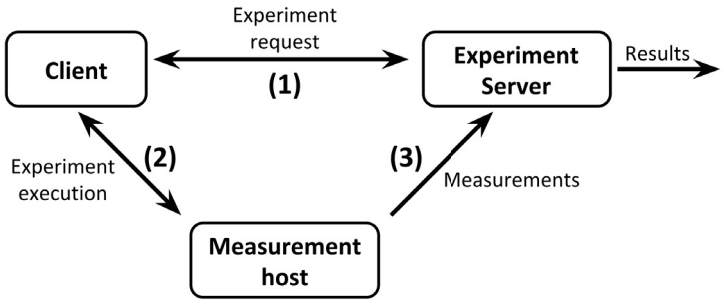


Figure 3.8: Packsen experiment execution.

The authors first evaluated Packsen in a controlled environment, a private testbed. This environment allowed the emulation of several types of traffic shapers, with different

parameters, as well as different combinations of cross-traffic. Experiments were also executed on PlanetLab on about 1000 hosts in order obtain results in a real large environment. The results obtained in the local testbed show that Packsen detected, with a low margin of error, both the occurrence of TD as well as the parameters used by the shapers, even in the presence of cross-traffic. Only a single false-negative was recorded in these experiments, in which there was TD but Packsen did not detect it. In the PlanetLab experiments TD was detected in only 0.7% of the tested host pairs (4 out of 518).

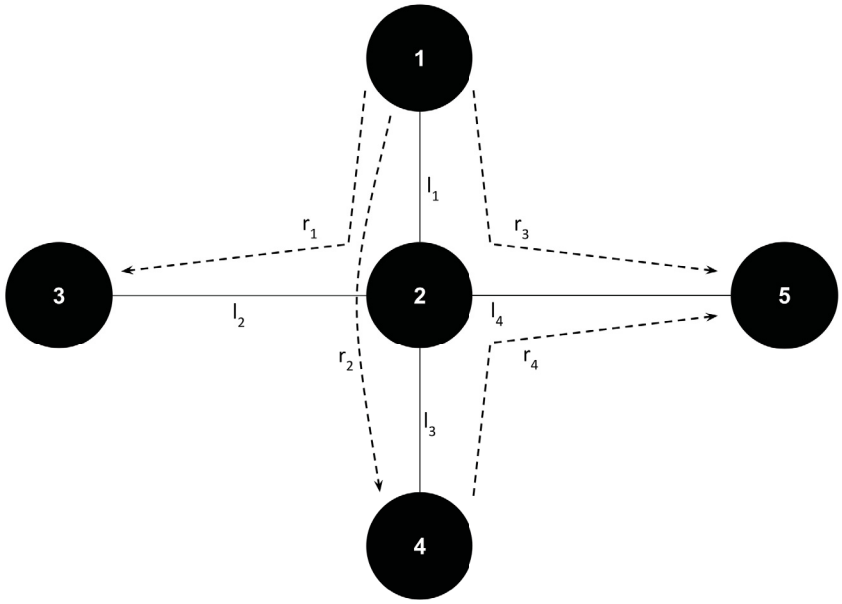
In synthesis, Packsen detects that an ISP is employing a non-neutral traffic shaper by comparing the performance of traffic from a given application with a baseline traffic that is assumed to be neutral. There is no mention of whether real traffic was employed or how synthetic traffic was generated, including features such as ports and payload. We also note that two assumptions are fundamental: the presence of a traffic shaper, and that the baseline traffic is neutral, the violation of any of these two assumptions may lead to inference errors.

3.1.8 Network Tomography Inference

An algorithm based on network tomography for inferring the presence of TD in a network is proposed in [46]. The algorithm is capable of identifying exactly on which link, or sequence of links, TD is occurring. The strategy is based on end-to-end measurements, i.e., it just uses external observations and does not need internal network measurements or information. The authors provide formal proofs indicating under which conditions the algorithm achieves these results. We call the proposed algorithm the tomography TD solution.

Network tomography [74] was originally proposed to allow the inference of network features such as the delay or loss rate of an internal link, only using end-to-end measurements. Network tomography usually combines multiple end-to-end measurements from different vantage points to infer properties of the network. The tomography TD solution builds a system of equations $y = Ax$ in which y is a vector containing the end-to-end measurements, A is the routing matrix that specifies the relationship between network links and end-to-end paths (i.e., it specifies which links are in each path), and x is a vector with the properties to be inferred for each link. An estimation for x is obtained by solving the system, or choosing one solution if there are multiple solutions (e.g., the solution with the highest probability). This tomography technique can only work with additive metrics, i.e., the sum of the measurements for each link of a path must be equal to the measurement obtained for the whole path (end-to-end). As examples, both delay and loss rate are additive metrics.

Figure 3.9 shows an example of the network tomography technique employed by the solution. Figure 3.9(a) shows a network with 5 hosts, with sequential identifiers from 1 to 5, interconnected by links $l_i, 1 \leq i \leq 4$. In the example four end-to-end measurements are executed over paths shown in the figure as $r_j, 1 \leq j \leq 4$. Figure 3.9(b) shows the routing matrix A for the measured paths. In this matrix, rows correspond to the paths and columns to the links. An entry of this matrix is set to 1 if the corresponding path traverses the corresponding link, and 0 otherwise. Figure 3.9(c) shows the resulting system $y = Ax$. In this system, $y = \{y_1, y_2, y_3, y_4\}$ and $x = \{x_1, x_2, x_3, x_4\}$, y_j corresponds to the measured value for path r_j and x_i is the metric to be estimated for link l_i. For instance, if link l_4 is non-neutral, there may be an inconsistency in the measurements corresponding to paths r_3 and r_4, since they share this link. In this case, the value of x_4 would be effectively different for each of the measurements, resulting in a inconsistent system that has no solution.



(a) Example of a network with 5 hosts and four end-to-end measurements.

	l_1	l_2	l_3	l_4
r_1	1	1	0	0
r_2	1	0	1	0
r_3	1	0	0	1
r_4	0	0	1	1

(b) Routing matrix A.

$$y_1 = x_1 + x_2$$
$$y_2 = x_1 + x_3$$
$$y_3 = x_1 + x_4$$
$$y_4 = x_3 + x_4$$

(c) The system of equations obtained from the four end-to-end measurements.

Figure 3.9: Network tomography technique employed by the solution.

This network tomography technique assumes that the network is neutral: all traffic from any path is treated equally on all links. In case this is not true, it becomes impossible to use the measurements obtained from different paths as a function of individual link metrics, and thus the resulting system of equations has no solution. Therefore, while conventional tomography techniques try to build solvable systems, the algorithm used in the tomography TD solution seeks unsolvable systems that reveal NN violations. Thus, the main idea behind the algorithm is that, when the network is not neutral, observations made from different vantage points will be inconsistent with each other.

The algorithm receives as input the topology of the network and a set of end-to-end measurements along with the corresponding paths on which the measurements were made. The output is a set of non-neutral link sequences, i.e., on which links, or sequence of links, TD occurred. The end-to-end measurements may use different types of traffic with the same source/destination, or the same type of traffic with different source/destination pairs, making it possible to identify different TD triggers. It is thus possible to detect TD based not only on content but also on source/destination.

As mentioned above, the algorithm searches for link sequences that result in an unsolvable system. For each sequence of links that are in more than one path, the algorithm builds a system using all the measurements that traverse that sequence and checks if it has a solution. If the system does not have a solution, the sequence of links is non-neutral. If the system has a solution, the link sequence is neutral or TD is not observable (i.e., it is a false-negative). In other words, the algorithm confronts measurements executed

on paths that traverse the same part of the network, trying to find inconsistencies that may be caused by TD. The authors claim that this algorithm generates no false-positives, since measurements executed on paths with only neutral links will always result in a solvable system. The authors also claim that the solution generates a small number of false-negatives, in which the algorithm mistakes as neutral link sequences that in reality are not neutral.

To evaluate the algorithm, two series of experiments based on emulation were carried out. The first experiment considered a topology with a single non-neutral link. In this experiment, all measurements were executed through this link. Different scenarios were tested, varying the behavior of the non-neutral link. In all cases the algorithm succeeded in detecting that the link was not neutral. In the second series of experiments a topology with several non-neutral links was used. Each of these links presented a different behavior. As in the first experiment, the algorithm always correctly detected the non-neutral links.

The authors also discuss the challenges to implement this tomography TD solution in a real environment. The most feasible option, according to the authors, is to use several end-hosts that periodically make end-to-end measurements of the paths between them and send the obtained measurements to a central server. It is also necessary to discover the topology of the network under analysis. Furthermore, another challenge is to collect measurements from a large enough number of different vantage points.

3.1.9 ChkDiff

ChkDiff [3, 78] is a tool for TD detection on ISPs that serve the domestic market (Tier 3). The tool first captures user traffic from a normal session and then replays a version of that traffic in the user ISP. ChkDiff measures packet loss and delays. The tool is able not only to detect TD but also to identify at which router TD occurred. The authors state that the strategy for measuring and detecting TD is independent of specific applications and the TD mechanisms employed by the ISP. Whatever the discriminated applications or techniques employed, TD typically will result in longer packet delays and more losses for the end-user.

The user traffic captured by ChkDiff is called a trace. This trace consists of traffic generated by a set of applications being executed by the user. The captured trace is used with minimal changes, thus the traffic shapers that the trace traverses should have the same behavior as if the packets were being generated by the user running the same applications. Only two modifications are made. The first is in the TTL field, so that packets only reach some desired hop. The second is that all packets have the same size, thus avoiding different transmission times.

ChkDiff takes its measurements by reproducing the captured trace several times. The source can be any end-host. The TTL is progressively incremented so that at each time the trace is transmitted it reaches one more router. When a packet arrives at the final router which is reached when the TTL field gets to zero, the router sends an ICMP Time Exceeded message back to the source host. ChkDiff measures the packet delay and losses with respect to these ICMP responses: the delay is the RTT measured from the time the original packet is sent to the time at which the ICMP message arrives. A packet loss corresponds to an ICMP message that is not received. Thus ChkDiff evaluates routers that are close to the user seeking for router behavior that identifies that TD has occurred. ChkDiff assumes that there is a non-neutral scheduler just before that router.

ChkDiff performs a statistical analysis to infer whether the traffic has suffered TD or not. The tool compares the delay and loss rate measured for a particular router

with the same metrics employed for the rest of the traffic. If measurements obtained for some particular type of traffic are significantly greater than those obtained for the rest of the traffic, then ChkDiff concludes that the router has applied TD to that traffic. Thus, the baseline used by ChkDiff is the whole traffic: NR states that a non-discriminated flow is treated in the same way as all other traffic, i.e., the measurements obtained for some traffic that is discriminated will stand out in relation to the rest of the traffic. In a simplified example, if the packet loss measured for some type of traffic is around 50%, while the loss measured for the rest of the traffic is around 10%, it is possible to conclude that the ISP is employing TD.

ChkDiff works in 4 steps as shown in Figure 3.10. In the first step (1) real user traffic is captured resulting in a trace. In the second step (2) the trace is preprocessed, generating a set of modified traces. In the third step (3), the set of modified traces is replayed and measurements are obtained. In the fourth step (4) the statistical analysis is performed to infer whether any traffic was discriminated and to locate where it was discriminated. The four steps are described in more detail below.

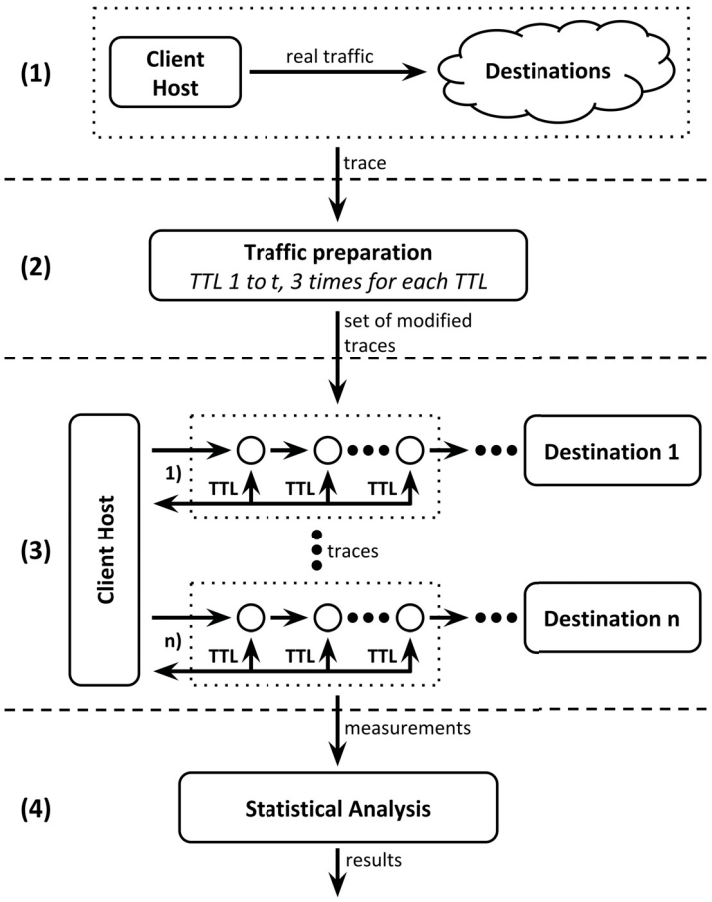


Figure 3.10: ChkDiff TD detection.

In the first step, the tool captures real user traffic from an end-host during a normal session. As ChkDiff employs measures on the upstream traffic, it favors data-intensive applications such as file sharing, VoIP, and instant messaging.

In the second step, ChkDiff processes the captured trace. This preprocessor generates a set of traces that will be replayed in the next step. The trace is separated into flows, grouping packets according to 5 items: source and destination address, source and destination port, and transport protocol. All packets have the same size, in order to

avoid different transmission times, which could result in false-positives, as delays must be comparable. Several new traces are then generated with packets of the same size separated in multiple flows. In each of these new traces, the packets are reordered and the TTL field is set accordingly; the values range from 1 to t and 3 traces are created for each TTL value. Thus, a set of $3t$ traces is generated, containing the same packets but in different orders and with different TTL values. The authors claim that using t equal to 3 or 4 is enough to traverse a typical tier-3 the ISP.

Packets are randomly reordered in each trace created in step 2, but keeping the global order of packets of a given flow. Reordering is necessary to ensure that all flows are treated under the same network conditions. According to the authors, this technique is also useful for minimizing problems such as background traffic and limitations on the maximum rate of ICMP responses that routers employ. At the end of this step the set of modified traces is ready to be reproduced.

In the third step, measurements are taken as traces are sent. Let h be the TTL of the packets of one of these traces. Each packet is sent to the original destination address and port. When one of the packets reaches the h-th hop, the corresponding router sends back an ICMP message to the source host. As mentioned above, delays and losses are measured based on these ICMP messages.

The fourth step consists of the statistical analysis to infer if any flow was discriminated and to identify where discrimination took place. ChkDiff only uses flows for which at least 20 received ICMP responses are received. As described previously, 3 traces are generated for each TTL value. As confirmed in the experiments which are briefly described below, the authors conclude that using 3 traces helps decrease the number of false positives. The idea is that if a given flow fails in the statistical test three times for the 3 traces, ChkDiff can conclude that the flow suffered discrimination. For the delay metric, ChkDiff compares the delay distribution of each flow with the delay distribution of the rest of the trace. ChkDiff checks the delay distributions with the Kolmogorov-Smirnov test. In a neutral network, this test is expected to indicate that the two distributions are equal. Thus, if a flow suffered delays greater than the rest of the trace, the test for this flow has failed. With regard to the packet losses, ChkDiff checks whether the packet loss for each flow is significantly different from the packet loss of the rest of the trace. ChkDiff employs a probabilistic test inspired by a binomial distribution. If a flow presented greater losses than expected, the probabilistic test for this flow fails and the hypothesis is false. When TD is detected at some hop h, it is observable for all hops after h; ChkDiff assumes that the shaper is placed between hops $h-1$ and h.

ChkDiff was first evaluated running in a neutral environment, with no TD, and later in a non-neutral environment. In both cases user traffic was captured during 3-minute sessions. Three types of applications were executed: images were uploaded to a social network; news webpages were accessed using a Web browser, and messages were sent using chat applications.

In the first set of experiments, executed on the neutral environment, ChkDiff was executed 100 times in a network in which the second hop router did not discriminate any traffic; however when a single trace was sent for each TTL value, about 30% of the executions presented 1 to 3 false-positives. When the experiment was executed with two traces for each TTL value: there was no false-positive. Based on these results, the authors generated 3 traces for each TTL value as mentioned above.

The second set of experiments was executed on a non-neutral environment, and initially only one type of flow was meant to be discriminated. Subsequently multiple

discriminated flows were used, with different fractions of the trace containing different discriminated flows. The source host was connected to a middlebox implementing a traffic shaper with the Dummynet [79] tool. The middlebox was then connected to a router at which the TTL of the packets expired. TD was implemented in two different ways: by limiting the bandwidth of selected flows and by discarding packets from the selected flows.

In the experiments with only one discriminated flow, ChkDiff was able to detect 100% of flows which suffered bandwidth limitations. When TD was based on packet dropping, some false-negatives were observed (discrimination occurred but was not detected). In the experiments with multiple discriminated flows, ChkDiff's statistical analysis stopped working correctly when the fraction of discriminated flows increased to 80% or more. ChkDiff was also evaluated and presented good results when the rate of ICMP responses from the router was limited.

In synthesis, ChkDiff detects TD in the first few hops from an end-host, and the purpose is to check residential ISPs. ChkDiff reproduces real traffic generated by end-users and checks if any type of traffic was treated differently from the rest. The solution relies on TTL-based probes to control up to which router the replayed traffic reaches – this does not always work, since the ICMP protocol is not guaranteed to be supported. Furthermore, ChkDiff modifies the captured traffic by using constant packet sizes and by shuffling the traffic - note that this shuffling may change the interval between the instants at which two packets are sent. These practices may change depending on how an ISP classifies the traffic, and this may lead to inference errors.

3.1.10 Wehe

A solution for detecting TD in mobile networks is presented in [44]. The goal is to measure whether an arbitrary application running on end-user devices such as smartphones and tablets is suffering TD. The main idea is to first capture the application traffic and then replay it twice: once using a VPN (encrypted tunnel), and once using a conventional non-encrypted channel. A statistical analysis is then performed on the measurements obtained in order to infer if there was TD. The metrics are the throughput, loss rate, and delay. The solution is named Wehe and is available in [80].

The authors assume that TD is performed by a traffic shaping middlebox which is in the network of the end-user ISP. Two commercial traffic shapers were used for validating the solution. Wehe does not classify as TD situations in which the rate enforced by these devices is equal to or higher than the rate at which the traffic is generated. The authors also assume that traffic classification can be based on header, payload or traffic behavior. Furthermore, as the solution employs a VPN to reproduce a previously captured trace, it also assumes that VPN traffic is not discriminated.

Figure 3.11 illustrates the three steps of the Wehe solution. In the first step (1), a VPN server captures real traffic of a mobile application while it communicates with the application server through a VPN encrypted tunnel. The captured trace is then replayed twice in the second step (2), this time the trace is sent from a replay application to a measurement host both using a VPN (encrypted IPSec tunnel) and using a conventional non-encrypted channel. The measurement host obtains information about the throughput, loss rate and delay from the trace replays. TD detection is performed in the third step (3), based on the collected measurements. The solution employs a statistical test based on KS in order to compare the different distributions and infer the presence of TD.

Wehe was first evaluated on a local testbed, using two commercial traffic shapers. The solution was also evaluated in the wild, through a mobile application made available

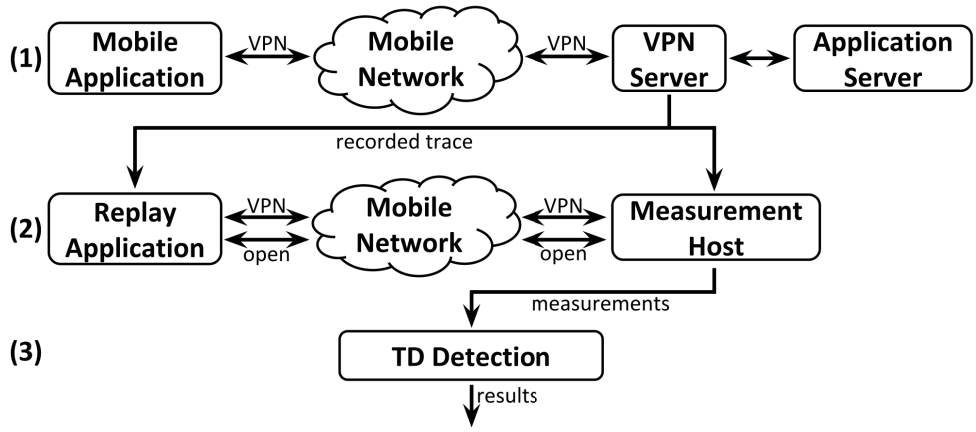


Figure 3.11: Wehe TD detection.

to the public. On the local testbed, the solution presented good accuracy. The results collected in the wild show that three mobile networks in the USA employed TD for services such as YouTube, Netflix, and Spotify.

According to the authors, by using a VPN to capture traffic, they design an application for recording traffic from any mobile application without the need for special permissions or modifications in the operating system, since traffic is captured by the VPN server that intermediates the communication between the end-user device and the application server. However, there are some potential limitations to the proposed solution. Detecting TD only when the actual rate is lower than the sending rate of the application may lead to false-negatives, specially considering that TD may only take place under congestion, which is not induced by the solution. Moreover, Wehe was designed and validated assuming that TD is implemented by ISPs using traffic shaping middleboxes, which may also lead to false-negatives, since there is several ways to implement TD. Furthermore, cross-traffic may impact both recording and replaying, and thus should be taken into account. The detection may be also hindered if VPN traffic is discriminated by the ISP.

In a more recent work [81] the authors report results based on measurements obtained during one year from end-users that used the solution on their mobile devices. 1,045,413 measurements were obtained, from 126,249 users connected to 2735 different ISPs in 183 countries/regions. From the obtained data, the authors were able to perform a large-scale study of TD practices in the Internet. Results show that TD occurred in 30 ISPs, located in 7 different countries. The majority of TD cases detected affected video streaming services.

3.1.11 NeutMon

NeutMon [48, 52] is solution for detecting differentiation of BitTorrent traffic between two end-hosts connected to mobile networks. It is able to detect TD based on throttling or routing, i.e., when the throughput of BitTorrent traffic is limited or when the BitTorrent traffic is forwarded to a different route. The main idea of NeutMon is to generate both BitTorrent traffic and a baseline traffic between a client and a server, and then compare the measurements obtained for each type of traffic. The BitTorrent traffic is generated based on the BitTorrent protocol specification. The baseline traffic is completely random,

but follows the same pattern of the BitTorrent traffic in terms of packets size and sequence. Both types of traffic are implemented on top of TCP.

In order to detect the two types of TD (throttling and routing), NeutMon performs two different tests between a client and a server: a speed test, and a traceroute test. The speed test measures the throughput experienced both by the BitTorrent traffic and the baseline traffic. The *traceroute* test discovers the path traversed by each type of traffic. Both tests are executed in the two directions – upstream and downstream.

NeutMon operates in two main steps, shown in Figure 3.12. The first step consists of executing the two tests (speed and *traceroute*), which exchange the two types of traffic (BitTorrent and baseline) between a client and a server. The client runs on an end-host connected to a mobile ISP, while the server runs on a dedicated machine. In the second step, the throughput and path measurements obtained during the tests are analyzed in order to detect TD.

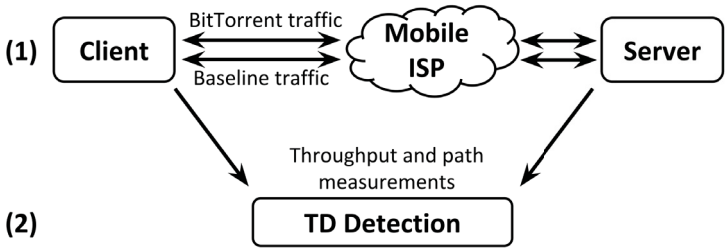


Figure 3.12: NeutMon TD detection.

The speed and *traceroute* tests are executed first employing the BitTorrent traffic. The BitTorrent traffic is transmitted for 10 seconds, and throughput measurements are obtained. Then, using the same TCP connection, the BitTorrent traffic is transmitted again, but each packet is sent multiples times, incrementing the value of the TTL field. The idea is to identify the path that each type of traffic traversed, in the same way as NetPolice (described in Subsection 3.1.2), ChkDiff (described in Subsection 3.1.9), and the *traceroute* tool. The same TCP connection is used for both tests in an attempt to cause the ISP to classify the traffic of both tests the same. After executing the tests for the BitTorrent traffic, NeutMon then executes the same two tests again, but employing the baseline traffic.

In order to detect TD based on throttling, the authors manually compare the distributions and averages of the throughputs measured for each type of traffic. In order to detect TD based on path, the authors aggregate several results of the traceroute test and check if there was a significant difference in the paths for one type of traffic when compared to the other. The rationale is that small differences in the paths may happen due to other reasons other than TD, such as load balancing, for example [53].

In order to evaluate NeutMon, the authors first executed the solution on a controlled environment. Next, they performed several experiments in the MONROE platform [82]. MONROE is a testbed containing hundreds of end-hosts connected to several different mobile ISPs, located across four different countries (Italy, Norway, Spain and Sweden). In the experiments, NeutMon clients were executed on several MONROE hosts, which communicated with a single server running on a dedicated machine.

The results show that there were throughput differences for some ISPs: at least one major mobile ISP in three of the four countries employed BitTorrent throttling. Regarding TD based on path, results show that there were differences in the path traversed by each type of traffic in some ISPs, but in these cases no throughput degradation was observed.

NeutMon detects differentiation of BitTorrent traffic on mobile devices, based on throttling and path. The solutions assumes that the baseline traffic is not discriminated, which may lead to inference errors if this assumption is not true. Furthermore, the *traceroute* test relies on ICMP responses in order to acquire the path traversed by the traffic. However, some ISPs limit the rate in which these responses are sent, and some routers simply do not issue them.

3.1.12 Other Solutions for Mobile Networks

In addition to the solutions presented in Subsections 3.1.10 (Wehe) and 3.1.11 (NeutMon), there are other works for monitoring NN in mobile networks. BonaFide [76] is an adaptation of Glasnost [43], described previously in this section, focused on mobile networks. BonaFide is an Android application that detects TD in a mobile network in the same way as Glasnost does for the traditional Internet. BonaFide can be seen a tailored version of Glasnost that complies to the restrictions of mobile devices.

WindRider [83] is a mobile application for detecting NN violations in a mobile network. It performs active and passive measurements. Active measurements are made using the Measurement Lab (MLab) [84] platform. Several applications using different ports are generated between a mobile device and a MLab server, in order to check if any portion of the traffic is being treated differently. Passive measurements are taken directly from the mobile device. The application collects the delays experienced by different webpages, and the explicit feedback from the end-users regarding different applications.

Furthermore, the authors of [85] advocate the creation of a "citizen observatory" of NN for mobile networks, by employing crowdsensing-based measurements and the open data paradigm. The authors claim that using a crowdsensing approach for making measurements on a mobile network can take advantage of the increasing number of smartphones, tablets and other mobile devices. Furthermore, making all measurements publicly available (an open data approach) would allow the creation of a "citizen observatory", containing NN-related information regarding different ISPs, thus increasing the transparency of mobile networks.

3.1.13 Other Related Works

In this subsection we describe other works that are related to NN monitoring, but not necessarily to the detection of TD. Some of these works address other practices that may also be considered as violations of NN, or are used just to measure network performance, not to infer TD itself. We compiled these tools in four categories, described below: QoS under-provision, censorship, content modification, and network performance measurement platforms and techniques.

QoS under-provision: Laws and regulations of several countries state that a violation of NN occurs when the Quality-of-Service provided by an ISP is lower than that contracted by the user. In this way, ISPs must deliver exactly the Quality of Service (QoS) specified in the contract. There are several solutions for monitoring the delivered QoS given the corresponding Service-Level Agreements (SLA) [86–102]. Some of these solutions, namely HAKOMetar and Adkintun described below, were developed due to the interest of governments in ensuring the compliance of networks with NN-related regulations.

HAKOMetar [86] is a tool that allows an end-user to check the QoS delivered by his/her ISP. The tool was developed by HAKOM, the regulatory agency of telecom-

munications in Croatia. The goal of the agency was to employ HAKOMetar to increase the transparency and competition in the broadband market. The tool was created based on previous results regarding traffic management practices, obtained from experiments conducted in Croatia [103]. The tool relies on active measurements of bandwidth, between an end-host and measurement hosts, to infer if the end-user is receiving the same bandwidth as announced by the ISP. According to the authors, the results confirm that HAKOMetar effectively increased the transparency in the Croatian broadband market, since consumers were able to check if their ISPs were really delivering the contracted bandwidth.

Adkintun [95, 96] is a solution for monitoring the QoS offered by ISPs in Chile. The solution was developed by NIC Chile Research Labs by request of SUBTEL, the regulatory agency for telecommunications in that country, with the purpose to monitor the compliance of ISPs with the Chilean NN law. Adkintun may be installed in end-users' devices or embedded in residential routers, provided to selected consumers. The tool periodically performs active measurements between end-hosts and several measurement hosts distributed across the country. Several metrics are employed, such as throughput, delay, and loss rate. All results obtained by Adkintun are publicly available through a website. The authors claim that Adkintun is helping consumers to protect their rights; the tool has been used as basis for complaints and even legal processes involving ISPs and SUBTEL. A similar tool, Adkintun Mobile [104, 105], was also developed to monitor the QoS of the mobile networks in Chile. This tool employs a combination of passive and active measurements obtained from mobile devices.

Censorship: The freedom of choice of end-users regarding the content they wish to access is also part of the worldwide NN debate. There are several solutions with the purpose to detect censorship in the Internet [106–109]. These solutions periodically perform measurements, creating a "census" of topics, services and websites that are blocked and/or filtered. A comprehensive survey on censorship detection in the Internet has been recently published [110].

Content modification: The modification of content generated either by users or providers can be employed to discriminate against unwanted traffic or to obtain advantages. Examples include: modifying the content of a website (such as by inserting advertisements); injecting forged packets into the communication of end-hosts; and modifying the content of packets (for corrupting BitTorrent data, for example). There are some solutions for detecting such practices. Switzerland [111] is a tool to detect the modification and injection of packets in the Internet. In [112], the authors present a solution for detecting modifications such as the injection of advertisements or malicious code in the pages of websites as they are being sent to the users.

NNSquad Network Measurement Agent (NNMA) [113] is a tool that monitors multiple metrics related to the network activities of a set of hosts. In the context of NN, the most relevant measurements made by NNMA are done to detect the injection of forged TCP reset (RST) packets. A RST packet terminates the connection between two end-hosts, thus ISPs may inject such packets in order to stop unwanted traffic [114], such as BitTorrent. While NNMA does not directly measure the impact of RST injection on different types of traffic, the technique could certainly be employed for TD detection.

Network measurement platforms and techniques: Network measurement platforms and services [84, 115–121] are used to acquire multiple types of measurements that can be

used to detect TD. These solutions continuously monitor several network properties possibly from several ISPs, also allowing for a comparison of different ISPs. A complete survey on Internet measurement platforms has been recently published [122]. Furthermore, several network measurement techniques [39, 65, 123–127] may also be employed for detecting TD, by comparing the measurements obtained for different types of traffic. We describe below four network measurement solutions which were designed specifically with NN issues in mind.

The Network Neutrality Bot (Neubot) [128, 129] is a software platform for continuously obtaining distributed measurements on the Internet. Neubot enables the implementation of solutions for verifying the QoS provided by ISPs based on the obtained measurements. Neubot performs several different measurements periodically on multiple end-hosts, and all data is public. Neubot measurements include different application protocols, such as HTTP, BitTorrent, RTP, and VoIP. Neubot does not implement TD detection, since it only collects measurements. Neubot has been running on the MLab platform since February 2012, making use of the several measurement hosts provided by MLab. The authors claim that the measurements collected by Neubot allow a systematic evaluation of the services provided by ISPs, which might contribute to the NN worldwide debate with real data.

Netalyzr [130] is a network measurement service, aimed at evaluating an end-user Internet connection, collecting data which may be further used for identifying NN violations. Netalyzr runs on end-user browsers, and makes measurements by communicating with several measurement hosts. The measurements include the usage of different protocols (such as TCP, UDP, HTTP, and DNS), the end-user local network (e.g., NAT) and ISP (such as IPv6 support, content modification, port filtering, bandwidth and delay). All measurements are publicly available, contributing to a deeper understanding of QoS and NN issues.

ISPANN [131] is a system for detecting NN violations from within an AS, according to the NN rules from the country where the AS is situated. They assume their solution runs inside an AS network, and that it has enough information about the network to communicate with its devices. The system checks if the AS complies with the NN rules in place, identifying where in the network NN violations occur. ISPANN addresses a different problem from this thesis: ISPANN audits a network from within it, instead of through end-to-end observations as in this thesis (as defined in Chapter 2).

In [132] the authors propose a framework for continuously monitoring several metrics related to NN. 17 different metrics are periodically collected from several client hosts connected to different ISPs, covering a wide range of protocols and network metrics. The authors do not, however, compare the obtained measurements in order to infer the presence of TD.

3.2 TAXONOMY

Given the fact that the multiple existing solutions for TD detection have been proposed independently and often using not only different approaches and features, but also different terms for the same concepts, objectives, and techniques employed, in this section we define a taxonomy with the purpose of unifying the description of the different types of TD and TD detection under a unifying framework.

The proposed taxonomy was built taking into account the existing solutions described in Section 3.1. The purpose is to have a common ground to understand the

differences and similarities between the solutions. In Section 3.3.2 we compare the existing solutions based on the taxonomy presented in the current section.

This section is organized in two subsections: in the first (3.2.1) we present a taxonomy of TD, in the next (3.2.2) a taxonomy of TD Detection. We make use of feature diagrams to present the taxonomy. Feature diagrams [133] are hierarchically arranged sets of features, with different types of relationship between features and sub-features – both optional and mandatory.

3.2.1 Traffic Differentiation: A Taxonomy

The feature diagram in Figure 3.13 presents a taxonomy for traffic differentiation. In the diagram, TD has four main features, which represent different aspects of TD: triggers, traffic classification, differentiation mechanisms, and perceived discrimination. The triggers are the conditions or characteristics of the traffic that may lead an ISP to employ TD. Traffic classification indicates which features are used by an ISP to classify the traffic. The differentiation mechanisms used by an ISP to implement TD are classified according to how they affect the traffic. Finally, the perceived discrimination describes how users or monitoring systems perceive TD. We further describe each feature and its subfeatures below.

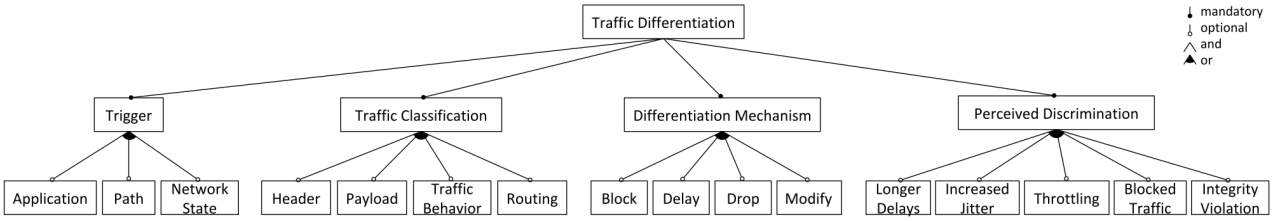


Figure 3.13: TD feature diagram.

Trigger: An ISP may start to discriminate traffic because of specific traffic properties or under certain conditions, or even because of a combination of properties and conditions. We call triggers those factors that lead an ISP to start TD. We compiled three types of triggers:

a) *Application*: TD can be triggered by an application, which is discriminated by the ISP. For instance, an ISP may avoid congestion by slowing down bandwidth-hungry applications (e.g., P2P and video streaming), or it may prioritize traffic from its own applications or from business partners.

b) *Path*: in TD triggered by path, all the traffic coming from, or going to specific end-hosts or traversing specific ASes may be discriminated. For instance, an ISP may prioritize traffic coming from a certain content provider or a neighbor AS due to commercial agreements (e.g., fast-lanes).

c) *Network state*: this TD trigger is employed depending on the state of the network. For instance, an ISP may employ TD only on links with high load, or at specific times of a day (e.g., peak hours).

For instance, if an ISP degrades all traffic from a specific application or with a specific destination address, we say that in this case TD is triggered by application and

path. The solutions presented in Section 3.1 are able to detect different combinations of these triggers. Furthermore, a solution may be agnostic to the trigger, i.e., it does not make any assumption regarding which triggers are employed, being able to detect TD regardless of the triggers.

Traffic Classification: Several different properties may be used to identify the triggers presented above and assess the priority of the corresponding traffic, as described previously in Chapter 2. We identified four categories of traffic classification:

a) *Header*: classification based on header information, e.g., source and destination addresses and/or ports, transport protocol used, application protocol used, type of service (TOS) required, among many others.

b) *Payload*: classification based on application data, either using information from the application PDU header (e.g., HTTP or BitTorrent headers), or based on application payload, which can be identified using DPI and pattern matching.

c) *Traffic behavior*: classification based on flow rate, flow duration, average packet size, inter packet interval, number of connections, total bandwidth.

d) *Routing*: classification based on source and/or destination end-hosts or networks, previous and next ASes.

For instance, an ISP may identify from which application some traffic corresponds to by checking packet headers – e.g., destination port. Some solutions for detecting TD, however, are agnostic to specific classification methods, i.e., they do not make assumptions regarding how ISPs classify traffic.

Differentiation Mechanism: There are several mechanisms that an ISP may employ to implement TD, as described previously in Chapter 2. Different mechanisms may affect the traffic in different ways. We identified four categories for these TD mechanisms:

a) *Block*: blocking mechanisms interrupt all traffic by simply not forwarding packets, or by injecting connection termination messages (e.g., messages with the FIN or RST flags set in the TCP protocol).

b) *Delay*: delay mechanisms either increase or decrease the delay of packets. These mechanisms may, for instance, prioritize packets according to their type (e.g., traffic shaping) and/or forward packets through internal routes that are faster or slower.

c) *Drop*: drop mechanisms degrade the traffic by dropping packets according to some criteria (e.g., traffic policing).

d) *Modify*: modification mechanisms alter packets, header and/or payload. For instance, an ISP may reduce the TCP window size to force the sender to slow down, or even modify specific application protocol fields to manipulate the application behavior (such as in transparent proxies).

For instance, a traffic shaper may be employed by applying some non-neutral scheduler to the traffic, forwarding different types of traffic according to priorities. This is an example of a delay mechanism, since its main goal is to delay low-priority traffic,

prioritizing other traffic. Some solutions, however, make no assumptions regarding which TD mechanisms are employed.

Perceived Discrimination: TD is perceived by users and monitoring systems in several different ways. These features reflect how users perceive and report TD, and are often the basis of proposed regulations and compliance surveillance.

a) *Longer delays*: users perceive longer delays to receive data from the network.

b) *Increased jitter*: the variation of the delay is high enough to disrupt specific applications.

c) *Throttling*: monitoring systems can perceive TD as a reduction of the available bandwidth, however this is often perceived by users as longer delays or unresponsive services, which are common in the case of video streaming applications.

d) *Blocked traffic*: users do not receive any or part of the packets of some particular application.

e) *Integrity violation*: the received information has been modified in the network in an unauthorized way.

For instance, some tools measure TD from the point of view of end-users, reporting delays or jitter that are higher than expected, besides throttling (bandwidth reduction), non-authorized modifications, or even blocked traffic which in some cases have been reported to be nation-wide.

3.2.2 TD Detection: A Taxonomy

The feature diagram in Figure 3.14 presents a taxonomy for TD detection. We identified four main features: measurements, monitoring architecture, traffic, and inference mechanism. These features represent different aspects of strategies for detecting TD. In order to detect TD, *measurements* are made, performed by hosts organized in different *topologies*. The *traffic* employed in such measurements may also have different properties. The data obtained may be processed in different ways to *infer* the presence of TD. We further describe these features below.

Note that this taxonomy does not represent the way any particular solution was designed, nor how new solutions should be designed. It is meant to organize concepts and features to allow comparisons. We hope though that the taxonomy can be an useful framework to help creating new solutions.

Measurements: Since the internal properties of ASes are not known *a priori*, TD detection solutions rely on measurements that are made from outside the network in order to infer what is happening inside. These measurements are thus made from end-hosts and, in the context of the TD detection problem, have three fundamental characteristics:

a) *Metrics*: there are several possible metrics that may be employed to assess different types of traffic. Different TD mechanisms affect traffic in different ways, thus different metrics may also be employed. For instance, traffic policing may have a deeper impact on the loss rate than on delay, since it favors dropping instead of queuing packets in order to enforce a maximum rate. Traffic shaping would have

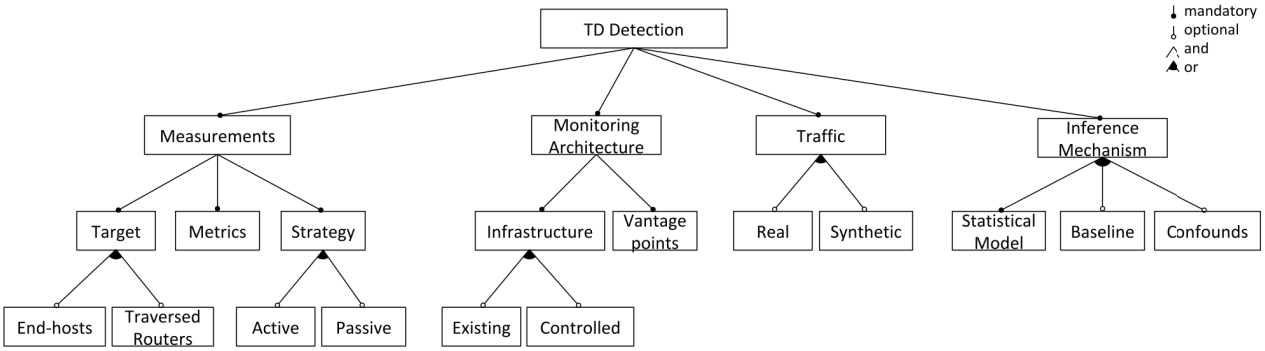


Figure 3.14: TD detection feature diagram.

the opposite impact. Throughput may be equally affected by shaping and policing, since both dropping and delaying will cause less packets to be transferred during a given interval of time. The most common metrics employed by the existing solutions presented in Section 3.1 are delay, loss rate and throughput, but other metrics are also used.

b) *Target*: measurements might be taken at *end-hosts*, not taking into consideration routers in the path; alternatively measurements can be taken relative to some router. For instance, a solution may measure the upload bandwidth of some end-host, and the download bandwidth at the other end, ignoring how the routers in the path interfered in the measurements. Moreover, a solution may measure the loss rate experienced by some traffic when traversing a specific router in an ISP network.

c) *Strategy*: measurements might be made following different approaches, such as active, passive, or hybrid. These different approaches are further described below in Subsection 3.3.1.

Monitoring Architecture: Measurements might be made using different topologies, and may require control of some (or all) of the hosts participating in the measurements. We defined two different aspects of a monitoring architecture:

a) *Infrastructure*: a monitoring solution may require *control* of one or several hosts in order to make measurements. It may also take advantage of a *pre-existent* infrastructure to make the measurements. For instance, a solution may explicitly run a measurement application on two end-hosts, which would require the installation of the application in the hosts, or it could make measurements based on some existing activity, for example of an existing P2P network, which would not require any modifications of the end-hosts.

b) *Vantage points*: measurements may be taken from different "points of view". For instance, a solution may require only measurements between a pair of end-hosts in order to infer TD, or it may require measurements from multiple pairs of end-hosts, aggregating and processing the collected data afterwards.

Traffic: Different types of traffic may be employed to make measurements and infer TD. Most existing solutions, described in Section 3.1, employ two or more types of traffic in

their measurements and check if there was any significant difference in the performance measured for each type. Traffic can be of two types:

a) *Real*: a TD detection solution may take advantage of already existing traffic. For instance, passive measurements only observe traffic, making measurements without introducing any new traffic in the network. Furthermore, a solution may record real traffic and use it later to perform measurements, instead of generating synthetic traffic.

b) *Synthetic*: a solution may generate traffic to execute measurements. For instance, some solutions generate different types of traffic by modifying some previously recorded real traffic, while others create traffic only by following the specifications of an application protocol.

Inference Mechanism: Different approaches are possible in order to detect TD from the obtained measurements. We identified three main features related to TD inference:

a) *Statistical model*: different statistical methods can be employed to analyze the collected data and make inferences regarding TD. Hypothesis tests may be employed to compare different sets of measurements.

b) *Baseline*: most solutions compare the measurements regarding different types of traffic with some baseline traffic, which is often assumed to be non-discriminated. This baseline for comparison may be obtained in several different ways. For instance, some solutions assume that a specific type of traffic of some application (e.g., HTTP) does not suffer TD, and thus can check how the traffic generated by other applications compare with the baseline.

c) *Confounds*: confounds are factors that may have an impact on the traffic or on how it is treated. Different solutions consider different confounds, and have different approaches to deal with them. For instance, some solutions repeat the same measurements multiple times in order to decrease noise (e.g., from cross-traffic).

3.3 CONSOLIDATION OF THE STATE OF THE ART

The purpose of this section is to consolidate the state of the art by presenting (i) a compilation of the techniques employed by the TD detection solutions described in Section 3.1; (ii) a comparison of the solutions; and (iii) a discussion of TD detection challenges given the state of the art.

3.3.1 Techniques Employed by the TD Detection Solutions

In this subsection we present a compilation of the different techniques employed by the solutions described in Section 3.1 to detect TD.

Passive/Active and Hybrid Measurements: As described in Chapter 2, different TD mechanisms affect traffic in different ways, such as increasing the delay or loss rate. Thus, existing solutions rely on network measurements to identify performance conditions that represent symptoms that part of the traffic is being treated differently from other parts. Network measurements may be active, passive, or hybrid.

Passive measurements consist of observing the real network traffic, without generating any new flows or packets. Measurements are usually made at both ends by evaluating performance characteristics for sending and receiving packets. For instance, a sniffer implements passive measurements, which captures and analyses all network traffic.

Active measurements generate traffic, i.e., probes or flows between one or more pairs of end-hosts. Measurements are then made to evaluate the performance of the probes, for example. For instance, the source host sends a file to an destination host, using the FTP protocol, then the performance is evaluated through metrics such as throughput, packets loss rate and delay, among others.

Hybrid measurements consist of any combination of active and passive approaches.

TTL-based Probes: Some solutions employ a technique based on the ICMP protocol in order to locate where TD occurred. This technique consists of sending measurement probes with a predefined TTL (Time To Live) value. The idea is to force the probe to reach up to a specific router between the source and the destination. The *traceroute* network tool employs this technique in order to obtain the exact path between two end-hosts.

In this technique, the TTL field of each probe is set to some value i, and is sent from one end-host to another. Unless it reaches its destination, the probe travels only up to the i-th router. Each router along the path between the end-hosts decreases the TTL field by one. When the TTL gets to 0, the probe is dropped, i.e., it is not forwarded to the next router. Furthermore, an ICMP *Time exceeded* message is sent back to the source host. The measurements are then made based on the ICMP responses. For instance, the loss rate is the ratio of ICMP responses not received, and the delay is the time interval from the instant the probe is sent to the time instant the ICMP response is received. This technique can thus be used to make measurements for each router along the path between a pair of end-hosts, by sending multiple probes with incremental TTL values.

Path Saturation: Depending on the traffic management policies adopted by an ISP, TD may be employed only when the network is under congestion. For instance, if an ISP employs traffic shaping for queuing and forwarding packets according to different priorities, different delays and/or loss rates will only be observed if the packets are effectively being queued. If the routers are able to forward packets fast enough, no queuing will effectively happen. Thus no packet will be delayed in favor of others, and since queues do not fill up, no packets will be selectively dropped.

Therefore, TD may only be observable when the network is under a very high load. Thus, active measurements often generate a large amount of traffic to saturate the path between two end-hosts, forcing TD to happen, e.g., forcing packets to be delayed and/or dropped, or bucket tokens to expire.

Client-Server Measurements: Several solutions make measurements based on traffic generated between two end-hosts, following a client-server model. These end-hosts are often called the *client host* and a server which is called the *measurement host*. On the one hand, this approach allows several different types of measurements, made both on the client host and on the measurement host, as well as total control of the traffic – such as what is sent, which protocols are employed, which responses are expected, etc. On the other hand, a problem with this technique is that it is not possible to evaluate other paths other than the path between the two end-hosts, neither does it allow the detection of TD triggered by path (such as coming from specific sources or destinations).

Measurements Involving Multiple Hosts: Some solutions make use of multiple hosts instead of a client-server pair. These solutions may make measurements from several hosts, and/or to multiple hosts or prefixes. The idea is usually to acquire data from multiple vantage points, and confront them in order to infer some property – which may indicate TD and where it took place.

For instance, some solutions control a large number of hosts, and generate traffic between them. Others send measurement probes from one or more hosts to multiple Internet destinations prefixes, in order to make the traffic traverse different paths, through different ISPs.

Another related technique is network tomography [74]. It consists of inferring properties of network internal links (such as delay or loss rate) only using end-to-end measurements, i.e., measurements for particular internal links are not available – in the TTL-based probes these measurements are available. Network tomography usually combines several end-to-end measurements, from different vantage points, to infer properties of a common core.

Traffic Recording: In order to make active measurements using real traffic, some solutions record the traffic in advance and replay it afterwards, as many times as needed. Traffic recording can be done for instance by capturing the traffic of some applications as it is being executed by some user in a real network [134]. Some solutions reproduce recorded traffic exactly as it was captured, while others make modifications before reproducing.

This technique allows the use of traffic from any arbitrary application or protocol, whether standard or not. However, traffic recording often requires special permissions and anonymization.

Traffic Emulation: This technique consists in generating artificial traffic that mimics an application or protocol. In this way, measurements can be made for any type of traffic, varying features such as port number, application protocol, payload, sending rate, packet size, among others. Several existing solutions also employ this technique to create a baseline traffic, which is assumed to be non-discriminated – e.g., carrying randomized payload.

Traffic emulation may be performed based on previously captured real traffic, or can be artificially generated [135]. The latter requires the protocol and application behavior to be well known, and is also usually based on statistical models. Artificial traffic, however, may not be realistic enough, being treated by ISPs differently than the real traffic would be [44].

Traffic Shuffling: In the traffic shuffling technique, different flows are sent simultaneously, with their packets interleaved in random order. Some solutions shuffle the packets each time they are sent, in order to decrease the bias. For instance, if packets are sent always in the same order, they might get queued in the same way every time, and this may be misinterpreted as TD: buffers gets full and most dropped packets are of a single application, which may lead to the conclusion that the traffic is being discriminated, i.e., a false-positive. This can be avoided by sending the traffic multiple times, with packets in random order. Note, however, that the relative order of packets of an application should be kept the same, since changing their order would change the behavior of the application, which might affect the traffic classification.

Relative Discrimination: A common approach for inferring TD is to compare measurements taken for different types of traffic, in order to determine if they have been treated differently, i.e., if one traffic was prioritized or degraded in relation to the other. The rationale is that if no TD was employed, the distributions of measurements for two different types of traffic will be statistically similar, since both types of traffic were processed under the same network conditions. However, if one type is treated differently, the corresponding measured distributions will be significantly different: one type of traffic was discriminated relative to the other. Most solutions that employ this technique assume the existence of a baseline traffic that is non-discriminated traffic. The strategy then is to compare other types of traffic with the baseline traffic.

Hypothesis testing is frequently used in order to infer relative discrimination. For instance, the hypothesis may be that the two sets of measurements are drawn from the same distribution. If the test fails, then the hypothesis is false; in this case the measurements are significantly different, and thus TD is characterized. Examples of tests employed by the existing solutions include the Kolmogorov-Smirnov (KS) test [73], Kullback-Leibler test (KL), two-proportion z-test (TPZ), and Mann-Whitney U-test (MWU) [77].

Measurements Clustering: Some solutions try to cluster the obtained measurements in order to compare several different sets of measurements, instead of just two as in the relative discrimination technique described above. Measurements for different types of traffic, from different sources or even from different ISPs may be grouped together and compared afterwards.

Measurements may be clustered, for example, according to the corresponding confounds – such as grouping measurements from hosts that are geographically close to each other. Another example is clustering different types of traffic according to the measured performance, building a ranking based on performance classes – which should result in a single class in a neutral network.

3.3.2 A Comparison of the TD Detection Solutions

The solutions described in Section 3.1 differ mainly on how they make measurements and compare the obtained data, presenting different monitoring architectures, metrics, employing different traffic generation strategies, and statistical methods. Most solutions perform active measurements between one or more pairs of hosts – employing traffic corresponding to different applications – and compare the obtained measurements in order to detect significant variations. Other solutions perform measurements on routers along the path between one or more pairs of hosts, in order to identify exactly where TD happened. There are also solutions that employ passive measurements of the real traffic generated by different applications. Table 3.1 describes how each solution addresses each different aspect of TD detection defined previously in the taxonomy presented in Figure 3.14.

Each solution is designed with different goals and assumptions in mind. Glasnost, for example, targets end-users, being an easy-to-use online tool that requires no technical knowledge, while NetPolice targets backbone ISPs, and requires more technical knowledge to be deployed and run. The solutions presented in this section employ different sets of techniques to achieve their goals under the assumptions made. Table 3.2 shows which of the techniques described previously in Subsection 3.3.1 each solution employs. Note that some solutions are based on similar sets of techniques. However, even when they are similar, the same techniques may be implemented in different ways, achieving different results.

In Table 3.3, we map our TD taxonomy, as defined in Figure 3.13, to the existing solutions. The table shows which types of TD each solution is capable of detecting. The cells in black indicate that the solution is agnostic to a specific feature, i.e., it does not make any assumptions regarding that feature.

Finally, Table 3.4 summarizes how each solution was evaluated, the main results obtained, as well as the particularities of each solution for dealing with noise. Most solutions were evaluated on simulated and/or emulated environments, and some were also evaluated in the wild. Simulation and/or emulation is necessary to measure the accuracy of TD detection, since there is no prior knowledge of how real networks employ TD. In this regard, the authors of Wehe used two commercial traffic shapers on their emulated environment, establishing a ground to evaluate the accuracy of their approach. Moreover, the authors of POPI contacted several network operators, some of which (quite surprisingly) confirmed their findings, while Glasnost received feedback from end-users. However, even in these cases, it is not possible to assess the actual accuracy of the TD detection in the wild.

3.3.3 A Discussion on TD Detection Challenges

We discuss next some of the main challenges for detecting traffic differentiation. The solutions described in Section 3.1 address some of these challenges. We also identify open challenges and future work in Section 3.4.

Challenges of End-to-end Measurements: Internal properties of the ISP networks are not known *a priori*. The topology, scheduling algorithms employed, specific devices in the network and how they are configured are examples of information that is not publicly available. Therefore, that information cannot be used to check if a network is neutral. Furthermore, since it is not feasible to test all possible types of traffic, at most what can be done is to run some tests in order to try to find cases in which TD can be identified. TD detection solutions rely on end-to-end measurements, from which they infer whether TD is being used or not.

Some solutions, however, make assumptions regarding specific characteristics of the network, such as the presence of traffic shapers, or support for specific protocols. For instance, some measurement strategies rely on the ICMP protocol, which is not universally supported by routers on the Internet [3]. However, some routers limit the rate of ICMP responses.

Confounds: Several other factors besides TD may result in observable differences for different types of traffic – the so-called confounds [69]. Examples include different routes, cross-traffic, congestion, geographic location, time of day, software, hardware, and other characteristics of the network (e.g. signal quality in mobile networks).

Measurements obtained in different periods of the day should not be compared, since the performance of applications may vary depending on the time they are executed. Furthermore, comparing traffic between different pairs of end-hosts is not always possible, since different hosts and the routes between them may have completely different characteristics. An ISP may also employ routes with different characteristics for different types of traffic due to reasons other than TD, such as load balancing or peering agreements. Congestion may not affect different types of traffic in the same way, as it depends on characteristics such as packet sizes, protocols employed, frequency of communication, among others.

Table 3.1: How each solution addresses each aspect of TD detection.

		Gnutella RSP	NetPolice	NANO	POPI	DiffProbe	Glasnost	Packsen	Tomography	ChkDiff	Wehe	NeutMon
Measurements	**Metrics**	Connectivity	Loss rate	Depend on application	Loss rate	Delay and loss rate	Throughput	Inter-arrival times, sent and received bandwidth	Any additive metric	Delay and loss rate	Throughput, loss rate, and delay	Throughput and path
	Target: end-hosts	Measurement host	–	Multiple end-hosts	Pair of end-hosts	Pair of end-hosts	Pair of end-hosts	Pair of end-hosts	Multiple end-hosts	–	Pair of end-hosts	Pair of end-hosts
	Target: traversed routers	–	Ingress and egress points	–	–	–	–	–	–	First few hops	–	–
	Strategy	Hybrid	Active	Passive	Active	Active	Active	Active	Active	Active	Active	Active
Monitoring Architecture	**Vantage points**	Multiple end-hosts and a measurement host	Multiple end-hosts	Multiple end-hosts	A pair of end-hosts	A pair of end-hosts	Multiple end-hosts and measurement hosts	Multiple end-hosts and measurement hosts	Multiple end-hosts	From a single end-host	A pair of end-hosts	Multiple end-hosts and a server
	Existing Infrastructure	Gnutella P2P network	–	–	–	–	–	–	–	–	–	–
	Controlled Infrastructure	A superpeer and a measurement host	Multiple end-hosts in different networks	Multiple end-hosts in different networks	A pair of end-hosts	A pair of end-hosts	End-host, measurement hosts, and a web server	End-host, measurement hosts and an experiment server	Multiple end-hosts	A single end-host	An end-host, a VPN server and a measurement host	Multiple end-hosts and a server
Traffic	**Real**	Gnutella protocol	–	Existing real traffic on end-hosts	–	–	–	–	–	Record the end-user real traffic	Record an application real traffic	–
	Synthetic	Referrals	HTTP, BitTorrent, SMTP, PPLive and VoIP	–	Several bursts containing multiple types of traffic	Based on applications real traffic	Based on real traffic	Not specified	Not specified	Reproduce traffic with modifications	Reproduce traffic through different channels	Based on BitTorrent protocol specification
Inference Mechanism	**Statistical model**	Probabilistic model to infer if ports were blocked	Compare the performance of different applications with the performance of the baseline	Group measurements with similar confounds and compare them with the baseline	Rank the measurements for each burst and verify if any given type of traffic was consistently ranked higher than others	Compare the performance of an application with the performance of the baseline	Compare the performance of an application with the performance of the baseline	Compare the performance of an application with the performance of the baseline	Employ network tomography to combine measurements from different vantage points and find non-neutral links	Compare each type of traffic reproduced with all the other types	Compare the recorded traffic performance when encrypted (VPN) and non-encrypted (conventional open communication)	Compare the performance of an application with the performance of the baseline
	Baseline	–	Assume that HTTP traffic does not suffer TD	The baseline is the average performance across several ISPs	–	The baseline is generated based on the target application and is assumed to not be discriminated	The baseline is generated based on the target application and is assumed to not be discriminated	The baseline is assumed to not be discriminated	–	The baseline is the whole traffic, expect for the type of traffic being evaluated	The baseline is the encrypted traffic, reproduced in a VPN, which is assumed to not be discriminated	The baseline is assumed to not be discriminated
	Confounds	Gnutella clients might ignore the referrals	Compare only measurements relative to a same core (AS)	Require the identification of all relevant confounds (related to the hosts, network and time)	Send multiple bursts of packets in random order	Both flows have same properties, such as packets size and sending intervals	Employ different measurement hosts to avoid evasive measures from ISPs	Repeat the measurements several times; keeps a constant bandwidth for both flows	–	The size of all packets is standardized	–	–

Table 3.2: Techniques employed by each solution.

	Gnutella RSP	NetPolice	NANO	POPI	DiffProbe	Glasnost	Packsen	Tomography	ChkDiff	Wehe	NeutMon
Active Measurements	X	X		X	X	X	X	X	X	X	X
Passive Measurements	X		X								
TTL-based Probes		X							X		X
Path Saturation				X	X						
Client-Server	X			X	X	X	X		X	X	X
Multiple Hosts		X	X					X			X
Traffic Recording					X	X			X	X	
Traffic Emulation		X		X	X	X	X	X			X
Traffic Shuffling				X					X		
Relative Discrimination		X			X	X	X		X	X	X
Measurements Clustering			X	X				X			X

Table 3.3: Types of TD detected by each solution.

		Gnutella RSP	NetPolice	NANO	POPI	DiffProbe	Glasnost	Packsen	Tomography	ChkDiff	Wehe	NeutMon
Trigger	Application	X	X	X	X	X	X	X		X	X	X
	Path		X									X
	Network State											
Classification	Header	X	X		X	X	X	Not specified by the authors		X	X	X
	Payload		X		X	X	X			X	X	
	Traffic Behavior										X	X
	Routing		X							X		
Mechanism	Block	X										
	Delay				X	X	X	X			X	
	Drop				X	X	X	X			X	
	Modify											
Perceived	Longer Delays					X		X		X	X	
	Increased Jitter					X		X		X	X	
	Throttling		X		X	X	X	X		X	X	X
	Blocked Traffic	X										X
	Integrity Violation											

Table 3.4: Solutions: evaluation, results, and particularities for dealing with noise.

Solution	Evaluation	Results	Dealing with Noise
Gnutella RSP	During 2 months, approximately 150,000 referrals were generated for about 72,000 distinct Gnutella peers, which were distributed in approximately 31,000 different prefixes, which can be considered a significant sample from the Internet.	From the 31,000 prefixes, in 256 at least a port was blocked. The most frequently blocked port was 136. Some ISPs in Canada, the USA and Poland blocked Skype ports.	At least 50 referrals are necessary to infer that a port was blocked with a confidence level of 99.5%.
NetPolice	Experimental results were obtained in the PlanetLab. 18 ISPs distributed across 3 continents were evaluated over a period of 10 weeks.	4 ISPs performed TD on 4 applications and 10 ISPs performed TD based on the previous AS of the packets.	Reduce noise by limiting the CPU utilization of the prober, keeping a large probing interval, and limiting the size of the probe packets to 40 bytes.
NANO	The authors executed experiments using the PlanetLab and Emulab testbeds. Application servers ran on geographically distributed PlanetLab nodes, while the application clients ran on Emulab, allowing the emulation of different TD practices and confounds.	Experimental results show that NANO is able to detect TD in different ways and for different types of applications, provided that all confounds are known and measured.	If NANO does not take into account all confounds, the causal relationship between the ISP and TD can lead to mistakes, either false-negatives and false-positives.
POPI	The authors first executed simulations using NS-2, in which two pairs of end-hosts generated traffic: one executing POPI and the other creating background traffic. Afterwards, experiments were conducted in the PlanetLab, in which POPI was executed on 162 nodes.	The simulation results show that POPI was effective even in the presence of a large amount of background traffic. In PlanetLab, the results detected traffic prioritization for 15 node pairs.	Based on the simulations results, the authors state that performing more than 30 rounds ($r > 30$) in each burst of measurements is sufficient to obtain reliable results.
DiffProbe	The authors evaluated DiffProbe using the NS-2 simulator and also emulation, with a client connected to a residential ISP and a server hosted at an university.	Both simulations and experiments in the emulated environment show that, whenever TD was observable and the amount of generated traffic saturated the link employed, the detection was accurate.	DiffProbe never alters the sending rate of the application flow, since that might change the ISP classification of that particular type of traffic.
Glasnost	Glasnost was used by thousands of Internet end-users around the world during several years.	The authors report that in 2010 Glasnost detected that 10% of BitTorrent users suffered TD, mostly on the upstream flow, with a few cases of TD on the downstream flow.	The authors claim that the threshold σ represents a trade-off between the ability of the system to detect TD and the generation of false-positives. According to the authors, 20% is a good value for σ.
Packsen	The authors first evaluated Packsen in a controlled environment, a private testbed. Experiments were then executed on PlanetLab on about 1000 hosts.	The results obtained in the local testbed show that Packsen detected, with a low margin of error, both the occurrence of TD as well as the parameters used by the shapers, even in the presence of cross-traffic. In the PlanetLab experiments TD was detected in only 0.7% of the tested host pairs (4 out of 518).	Packsen repeats the measurements several times until the results are reliable.
Tomography	Two series of experiments based on emulation were carried out, using different topologies and TD scenarios.	In both series of experiments, the algorithm always correctly detected the non-neutral links.	The authors claim that this algorithm generates no false-positives, since measurements executed on paths with only neutral links will always result in a solvable system, and only a small number of false-negatives.
ChkDiff	ChkDiff was first evaluated running in a neutral environment, with no TD, and later in a non-neutral environment, both emulated.	In the experiment in a neutral environment, ChkDiff had no false-positives after adjusting some parameters. In the non-neutral environment, some false-negatives were observed, and the statistical analysis stopped working correctly when the fraction of discriminated flows increased to 80% or more of the traffic.	All packets have the same size, in order to avoid different transmission times, which could result in false-positives, as delays must be comparable. Packets are randomly reordered on each probe, in order to ensure that all flows are treated under the same network conditions.
Wehe	The solution was first evaluated in a local testbed, using two commercial traffic shapers. The solution was also evaluated in the wild, through a mobile application made available to the public.	In the local testbed, the solution presented good accuracy. The results collected in the wild show that TD occurred in 30 ISPs, located in 7 different countries. The majority of TD cases detected affected video streaming services.	The two traffic traces compared (encrypted and non-encrypted) have the same characteristics of the original traffic, no modifications are made. The authors claim that this ensures that both traces will be treated in the same way as the original application.
NeutMon	The authors first evaluated NeutMon in a controlled environment. Experiments were then executed on the MONROE testbed.	Throttling was detected in some mobile ISPs. Furthermore, path differences were also found, but in these cases no throughput degradation was observed.	The baseline traffic has similar features than the application traffic. Furthermore, for detecting TD based on path, several path measurements are grouped in order to deal with path differences caused by other reasons, such as load balancing.

Therefore, robust statistical models are necessary for obtaining reliable results [37], avoiding false-negatives and false-positives. There is no automatic way for enumerating all confounds or to check if a set of confounds is enough. Depending on the approach adopted for detecting TD, it may be necessary to identify the confounds and collect data about them in addition to measuring the performance of applications.

Cross-traffic: Among the confounds described above, one of the most relevant which can have a deep impact on TD detection is cross-traffic. Traffic generated by other sources other than the TD detection solution may impact the measurements made, and thus how precisely TD can be inferred. Cross-traffic may be present in the same host and/or in the same local network. A large amount of cross-traffic may affect different types of traffic in different ways, this is particularly complicated when the cross-traffic presents a large variation over time.

TD Detection Evaluation: Validating a new solution in the wild may not be feasible, since there is no knowledge about the internals of the network between the end-hosts. It is important to define methods that avoid biased results. Most existing solutions rely on simulation and emulation in order to validate their strategies.

Simulated or emulated environments, however, do not have the same conditions found in a real environment. ISPs may classify and/or differentiate traffic in several different ways that may not have been covered by the simulation/emulation. Artificial traffic generated by the solutions may also be treated differently than real traffic would be. Solutions may experience thus more false-negatives and false-positives than expected when deployed on the wild.

3.4 OPEN CHALLENGES

We presented above in this chapter an overview of existing techniques and solutions for TD detection and a discussion of the challenges they face. However, we envision several other challenges which still need to be further investigated for designing effective solutions that can be considered to be future-proof. We list below some of this still open challenges we identified.

3.4.1 TD Location

As defined in Chapter 2, in addition to detecting the presence of TD, in this work we also address the problem of locating where in the network TD occurred. Determining that TD is taking place at a particular point of a network is not a trivial task, since there is no prior knowledge about the network internals. There are still only a few solutions for locating where in the network the source of TD is.

Some proposals for locating TD rely on path discovery techniques, such as the *traceroute* tool. Unfortunately, *traceroute*-like techniques may not succeed in obtaining the exact path between any given pair of end-hosts in the Internet, which may turn those proposals to locate TD ineffective. There is also another strategy that assumes prior knowledge of the exact topology for the whole network, which may not be feasible in the Internet. Therefore, more proposals are necessary to address the shortcomings of current solutions.

3.4.2 Emerging Technologies

There are several emerging technologies not yet explored by the current state of the art. Examples include 5G networks [136], Smart Cities [137], Future Internet [138], and the IoT [50]. These new technologies are expected to constitute a significant portion of the Internet in the future, both in terms of traffic and market share. As different confounds and constraints may apply to these emerging technologies, different techniques for monitoring NN violations might be necessary. For instance, path saturation may not be feasible in IoT devices, since they may not be able to transmit large amounts of traffic, due to energy consumption or low bandwidth, for example. Furthermore, 5G network *slices* are expected to support the specific QoS requirements of different applications, which currently is, by definition, a NN violation.

3.4.3 Measurements and Monitoring

Measurement techniques employed by the existing solutions present limitations. Active measurement strategies often require the path to be saturated first, resulting in high network overhead that may not represent the real conditions in which most applications run. Moreover, some techniques rely on TTL-based probes (which are not universally supported), or prior knowledge of the network topology to infer which ISP is employing TD. Furthermore, some existing solutions require control of a large number of end-hosts to monitor the network, which might not be a realistic assumption.

Another limitation refers to traffic recording techniques, by which previously captured traffic is replayed between two end-hosts. Some applications generate traffic between several pairs of nodes, and not just a single pair. Therefore, it might not be possible to properly mimic every application by reproducing its traffic only between a pair of end-hosts.

Further investigation on traffic monitoring and measurement techniques, including the metrics used, are also necessary to detect TD triggered by network state. Another related challenge is the detection of dynamic behaviors, such as for example when an ISP employs TD just on specific periods of the day, or when the ISP constantly changes the TD mechanisms over time. Moreover, most current solutions do not address traffic classification based on traffic behavior, or TD mechanisms based on traffic modification.

3.4.4 ISP Evasion

Most existing solutions generate their own traffic in order to make measurements and infer TD. However, the artificial traffic generated by such solutions might be identified by ISPs [139], which could then evade the TD inference, by prioritizing the measurement traffic, for example.

3.4.5 Solution Adoption

In order to achieve meaningful results, some solutions require that a large number of end-users report measurements for several different applications, and from multiple vantage points. Therefore, it is important to create incentives which may increase the adoption of the solution by a large number of users. Another challenge is to allow any arbitrary application to monitor how its traffic is performing compared to others, without having to implement TD detection on its own. This would enable not only end-users, but also applications and services to benefit from TD inference and to contribute to increase its

accuracy. Taking advantage of pre-existent infrastructures and/or real traffic monitored passively also allows measurements to be made without the need to control a large number of end-hosts or rely on a large number of end-users.

3.4.6 Network Programmability

The infrastructure of the Internet and TD mechanisms employed by ISPs are in constant evolution. As the authors of [44] describe, some commercial traffic shapers are able to identify a large number of applications, and classify the traffic based on payload and port. However, emerging technologies that allow network programmability, such as SDN [66], will increase the flexibility and enable different types of discriminatory behavior to be easily implemented in the network [140], thus the techniques employed will not be limited anymore by only what commercial shapers do. We can see that in the future it will not be easy to make assumptions on how TD is implemented. The design of TD detection solutions will have to make room for extensions on the fly, enabling them to keep up with network dynamics and evolution.

3.4.7 ISPs and Content Providers are Becoming Indistinguishable

A trend that is easy to see is the fact that commercial agreements between content providers and ISPs are becoming increasingly common and varied. Actually several ISPs are becoming content providers, while content providers are becoming ISPs. Examples of commercial agreements include the usage of Content Delivery Networks (CDN) [141, 142] and the adoption of zero-rating practices [143]. All these factors can result in content prioritization and discrimination and without a doubt pose new challenges on both the ability to detect TD and even more, on the very definition of what constitutes a NN violation.

3.4.8 Counteracting Discrimination

In addition to detecting TD, counteracting TD may enable applications and end-users to reduce any disruptive effects the discrimination may cause. Avoiding TD by redirecting the traffic to circumvent the discriminatory portion of a path are examples of strategies to counteract TD [144]. However, this is still an under-explored topic. New solutions that are feasible to be employed in real networks are necessary.

3.4.9 A Path May Traverse Countries With Different Regulations

A path between two end-hosts may traverse several different countries. For instance, a path from an end-host in Argentina to another end-host in Canada may traverse several ASes in other countries, such as Brazil and the USA. Regulations regarding NN may be significantly different in each country along the path. TD may be legal in one country, while illegal in another. Therefore, it is important when detecting NN violations to consider the regulations in place [145]. For instance, locating at least in which portion of a path the discrimination is occurring may help bringing transparency to traffic management practices on each country, and how regulations are influencing them. However, currently only a few proposals take into account the regulations in place when looking for NN violations.

3.4.10 Privacy

As described in Section 3.1, the existing solutions for detecting TD in the Internet gather several measurements from end-hosts. The data acquired by these solutions may reveal sensitive information about users [71]. However, the vast majority of TD detection proposals do not address the potential privacy issues that their measurement collection may create.

4 TRAFFIC DIFFERENTIATION ON THE INTERNET OF THINGS

The Internet of Things (IoT) is becoming increasingly present in modern life. It consists of a combination of numerous connected devices, sensors and actuators, that gather huge amounts of data and provide different services. Estimates show there will be about 212 billion IoT devices by 2020, and about 45% of Internet traffic will be related to IoT by 2022 [146]. These estimates indicate that IoT will constitute a significant portion of the Internet in the future, both in terms of traffic, and market share. The rapidly growth of IoT will most certainly have a high economic impact in several areas, providing device manufacturers, Internet Service Providers (ISPs), and application developers with new opportunities in the market [146].

There are currently several research projects covering key aspects of IoT [146], such as architectures, availability, reliability, mobility, performance, management, scalability, interoperability, security, and privacy. Innovation is thus essential in order to address the large set of challenges presented by the IoT. New devices, protocols, platforms, and cloud services are examples of different aspects that still need innovative solutions if the IoT is to achieve its full potential, contributing to quality of life and economy growth.

However, innovation, and thus the success of IoT, may be hindered by unfair traffic management practices from ISPs [51]. TD may impact selectively the quality of experience (QoE) of different IoT applications, resulting in a non-competitive market, since a difference in QoE may determine the success or failure of a device or application over competitors [147, 148]. For instance, if IoT sensors from one manufacturer have their traffic prioritized, the potential lower loss rates experienced by these sensors might cause much less packet retransmissions. This scenario could result in lower energy consumption, giving the manufacturer a competitive advantage.

In this chapter, we aim at evaluating how TD may impact the traffic generated by different types of IoT devices. The goal is to check if even a small prioritization of IoT traffic may result in a significant difference on the QoE perceived by end-users. We first discuss how TD may be implemented by ISPs to discriminate IoT devices, applications, and services. We then present common traffic patterns generated by IoT applications and how they might be affected by TD. Simulation results of each traffic pattern under different TD scenarios are then presented. An earlier version of this chapter was published in [50].

The rest of this chapter is organized as follows. In Section 4.1 we discuss how ISPs may discriminate IoT traffic. Traffic patterns generated by IoT applications and how TD may impact these patterns are then described in Section 4.2. We describe our simulations in Section 4.3, and discuss the results in Section 4.4.

4.1 IMPLEMENTING TD ON IOT

IoT devices usually generate small amounts of traffic [149], but the aggregate traffic from billions of such devices may motivate ISPs to throttle IoT traffic and/or charge for prioritization in the future. TD may take place at different levels of the IoT architecture, affecting different components as they interact with each other. These components connect to the Internet through several different access networks. Therefore, in this work we make

no distinction between wired and wireless networks, since TD may be employed on both, equally affecting IoT traffic.

An IoT system is a combination of connected components with different capabilities and purposes, from low-capacity sensors to high-performance cloud servers. In general, a plethora of sensing devices generate raw data, which is sent to the Cloud for processing and storage. Cloud services may also send data to the edge devices, such as commands and notifications. IoT platforms are often implemented as middleware [150] coordinating the interaction between the heterogeneous edge devices and cloud services. There could also be IoT gateways which aggregate data from several sensing devices, intermediating the communication between them and IoT platforms or other cloud services.

Figure 4.1 shows a common IoT architecture and shows where TD may take place. Any traffic that traverses the Internet is subject to TD: from/to edge devices or gateways, from/to IoT platforms, as well as from/to cloud services. An ISP may differentiate traffic involving specific device manufacturers (e.g., a brand of sensors or vehicles), applications (e.g., domain-specific or proprietary protocols), or origin/destinations (e.g., premium clients, cloud vendors, IoT platforms).

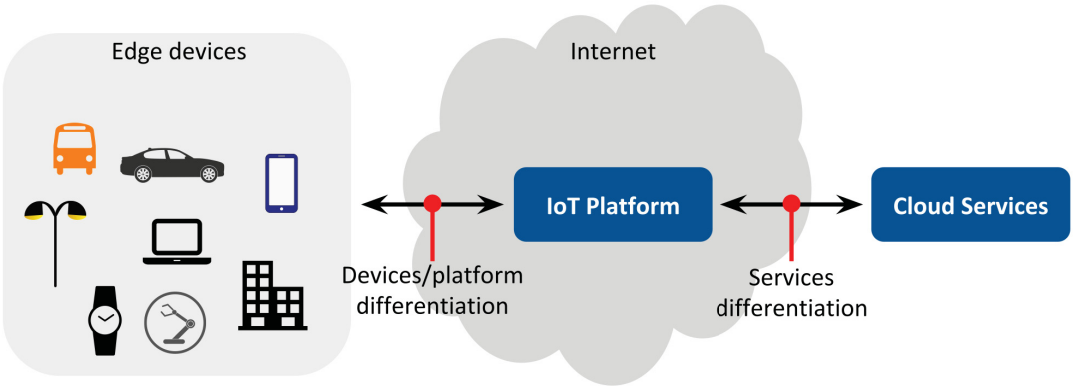


Figure 4.1: TD on a common IoT architecture.

4.2 IOT TRAFFIC PATTERNS AND THE IMPACT OF TD

Data traffic in the IoT is mostly comprised of the so-called Machine-Type Communication (MTC) [146] – also known as Machine-to-machine (M2M) communication. MTC is characterized by the communication of several devices among each other without the need of human interaction, as opposed to Human-type communication (HTC), which consists of the traditional Internet traffic. In the IoT, sensing devices, gateways, middleware, and cloud services communicate autonomously with each other. Human interaction is also present, such as command inputs (e.g., smart home), or during critical situations (e.g., decision making, security alarms).

MTC traffic is usually comprised of short and sparse transmissions of small packets [151]. It may be real-time or not, with varying intervals between transmissions. HTC, on the other hand, is characterized by a continuous flow of large packets. In access networks, MTC generates traffic that is predominantly in the upload direction (from sensing devices to the Cloud), while HTC generates traffic that is mostly in the download direction (from the Cloud to end-users devices). Examples of HTC traffic include instant messaging, VoIP, video/audio streaming, web pages, and file sharing. We present below,

in Subsection 4.2.1, common IoT traffic patterns. We then discuss, in Subsection 4.2.2, how TD might impact the different traffic patterns.

4.2.1 IoT Traffic Patterns

Three common MTC traffic patterns have been recently identified [151]: Periodic Update (PU), Event-Driven (ED), and Payload Exchange (PE). According to the authors, these patterns are those observed in the majority of M2M applications. IoT applications are often comprised of a combination of these patterns. For instance, a river monitoring system periodically sends measurements regarding the water level of rivers to a central server (PU). If the water level surpasses a certain threshold, a flood alarm may be issued (ED). In order to deal with this event, pictures may be sent to the server, or a data stream may be initiated to update the measurements in real-time (PE). The system may also operate dams for controlling the water level by sending commands to actuator devices (ED). These three patterns are described below.

Periodic Update (PU): The PU pattern consists in periodically sending update reports to a central entity. Traffic is generated at a regular interval (e.g., each second), usually comprised of small packets of constant size. An example of this pattern is river monitoring, presented above. Other examples include smart meters reading, and remote health monitoring.

Event-Driven (ED): In this pattern, traffic is sporadic, generated only when an event occurs. An event may be detected by sensing devices (e.g., when a threshold is exceeded), or may be issued by servers (e.g., a human inputs commands). Data size may vary depending on the application and the amount of information of each event. This type of traffic is usually generated in real-time, specially when the events refer to situations that must be acted upon quickly. An example of this traffic pattern is the generation of a security alarm in a surveillance system when something suspicious is detected. Other examples include health emergencies, disaster alerts, and notification of new routes.

Payload Exchange (PE): The PE pattern corresponds to the transfer of large amounts of data. It usually takes place after an event is notified, in case more data is required to deal with the situation. For instance, in the security alarm example presented above for the ED pattern, it is possible to start a video streaming from surveillance cameras to better assess the situation and act accordingly. Another example is firmware upgrading. After receiving the notification that a new firmware version is available, the corresponding devices may start downloading the new version. This pattern occur in real-time or not, depending on the corresponding event.

4.2.2 Impact of TD

Each IoT traffic pattern is sensitive to different network performance metrics. In this work, we consider three metrics that might impact QoE on IoT [148]: end-to-end delay, loss rate and throughput. We analyze below how these performance metrics may affect each pattern, as well as how TD could benefit a prioritized application/device over competitors.

Impact on PU: In the PU pattern, large delays might be misinterpreted by the system as inactivity or faulty behavior. For instance, an IoT device may periodically send information

regarding its status (e.g., uptime, battery level) through the Internet to an IoT platform. If this periodical report is largely delayed, specially if delays are perceived repeatedly, the system may wrongly assume that the device is inactive or faulty. Furthermore, high loss rates may significantly increase the amount of data transmitted over long periods of time due to the retransmissions. Throughput, however, may not be an important performance metric for this traffic pattern, since the data rate is small. In conclusion, TD may turn a prioritized device less likely to be considered inactive or faulty by the system, and smaller loss rates may result in lower energy consumption than competitors (less retransmissions).

Impact on ED: It is important that the real-time notifications from this pattern arrive within the required time limits. ED traffic is thus sensitive to end-to-end delay. Packet loss might also affect this pattern, since retransmissions increase the end-to-end delay. Throughput, as in the PU pattern, may not be important since the amount of ED traffic is small, in most cases. For instance, let us consider the following scenario. A person driving a smart car is making use of a guidance application, which should present to the user the fastest route to the desired destination. The smart car is connected to the Internet, enabling such application to constantly check for better routes. If an accident in the current route occurs, it may cause a traffic jam, significantly affecting the driving time until the destination. In such a situation, the smart car might receive a notification about the accident, causing the driving guidance application to provide a new and faster route to the user. If this notification gets delayed, it might arrive after the user has reached the traffic jam, and from this point it may be impossible to take a detour. Therefore, TD might result in better response times upon the occurrence of events for prioritized devices and applications. In the scenario described above, if cars from a given manufacturer have priority over others, they will perceive smaller delays that may cause a significantly difference in the QoE perceived by users, and this in turn may affect consumer decisions when buying a new car.

Impact on PE: PE traffic is similar to HTC traffic, since it consists of a continuous transfer of large amounts of data. Throughput is thus relevant, while end-to-end delay is not as important as with the ED pattern. Packet loss may impact throughput, since less data is transferred in the same amount of time. A prioritized application may experience a higher throughput. This may result in an overall better QoE, specially when there is human interaction (video/audio streaming).

4.3 SIMULATION RESULTS

In this section we describe several experiments executed with simulation to evaluate the impact of TD on different IoT traffic patterns. The goal of these simulations is to check if even a small prioritization of IoT traffic may result in a significant difference on the QoE perceived by end-users. We simulate each pattern under three different TD scenarios, with a total of 9 simulations. We employed the OMNeT++ [152] simulation framework for implementing and executing these simulations. The duration of each simulation was 1800 seconds, which was set empirically. We observed no significant difference in the results when we executed experiments that took longer than that to complete.

Figure 4.2 shows an overview of the experiments executed. In each simulation, there are three different sets of traffic sources: cross-traffic, high priority, and low priority. All the traffic ingressing at the network goes through a classifier, which identifies the

priority of the traffic (as high or low). A TD mechanism is then employed based on the classification. In each of the three TD scenarios a different TD mechanism was employed. The traffic is then routed to the destination through a single router. The links between the traffic sources and the network have maximum rate of 10 Mbps and a propagation delay of 10 ms, the same delay of the links between the host running the TD mechanism and the router, and between the router and the destinations. The total propagation delay is thus 30 ms, and the maximum output rate of the network is 10 Mbps.

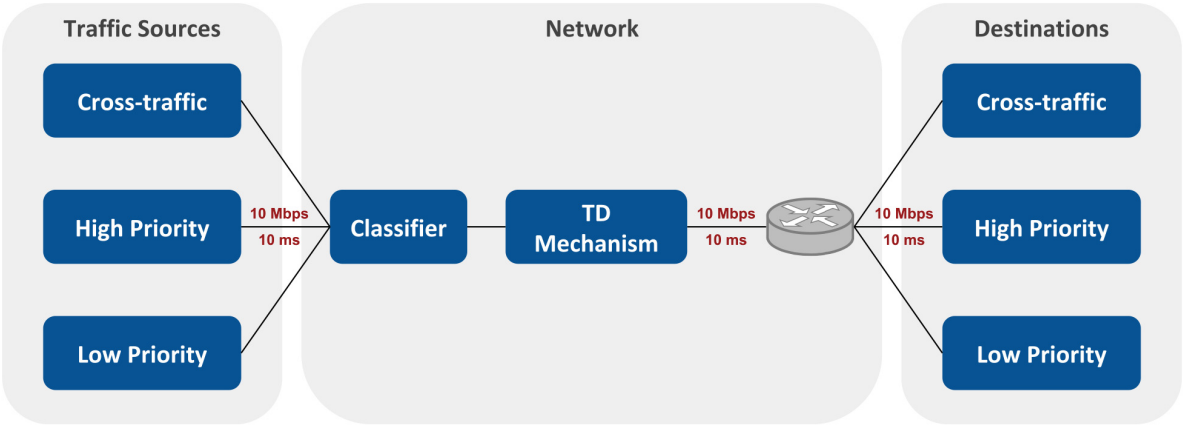


Figure 4.2: Simulation: the main modules.

All the traffic generated by the traffic sources traverses the same path in the network, competing thus for the same network resources. However, traffic from one of the sources (called high priority) is prioritized over others (low priority and cross-traffic). The goal is to check how this prioritization affects the end-to-end performance. We implement this prioritization by reserving a small ratio (1%) of the maximum rate (10 Mbps) of the output link to the high priority traffic, i.e., the high priority traffic has at least 100 Kbps of the bandwidth guaranteed.

The high priority and low priority traffic sources generate traffic corresponding to the three IoT traffic patterns described previously in Section 4.2: Periodic Update (PU), Event-Driven (ED), and Payload Exchange (PE). Cross-traffic simulates background Internet traffic generated by sources other than IoT devices and applications.

The rest of this section is organized as follows. We describe how cross-traffic is generated in our simulations in Subsection 4.3.1. Next, we describe how we implemented the three IoT traffic patterns in Subsection 4.3.2. The TD scenarios are described in Subsection 4.3.3. We then present the results in Subsection 4.3.4.

4.3.1 Cross-traffic

In our simulations, cross-traffic is generated according to the HTC pattern, since IoT traffic competes with HTC for network resources in the Internet [149]. Therefore, cross-traffic was implemented by generating several continuous flows of packets of variable sizes and in different rates. We employed the UDP transport protocol, since it allows better control of the amount of traffic introduced in the network (no congestion control, ACKs, retransmissions, etc.). Each different flow consists of packets with random sizes ranging from 250 to 1000 bytes. Packets are sent at random intervals ranging from 8 to 12 ms, resulting in an average sending rate of 500 Kbps.

In order to evaluate how IoT traffic patterns fare under different conditions of cross-traffic and congestion, cross-traffic is generated in 4 different levels during the 1800 seconds of each simulation. In the first 100 seconds, there is no cross-traffic. From seconds 100 to 500 of the simulation, the cross-traffic rate increases gradually (as new flows are started) up to 10 Mbps. At 1300s the cross-traffic increases by 500 Kbps, and that is repeated at 1600s. Figure 4.3 shows the cross-traffic sending rate during the 1800 seconds of each simulation.

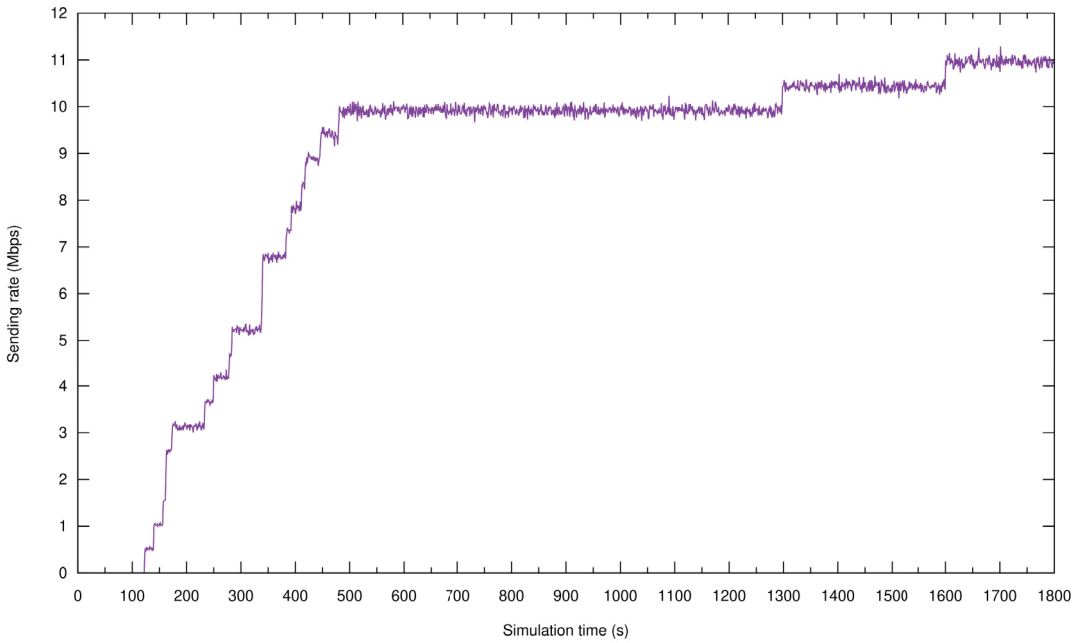


Figure 4.3: Cross-traffic sending rate.

4.3.2 IoT Traffic Patterns

We implemented the different traffic patterns based on the MTC traffic model proposed in [151], and according to the IoT traffic characterization presented in [153]. Each pattern differs in terms of the following parameters: the number of traffic sources, packet sizes, total data sizes, sending rates, and the intervals between transmissions.

The PU pattern consists of sending a constant sized packet (500 bytes) every 5 seconds, employing the TCP protocol. In the experiments we employed 50 high priority sources and 50 low priority sources. Figure 4.4 shows the average sending rate of the aggregate PU traffic of each priority class during the simulations.

The ED pattern consists of sending short bursts at random times. The number of packets of a burst is selected randomly and varies from 1 to 900 packets, each packet size ranges from 800 to 1200 bytes. This pattern represents the notification of an event from sensing devices to a cloud server, or vice versa. In our simulations, we employed 50 high priority sources and 50 low priority sources generating ED traffic. Each source generates an event at a random time instant. Figure 4.5 shows the average sending rate of the aggregate ED traffic of each priority class during our simulations.

We implemented the PE pattern as continuous flows of UDP traffic, similar to the HTC traffic employed as cross-traffic. We employed 30 sources of each priority, thus a total of 60 sources generate PE traffic. Since this pattern usually takes place after the

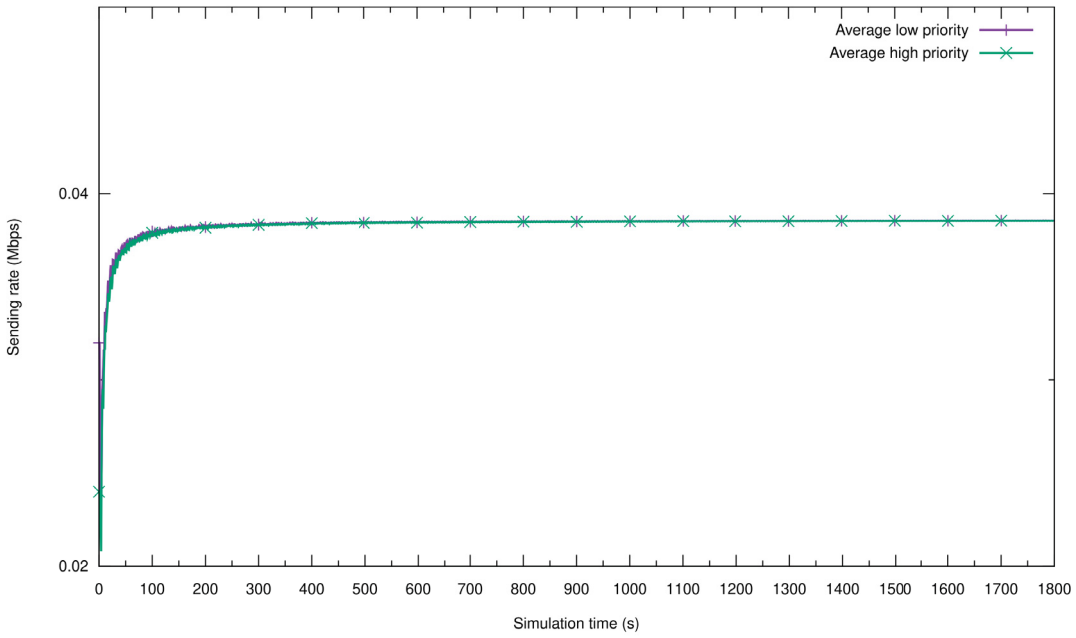


Figure 4.4: PU pattern average sending rate.

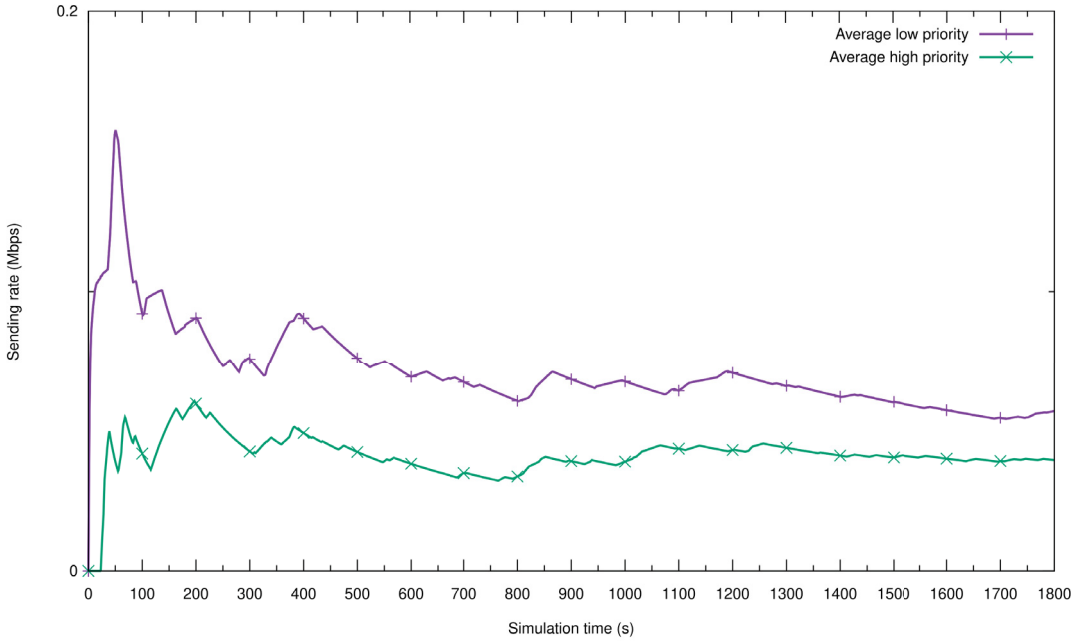


Figure 4.5: ED pattern average sending rate.

notification of an event, we start PE transfers in the same way as in the ED pattern, i.e., each source initiates a transfer at a random time instant. The amount of data transferred varies randomly from 4 to 8 MB. Figure 4.6 shows the average sending rate of the aggregate PE traffic of each priority class in the simulations.

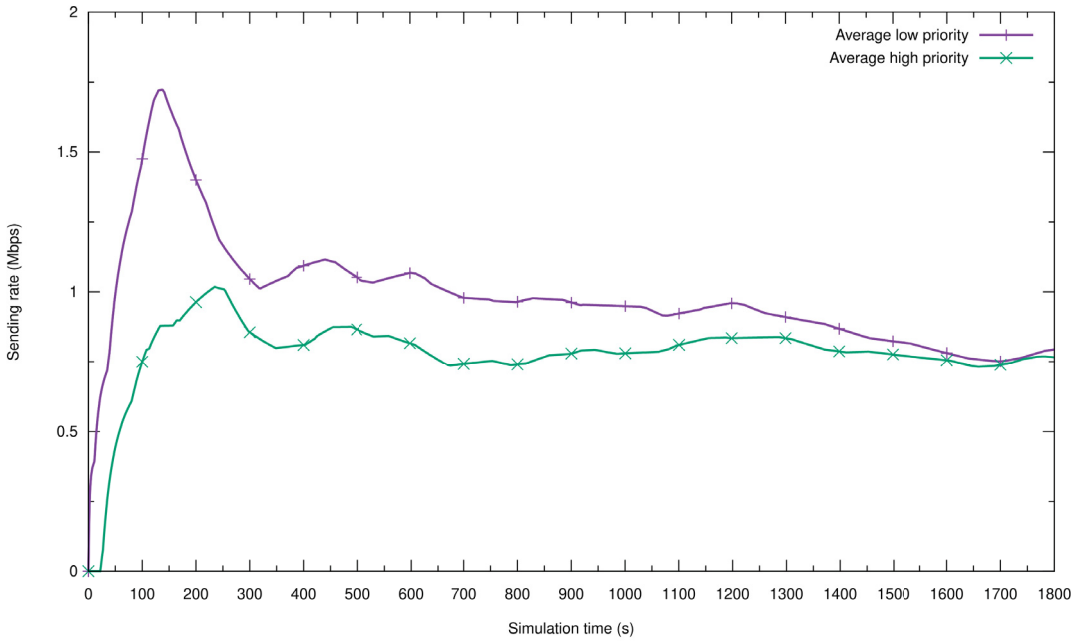


Figure 4.6: PE pattern average sending rate.

4.3.3 TD scenarios

We implemented three different TD scenarios, namely Neutral, Shaping, and Policing. The Neutral scenario employs no TD. The Shaping scenario is based on traffic shaping [39], while the Policing scenario is based on traffic policing [40].

In the Neutral scenario, no TD is performed, thus all traffic is treated equally. A single packet queue is employed. Packets are dequeued and forwarded in the order they arrive, i.e., according to the *First In, First Out* (FIFO) policy. The queue has maximum size equal to 100. When the queue is full, all arriving packets are dropped, employing the *Drop-tail* (DT) approach.

In the Shaping scenario, the reserved rate (100 Kbps) is enforced by queuing high priority packets in a separate queue and forwarding them first. Two DT queues are employed, one for each priority. All high priority packets that fall under the reserved rate are queued in the high priority queue. High priority packets that exceed this rate are queued together with the rest of the traffic, in the low priority queue. Packets from the high priority queue are always forwarded first. Both DT queues have maximum size equal to 100.

In the Policing scenario, a single DT queue with maximum size of 100 is employed. Low priority packets that exceed the maximum output rate of the network (10 Mbps) are dropped. High priority packets exceeding the reserved rate are reclassified as low priority, thus becoming subject to the same dropping conditions as the low priority packets.

4.3.4 Results

We present below the results for each traffic pattern under the different TD scenarios. We focus on the most relevant metrics for each pattern, which were discussed in Section 4.2.

PU results: As discussed previously, packet losses in the PU pattern may result in a significant increase in the amount of data transmitted over long periods of time. We

then evaluated the number of Retransmission Timeouts (RTOs), computed exactly as the TCP protocol does. Figure 4.7 shows the Cumulative Distribution Function (CDF) of the number of RTOs, for each priority class and under each TD scenario. Each CDF corresponds to the portion of the time during which the corresponding number of RTOs occurred.

In the Neutral scenario (Figure 4.7(a)), the CDFs for both priorities were very similar. 15092 high priority packets and 15086 low priority packets were sent. High priority traffic raised 2418 RTOs in total, while the low priority traffic raised 2429 RTOs.

In the Shaping scenario (Figure 4.7(b)), the high priority traffic raised no RTOs. 17341 and 14916 packets were sent by the high and low priority sources, respectively. A total of 2626 RTOs were raised by the low priority traffic.

In the Policing scenario (Figure 4.7(c)), the high priority traffic presented RTOs in only about 10% of the simulation, while the low priority traffic in about 25%. High priority traffic consisted of 17133 packets, while the low priority consisted of 16866 packets. A total of 123 RTOs were raised by the high priority traffic, and 531 RTOs by the low priority traffic. In this scenario, there were less RTOs than in the Neutral scenario, for both priorities. However, even with this improvement, the Policing scenario introduced a difference between the priorities, which did not exist in the Neutral scenario.

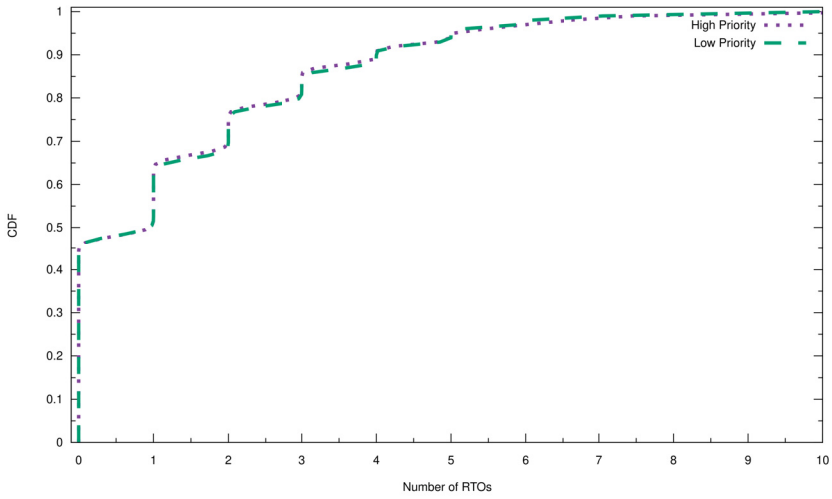
ED results: The end-to-end delay is an important metric to evaluate the ED pattern, since it consists of real-time event notifications. Figure 4.8 shows the average end-to-end delay, in milliseconds, experienced by packets from of each priority class, under each TD scenario.

As the amount of cross-traffic increased, the average end-to-end delay also increased in a similar way for both priorities in the Neutral scenario (Figure 4.8(a)). In the other two scenarios (Figures 4.8(b) and 4.8(c)), however, the average end-to-end delay increased significantly more for the low priority traffic.

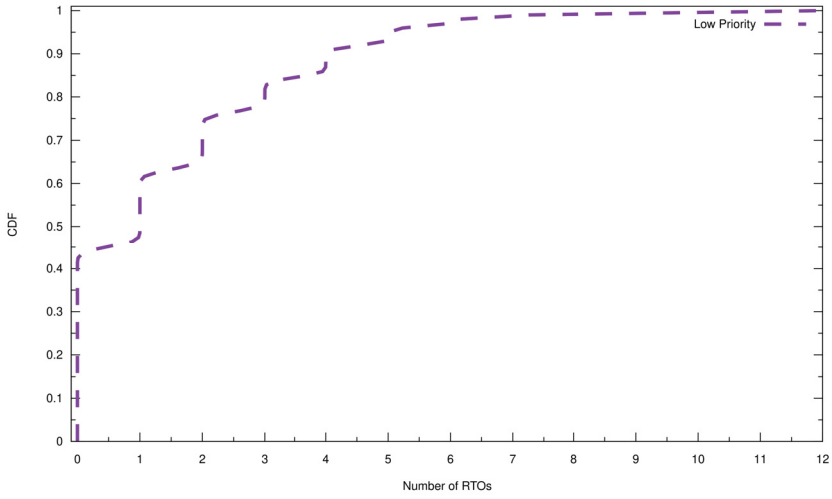
In the Neutral scenario, 759 low-priority traffic packets suffered end-to-end delays larger than 1 second, while for the high priority traffic 1037 packets suffered similar delays. In the Shaping scenario, 815 low priority traffic packets had end-to-end delays larger than 1 second, while only 297 high priority packets suffered similar delays. In the Policing scenario, 475 low priority packets suffered delays larger than 1 second, while 239 high priority packets suffered similar delays.

PE results: PE traffic consists of transferring larger amounts of data than the other IoT patterns. Therefore, as discussed previously, throughput is important, since faster transfers may result in better QoE. We evaluated the throughput achieved by traffic of both priority classes under each TD scenario. At first the reserved rate (100 Kbps) had no significant impact on the throughput. We thus ran our simulations again, employing a larger reserved rate for the high priority traffic. We set the reserved rate to 10% of the link bandwidth, i.e., 1 Mbps. The goal was to check if a larger reserved rate would result in a significant difference in throughput between the two traffic priorities.

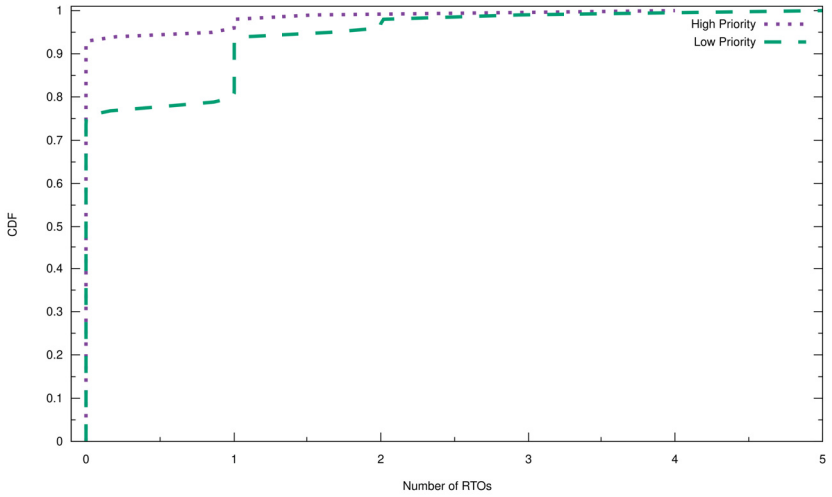
Figure 4.9 shows the average throughput for each priority, under each TD scenario. It is possible to observe that the average throughput for high priority traffic increased in the Shaping and Policing scenarios, in comparison with the Neutral scenario. The difference is most noticeable after 1300 seconds of simulation, when cross-traffic reaches a higher level.



(a) Neutral scenario.

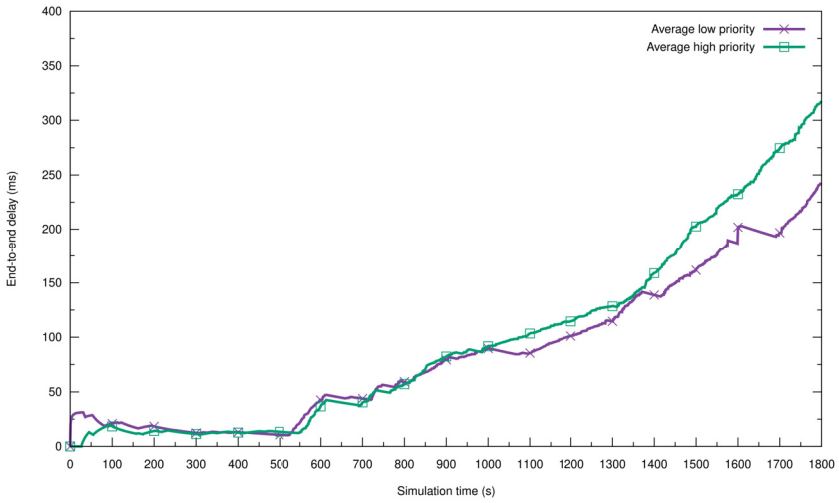


(b) Shaping scenario.

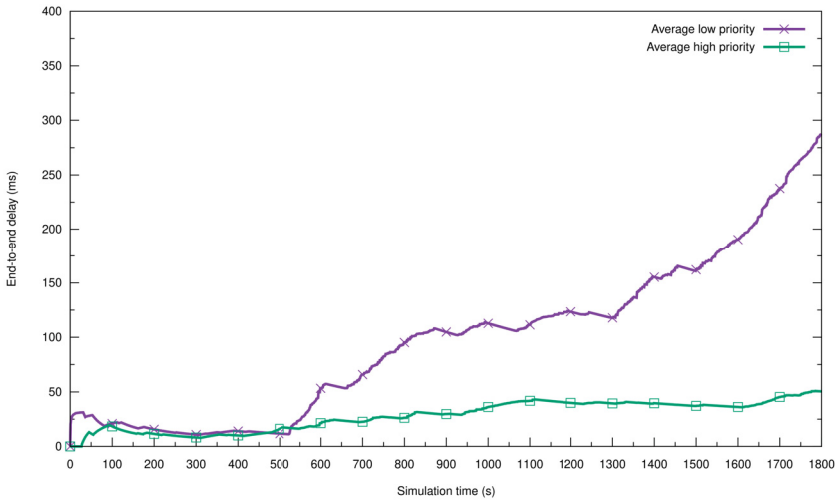


(c) Policing scenario.

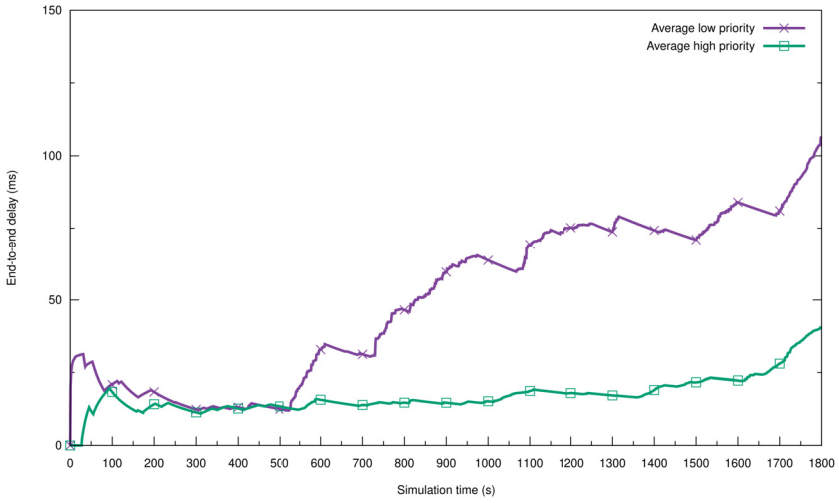
Figure 4.7: CDF of the number of RTOs for the PU pattern.



(a) Neutral scenario.



(b) Shaping scenario.

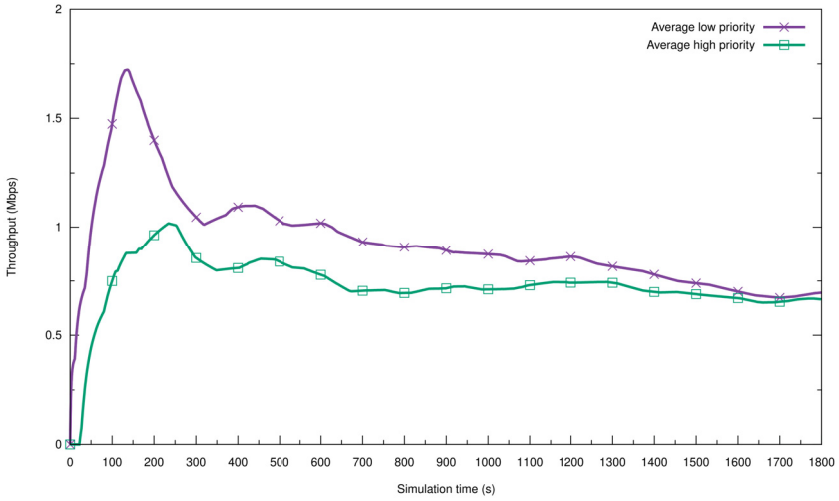


(c) Policing scenario.

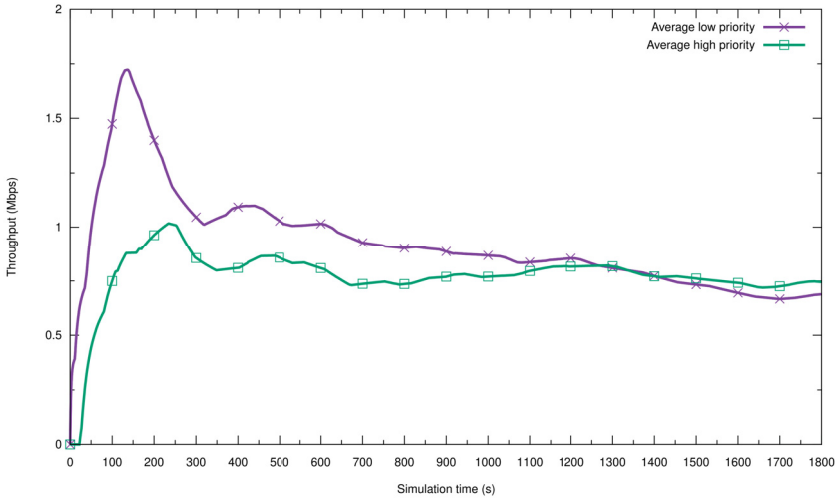
Figure 4.8: Average end-to-end delay for the ED pattern.

4.4 DISCUSSION

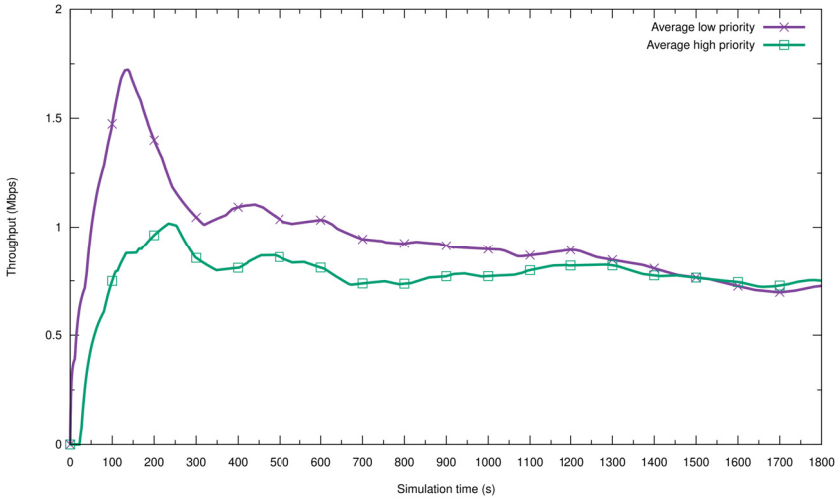
Results show that even a small reserved rate (1%) may be enough to create a significant difference on the QoE perceived by an end-user, which might result in unfair competition.



(a) Neutral scenario.



(b) Shaping scenario.



(c) Policing scenario.

Figure 4.9: PE pattern average throughput.

Regarding the PU pattern, TD may have a meaningful impact depending on the application. In cases where energy consumption is important, for example, a larger number of packet

retransmissions (due to packet loss) may increase the amount of data transmitted during long periods of time. The impact on the PE pattern may also depend on the application. For example, for real-time streaming an affected throughput may result in significant QoE degradation. We argue that TD impacts the ED pattern the most, since it is important that real-time notifications arrive on time. Therefore, the end-to-end delay may be a good metric for detecting differentiation of real-time IoT traffic.

5 DETECTING TRAFFIC DIFFERENTIATION

In this chapter, we evaluate a solution for detecting TD between two end-hosts in the Internet. This solution allows the detection of TD triggered by application, and combines techniques employed by several existing solutions for TD detection, which were presented in Chapter 3. We employ several different metrics to enable the detection of multiple types of TD. For instance, depending on how TD is affecting traffic, analyzing only one metric may not be enough. We employ four metrics: delay, inter-arrival times of packets, throughput, and loss rate.

The rest of this chapter is organized as follows. Section 5.1 describes the solution for detecting TD. We evaluate the solution in Section 5.2. Then we discuss the results in Section 5.3.

5.1 PROPOSAL FOR DETECTING TD

Our TD detection solution requires access to the two end-hosts between which the presence of TD is to be checked. The solution transmits two different types of traffic between the end-hosts, and then compare the distributions of the measurements taken for each type.

The rest of this section is organized as follows. In Subsection 5.1.1 we describe how measurements are taken in our proposal. Next, the method for comparing the measurements and inferring TD is described in Subsection 5.1.2.

5.1.1 TD Detection Measurements

Measurements are made using two types of traffic, called application and baseline, which are sent simultaneously between the two end-hosts for t seconds. The application flow consists of a previously prepared traffic from an application, such as BitTorrent, Netflix, Skype, or on a specific port. The baseline flow corresponds to the application flow: it may be the same as the application flow but encrypted (as in the Wehe solution, described in Chapter 3), generated on-demand with a random payload (as in the DiffProbe solution), and/or on a different port. It is essential that both flows have similar features, such as rate and packet sizes, in order to make them comparable (as in the Packsen solution). The rationale is that if one flow generates more packets, it has a higher chance of being affected by packet drops, even if TD is not being employed, since there will be many more packets from this flow in the routers queues than from the other flow.

For each flow, the following measurements are taken on the end-host receiving the traffic: delay, inter-arrival times of packets, throughput, and loss rate. The flows are sent r times between the end-hosts, in both directions (downstream and upstream). It is necessary to repeat the flows several times in order to reduce the effect of noise in the obtained measurements [45].

5.1.2 TD Detection Inference

For inferring the presence of TD between the end-hosts, the proposed solution compares the measurements obtained for the two flows (baseline and application). We employ the Kolmogorov-Smirnov (KS) hypothesis test in order to check if any measurement distribution of one flow is statistically different from the same measurement distribution of

the other flow. The idea is that, in a neutral communication, the measurements for both flows follow the same distribution: they were sent simultaneously and in the same way, then they should be received in the same way as well [45]. We employ a 95% confidence level for the KS test.

For each traffic direction (downstream and upstream), r sets of measurements are obtained. Each set contains results for four metrics (delay, inter-arrival times, throughput, and loss rate) for each type of flow (application and baseline). We employ the KS test to compare each metric from both flows, for each of the r sets. TD is detected between the end-hosts if for at least one metric and in at least one direction, the distributions from each flow are different on all r sets, according to the KS test.

Depending on how TD is implemented, different metrics may be affected. Therefore, a difference in the obtained measurements may be observed only for some of the four metrics employed. For instance, if the traffic from a specific application is forwarded with lower priority, it is possible that only the delay is affected, while the loss rate is not. Similarly, traffic may be discriminated only in one direction. Furthermore, the path effectively traversed by the traffic between the end-hosts may be different in each direction, which could cause the TD to happen only in one direction.

5.2 EVALUATION: DETECTING TD

In this section we describe simulations for evaluating our solution for detecting TD. The goal is to check if the proposed solution is capable of detecting differences on the performance experienced by two separate flows while traversing the same network, under different scenarios. In particular, we aim at checking if the strategy of combining four different metrics yields better results than employing a single metric. Furthermore, we also evaluate the parameters t and r – duration and repetitions of each flow, respectively. We employed the OMNeT++ [152] simulation framework for implementing and executing these simulations.

We executed several simulations, varying how TD was employed and the amount of cross-traffic present. Several different TD scenarios were defined, employing different types of TD, or no TD at all. We repeated each simulation 10 times, for evaluating different values of the parameter r (1 to 10). In each simulation, two flows were transmitted for 30 seconds between end-hosts. This allows us to evaluate different values of the parameter t using subsets of the obtained data (5, 10, 15, 20, 25, and 30s). From the two flows, one corresponds to a low priority traffic, which will have its performance degraded by the network. The other flow corresponds to a high priority traffic, which will not suffer interference by the network. Then, based on the measurements acquired in the simulation, we employ our proposal to detect differentiation of the two flows.

Figure 5.1 shows the topology employed by the simulations. Four types of traffic are generated by four different sources: (i) high priority measurement traffic and (ii) low priority measurement traffic are each generated by different end-hosts; Furthermore, (iii) high priority cross-traffic and (iv) low priority cross-traffic are each generated by different sets of end-hosts. The size of this sets depends on the amount of cross-traffic in each simulation. All traffic sources are connected to a module from the simulator that represents the Internet. This module is responsible for treating the different types of traffic in different ways, according to the TD scenario being employed. The TD module is then connected to a router that forwards traffic to the corresponding destinations. For each type of traffic there is a receiver host, on which the measurements are taken. All links –

from sources to the TD module, and then to the router, and finally to the receiver hosts – have a bandwidth of 100Mbps and propagation delay of 10ms.

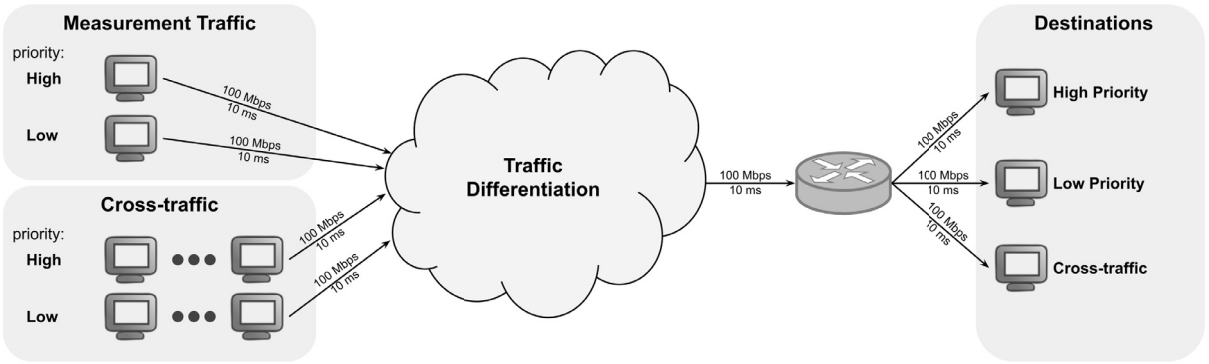


Figure 5.1: Topology employed by the simulations.

The high and low priority flows are generated simultaneously and in the same way. Both consist of a continuous stream of UDP packets. The UDP protocol provides a finer control over how traffic is generated than the TCP protocol, thus we employed UDP in the simulations. The size of the packets varies randomly from 450 to 550 bytes. The sending interval varies randomly from 343,35μs to 419,65μs, resulting in a sending rate of 10Mbps, on average. Cross-traffic is generated the same way, but the size of packets varies from 400 a 600 bytes. The number of cross-traffic flows generated in the simulations varies from 0 to 5 for each priority. Therefore, the different levels of cross-traffic are 0, 20, 40, 60, 80 and 100Mbps, depending on the configuration of the simulation. Half of the cross-traffic is always of high priority, while the other half is of low priority. All random values employed follow an uniform distribution, and each of the 10 repetitions employs a different seed for the random number generator.

The TD scenarios employed in the simulations were defined according to three parameters: delay, throughput, and drop probability (the chance of a packet being dropped). TD is done by employing different values for these parameters for each traffic priority. We defined 17 TD scenarios. One of these scenarios is the *Neutral*, in which the TD module employs the same values for both priorities: the delay ranges from 90 to 100ms, throughput ranges from 90 to 100Mbps, and the drop probability is 1%. In the other TD scenarios, values affecting negatively the parameters are employed for low priority traffic, while the same values of the *Neutral* scenario are always employed for high priority traffic. Table 5.1 shows the values for all TD scenarios defined. In the table, the TD scenarios named *DelayX* indicate that delays about X% larger are employed for low priority traffic. Similarly, *RateX* scenarios employ a throughput about X% smaller. *DropXY* corresponds to a X.Y% drop probability. Some TD scenarios combine larger delays and smaller throughputs (*DelayRateX*).

In each simulation, we compare the distribution of each of the four metrics (delay, inter-arrival times of packets, throughput, and loss rate) of the low priority measurement traffic with the corresponding distribution of the high priority measurement traffic, employing the KS test, according to the TD detection proposal described in Section 5.1. The rates of cross-traffic were 0, 20, 40, 60, 80 and 100Mbps. Each of the 17 TD scenarios was simulated 10 times for each amount of cross-traffic, resulting in 1020 simulation runs. We evaluate the values 5, 10, 15, 20, 25, and 30s for the parameter t, by considering only subsets of the obtained measurements. For instance, for evaluating $t = 10s$, we only considered the first 10s worth of measurements from each flow. Similarly, parameter r was

Table 5.1: TD scenarios: values employed for low priority traffic.

TD Scenario	Delay (ms)	Throughput (Mbps)	Drop (%)
Neutral	90-100	90-100	1
Delay5	**90-105**	90-100	1
Rate5	90-100	**85-100**	1
DelayRate5	**90-105**	**85-100**	1
Delay10	**90-110**	90-100	1
Rate10	90-100	**80-100**	1
DelayRate10	**90-110**	**80-100**	1
Delay15	**90-115**	90-100	1
Rate15	90-100	**75-100**	1
DelayRate15	**90-115**	**75-100**	1
Delay20	**90-120**	90-100	1
Rate20	90-100	**70-100**	1
DelayRate20	**90-120**	**70-100**	1
Drop11	90-100	90-100	**1.1**
Drop12	90-100	90-100	**1.2**
Drop15	90-100	90-100	**1.5**
Drop20	90-100	90-100	**2**

evaluated with values ranging from 1 to 10 – e.g., for $r = 5$ TD inference is based on the first 5 executions.

We present the results in two parts. First, in Subsection 5.2.1 we present results for each metric separately: we show how each metric fares for different values of t, under different TD scenarios and cross-traffic. Next, in Subsection 5.2.2 results combining the four metrics are presented: we show whether our TD inference is correct or not for different values of r and t, under different TD scenarios and cross-traffic.

5.2.1 Evaluating the Metrics Separately

In order to evaluate each metric, we observed in how many of the 10 repetitions TD is detected for each metric, individually. For each metric, under each TD scenario and level of cross-traffic, and for each value of t, we count how many times the distribution of measurements from the low priority traffic is statistically different from the distribution from the high priority traffic – which characterizes TD. We call the rate TD is detected across the 10 repetitions the *detection rate*. For instance, if for a particular configuration TD was detected in 3 out of the 10 simulations, the detection rate is 30%. Therefore, the results presented in this subsection are all in terms of the detection rate achieved by each metric under several different configurations.

We start with the *Neutral* TD scenario. Figure 5.2 shows the detection rate for the *Neutral* TD scenario. In this scenario, both flows (low and high priority) are treated the same while traversing the network. Therefore, a TD detection in this case corresponds to a false-positive. However, we propose to detect TD combining several repetitions and metrics, which may mitigate inference errors. Each graphic in Figure 5.2 corresponds to a metric, and in each graphic the results for different values of t are plotted. The vertical axis shows the detection rate, and different levels of cross-traffic are shown in the horizontal axis. On the leftmost graphic (delay), larger values for the detection rate can be observed. In the other cases, the detection rate is smaller, specially for the throughput metric (third from left to right).

We now present results for scenarios in which TD is employed. Figure 5.3 shows the detection rate for all *DelayX* scenarios. In these scenarios, the detection rate for delay

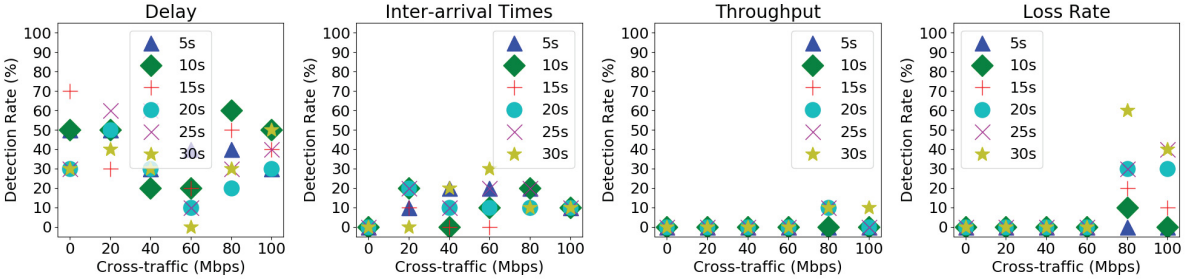


Figure 5.2: Detection rate in the *Neutral* scenario.

and inter-arrival times metrics was 100% for all values of t and cross-traffic. For the other two metrics, the detection rate was larger for $t \geq 15s$. The same can be observed for the *DelayRateX* scenarios in Figure 5.4.

Figure 5.5 shows the detection rate in the *RateX* scenarios. In these scenarios, the inter-arrival times metric had larger values for the detection rate. For the *Rate5* scenario (Figure 5.5(a)), the detection rate was larger when the cross-traffic was up to 40Mbps. However, for all the other scenarios, the inter-arrival times metric achieved 100% detection rate when $t \geq 20s$ for all levels of cross-traffic.

Figure 5.6 shows the detection rate for the *DropXY* scenarios. In these scenarios, the throughput and loss rate metrics presented larger detection rates in the *Drop15* and *Drop20* scenarios (Figures 5.6(c) and 5.6(d), respectively) than the other metrics. However, even for these two metrics, the detection rate was consistently larger only in the *Drop20* scenario when $t \geq 20$.

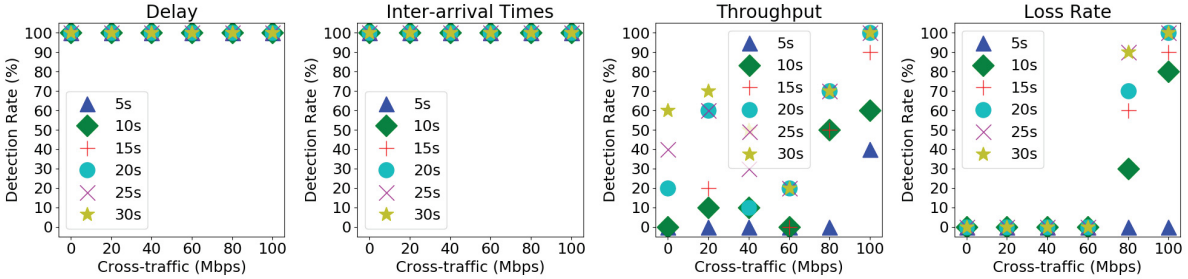
5.2.2 Evaluating the Metrics Combined

We now show results of TD detection combining all four metrics, and for different values of r (number of repetitions). Figure 5.7 shows whether the TD detection was correct or incorrect in the *Neutral* scenario. Each graphic corresponds to a different value of t (5, 10, 15, 20, 25, and 30s). The vertical axis corresponds to the level of cross-traffic, and the horizontal axis corresponds to the value of r. Despite the large detection rates observed previously for the delay metric in Figure 5.2, there were only a few false-positives, all for less than 2 repetitions ($r \leq 2$).

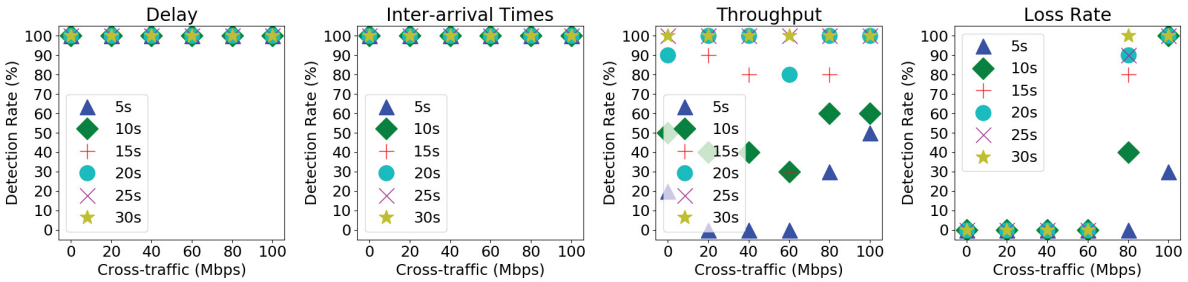
In the *DelayX* and *DelayRateX* scenarios, the TD detection was correct for all values of r, t, and cross-traffic. This result was expected, given the detection rates achieved by the delay and inter-arrival times metrics, shown in Figures 5.3 and 5.4.

In the *RateX* scenarios, TD detection was always correct when $t \geq 20s$ for all scenarios except *Rate5*, which is shown in Figure 5.8. It is possible to observe in the figure that TD detection for the *Rate5* scenario was correct when $t \geq 15s$ and the level of cross-traffic was up to 40Mbps.

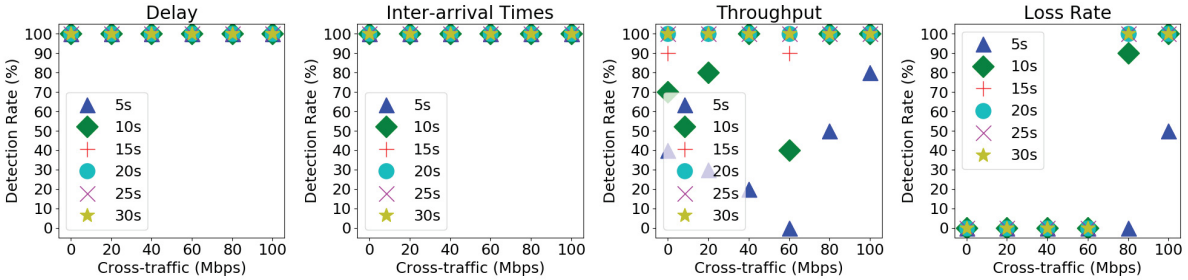
In the *DropXY* scenarios, only *Drop20* presented good results. Figure 5.9 shows the TD detection results for the *Drop20* scenario. The TD detection was always correct when $t \geq 20s$ and cross-traffic up to 80Mbps. When $t = 30s$, TD detection was always correct, regardless of cross-traffic.



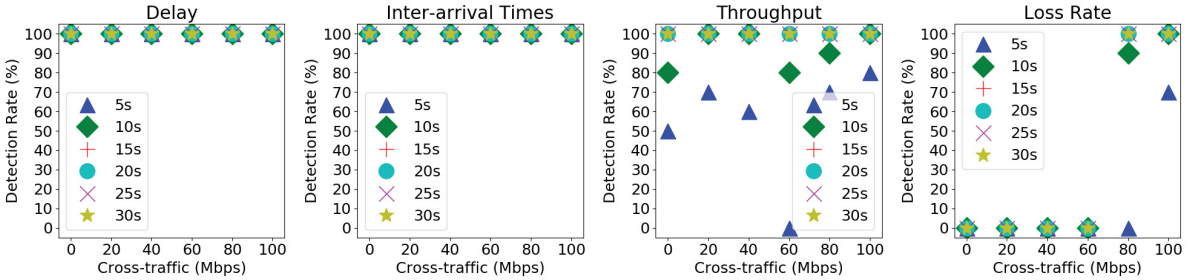
(a) Delay5



(b) Delay10



(c) Delay15

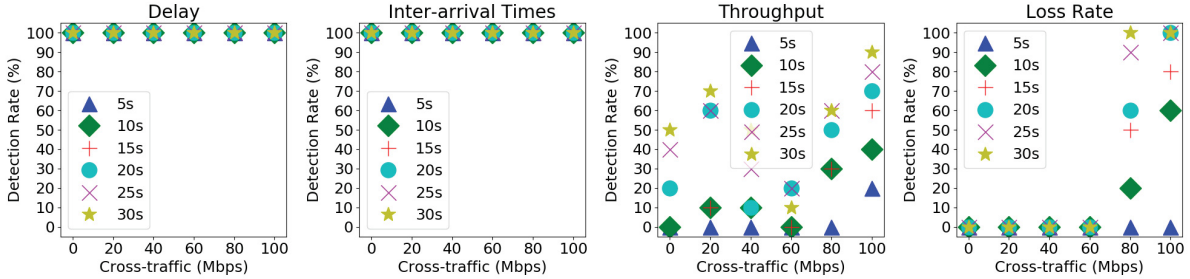


(d) Delay20

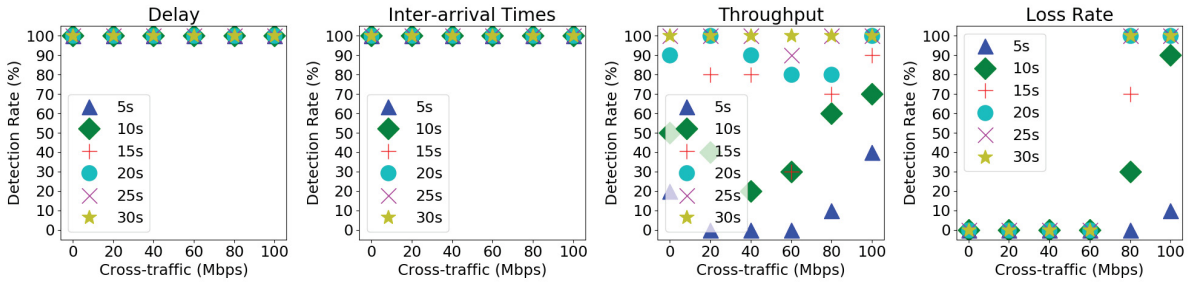
Figure 5.3: Detection rate in the *DelayX* scenarios.

5.3 DISCUSSION

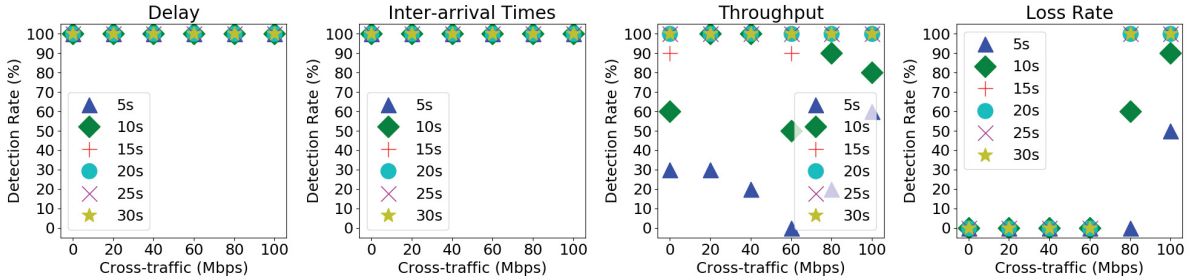
Results show that our proposal for detecting TD was able to identify differences in the performance experienced by two different flows while traversing the same network under several conditions. Our strategy of combining four different metrics was able to consistently detect the different types of TD being employed. Results also show that it is important to repeat the measurements several times, specially for avoiding false-positives in case there is no TD. In our simulations, no false-positives occurred for $r \geq 3$. Results were also impacted by the duration of the flows. Overall, detection rates were larger for $t \geq 20s$.



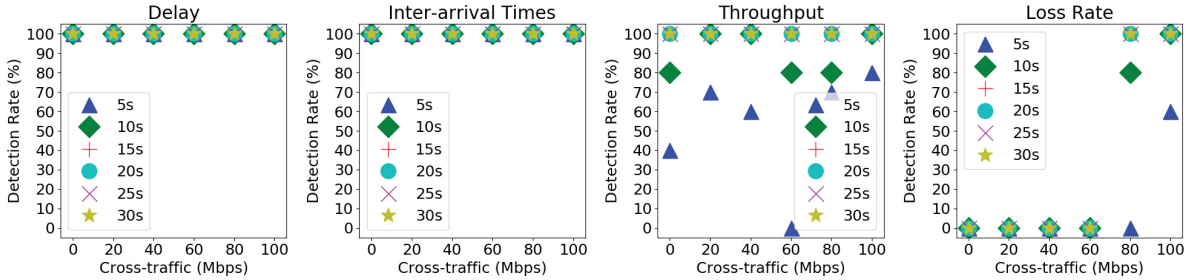
(a) DelayRate5



(b) DelayRate10



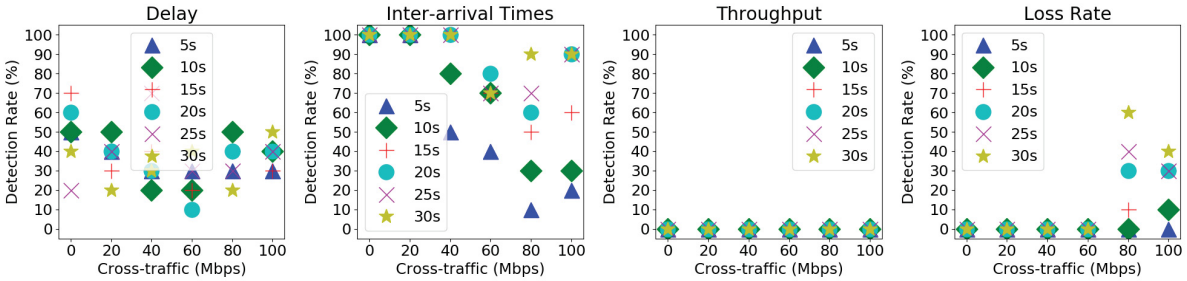
(c) DelayRate15



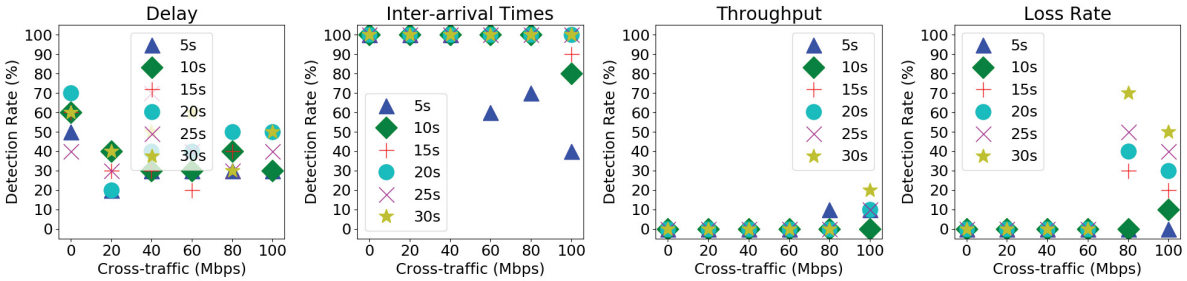
(d) DelayRate20

Figure 5.4: Detection rate in the *DelayRateX* scenarios.

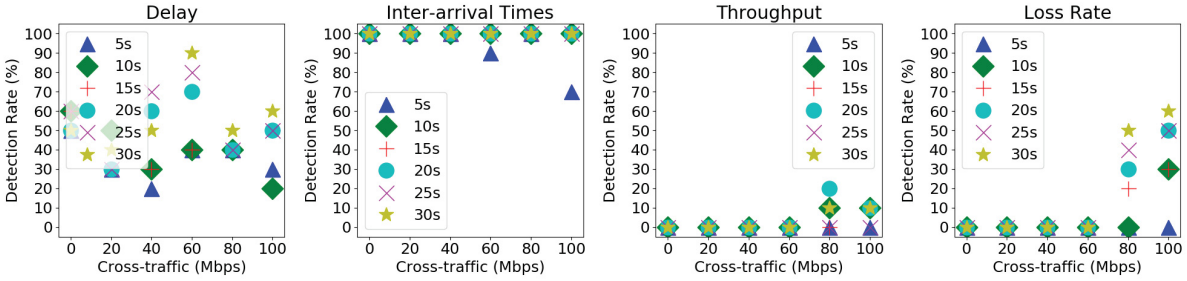
However, in the *DropXY* scenarios, our proposal was able to detect TD only when the drop probability for low priority flows was twice the probability for the high priority flows (2% vs 1%). Furthermore, higher levels of cross-traffic also caused our proposal to fail in detecting TD in some cases. When the total volume of traffic traversing the network is greater than the available bandwidth, routers queues may become full and start dropping and/or delaying packets. In such cases, flows of both priorities may experience larger delays and loss rates. If the impact of this congestion on the measurements is much greater than the impact of the TD, the distributions of measurements may not be different enough for the KS test to detect.



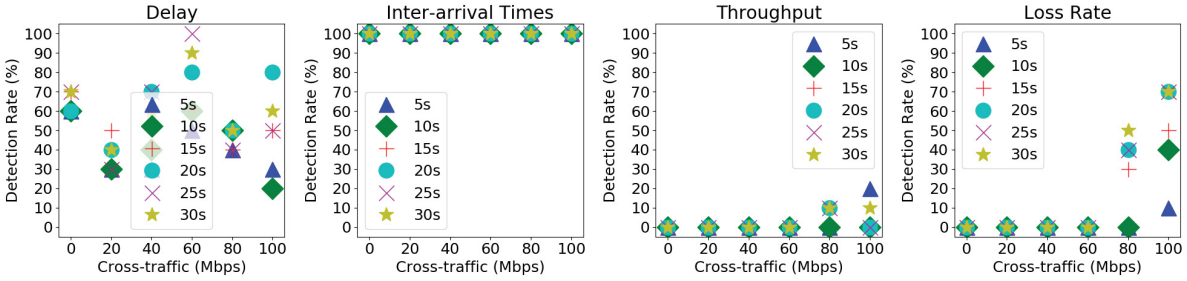
(a) Rate5



(b) Rate10

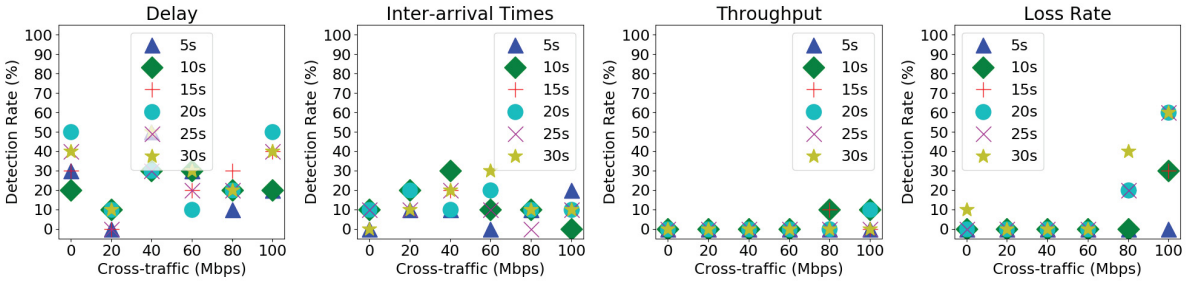


(c) Rate15

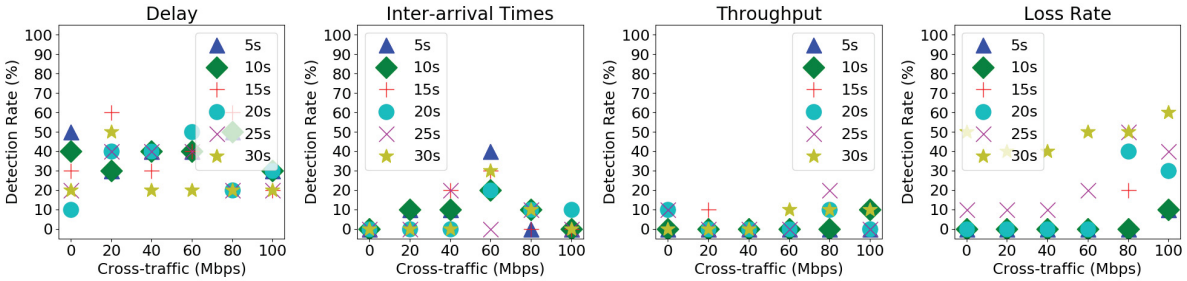


(d) Rate20

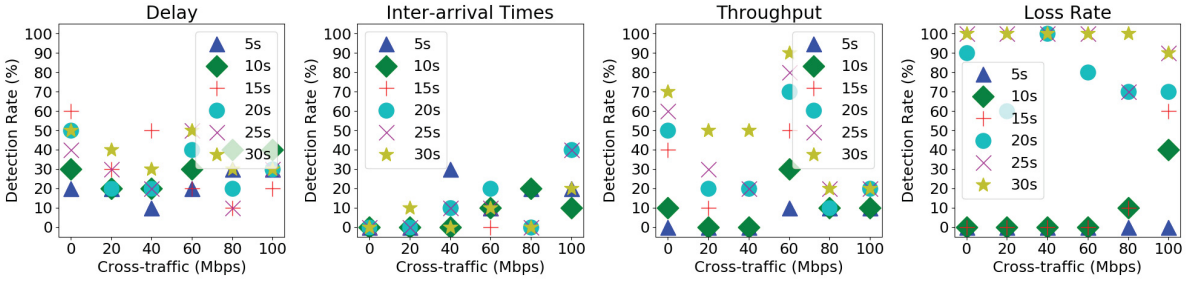
Figure 5.5: Detection rate in the *RateX* scenarios.



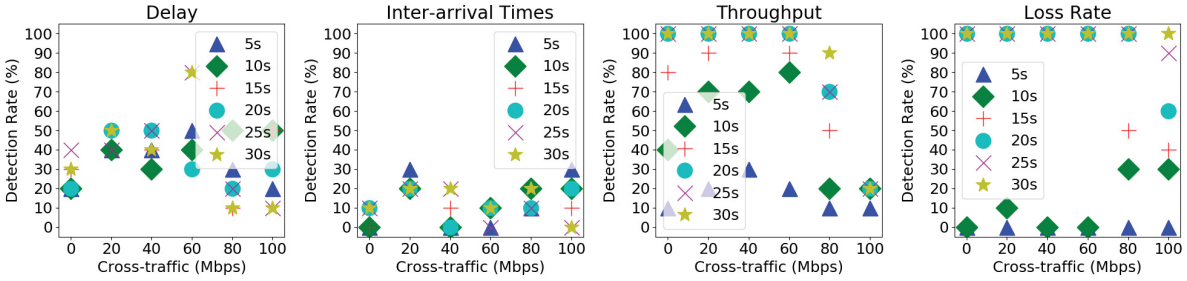
(a) Drop11



(b) Drop12



(c) Drop15



(d) Drop20

Figure 5.6: Detection rate in the *DropXY* scenarios.

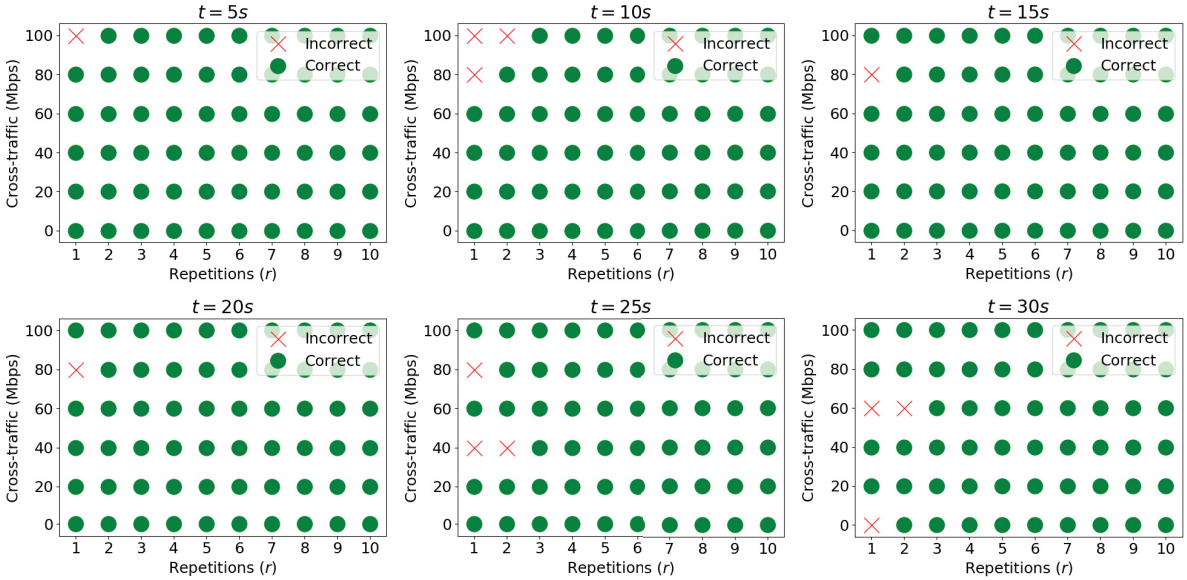


Figure 5.7: TD detection for the *Neutral* scenario combining all metrics.

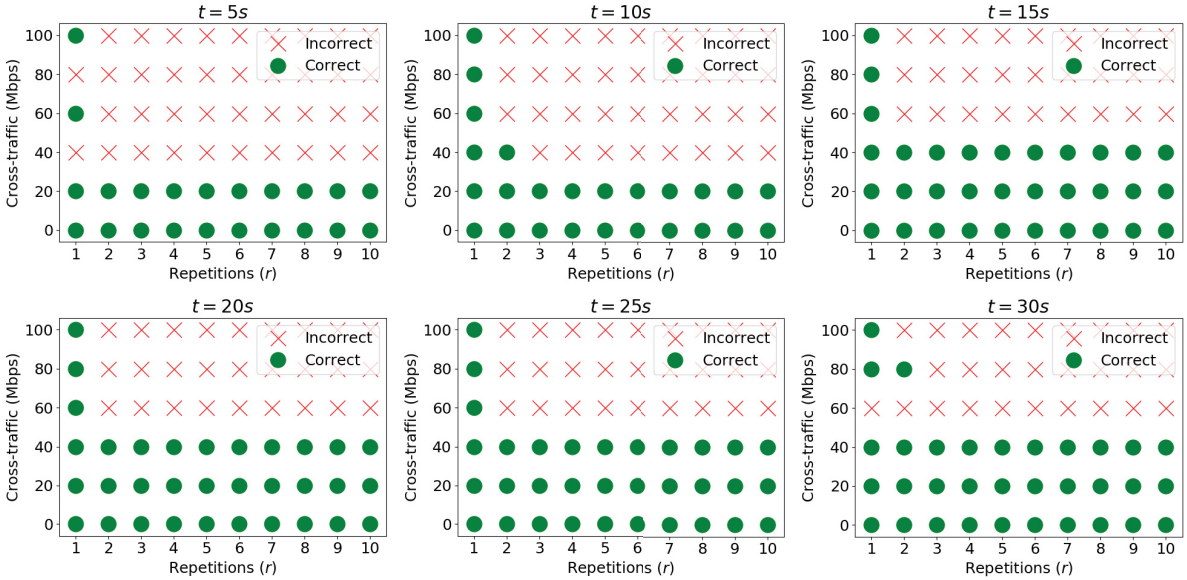


Figure 5.8: TD detection for the *Rate5* scenario combining all metrics.

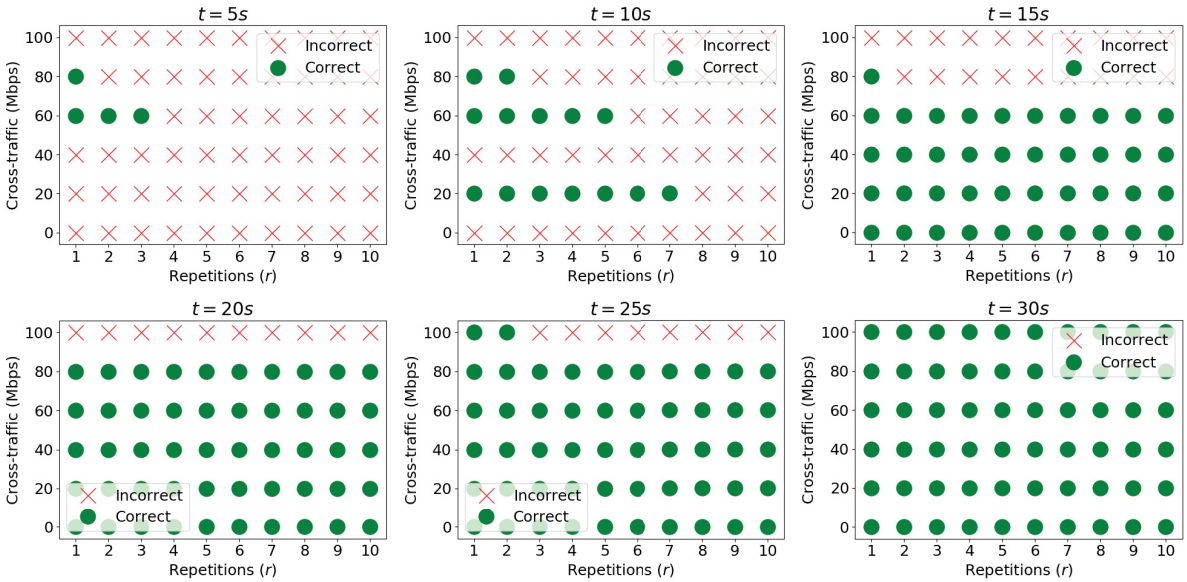


Figure 5.9: TD detection for the *Drop20* scenario combining all metrics.

6 LOCATING TRAFFIC DIFFERENTIATION

In this chapter, we present proposals for building a complete and future-proof solution for monitoring NN in the Internet. There are several solutions for detecting different types of NN violations, as we described in Chapter 3. One of the goals of the proposals presented in this chapter is to unify those solutions in a single framework, addressing several of the open challenges we identified in the current state of the art.

We first propose a conceptual NN monitoring model that continuously gathers measurements and/or TD inferences from any kind of device (mobile or wired) or third-party applications participating in the system. The idea is to combine all available data to make NN-related inferences. Then, in order to enable the implementation of this model, we propose a strategy for locating which AS is employing TD that combines TD detection results and/or measurements from several different vantage points. The proposed strategy thus addresses two problems: the problem of locating TD in the Internet, as well as the problem of combining several different inferences and/or measurements for enabling the implementation of our general model.

Although multiple solutions for detecting TD in the Internet have been proposed, there are still only a few solutions for locating where in the network the source of TD is. As discussed previously in Chapter 3, some proposals for locating TD rely on *traceroute*-like techniques, which may not be reliable, while others assumes prior knowledge of the exact topology for the whole network, which may not be feasible in the Internet. Our proposal does not rely on *traceroute*-like techniques, and does not require the exact host-level topology of the Internet, as previous proposals do. We assume prior knowledge of the AS-level topology of the Internet, instead of the host-level topology. Our proposal presents an alternative to the *traceroute*-based solutions, while making more realistic assumptions.

The proposal takes advantage of AS-level routing properties to identify which AS is practicing TD. Our strategy combines measurements from different end-hosts, and makes inferences based on the possible AS-level paths between the end-hosts. The rationale is that if a particular AS is in all possible paths between two end-hosts, then the measurements from such end-hosts is guaranteed to have traversed that AS, and thus its behavior can be assessed. Therefore, we examine all possible AS-level paths between end-hosts according to inter-domain routing policies, instead of running measurements with *traceroute*-like techniques. Furthermore, we also present metrics for defining where the measurement points should be located, since the effectiveness of our proposal depends on how well positioned measurement points are.

In this chapter, we evaluate our proposal for locating TD in two parts: first, we conducted experiments on the PlanetLab global testbed. The goal of these experiments is twofold: (i) to check our assumptions regarding the properties of AS-level paths; and (ii) to confirm the limitations of *traceroute*-like techniques for obtaining such paths. We then describe several simulations for assessing the efficiency of our proposal for locating TD under different scenarios.

The rest of this chapter is organized as follows. Section 6.1 describes the general NN monitoring model. Next, we present an overview of AS-level routing in the Internet, in particular the properties of AS-level paths in Section 6.2. We then describe our strategy for locating TD in Section 6.3. In Section 6.4, we first describe the AS-level topology graph we employ in our evaluations, which is required by the proposed solution for locating TD.

We then describe the PlanetLab experiments. Finally, in Section 6.5, we present several simulations for evaluating our proposal for locating TD.

6.1 A GENERAL NN MONITORING MODEL

In this section, we propose a conceptual NN monitoring model which addresses several challenges presented in Section 3.4. The model allows any kind of device (mobile or wired) or third-party applications to join the system, contributing with measurements and/or checking how their traffic is being treated. One of our goals is to unify the solutions for TD detection (such as those presented in Chapter 3) in a single framework, making it possible for them to contribute to a common knowledge base. An earlier version of this model was published in [51].

The main idea is to continuously gather measurements and/or TD inferences from a plethora of sources, such as devices and other TD detection solutions, in a crowdsensing approach. The obtained data is then analyzed in real-time. If TD is suspected to be happening, and/or some AS is suspected of being responsible for TD, more measurements may be requested if necessary for further investigating the suspicious case – in a hybrid active/passive approach. We aim at not only detecting TD, but also locating where in the network it happened, by combining all the measurements and/or inferences collected by the system.

All acquired data is made available in an Open Data paradigm for further analysis by the system itself or by any third party system. This strategy takes advantage of several features of distributed systems, thus enabling such systems to incorporate the proposed model. We argue that the NN-related issues discussed in this work should be taken into account when designing distributed systems or any other Internet-based application.

The authors of [85] advocate the use of a similar approach to build a "citizen observatory" of NN in the context of mobile Internet. We build on that idea targeting a NN monitoring system that can gather data from any kind of source (not only mobile devices) for better assessing the behavior of the network. Furthermore, we also propose an actual model and possible directions for implementing such ideas in a real system. To the best of our knowledge, there is no solution currently that employs hybrid active/passive measurements and crowdsensing for monitoring TD.

We describe the proposed model in Subsection 6.1.1, and discuss it in Subsection 6.1.2.

6.1.1 The Model: How It Works

The model is divided in four components, as shown in Figure 6.1: measurement agents, continuous monitoring, storage, and data analyses. Each component has a specific purpose, described below.

The measurement agents can be embedded in virtually any hardware or software capable of making and reporting measurements. Examples include smartphones, tablets, personal computers, IoT sensors, middlewares, Cloud services, Virtualized Network Functions (VNFs), TD detection tools, among others. During the normal operation of such devices or applications, the agents collect measurements (passively and/or actively), TD inferences, and/or confounds. They may also perform active measurements when requested. Examples of traffic-related metrics are delay, loss rate, and throughput. Confounds are the factors that may affect the measurements (other than TD) and/or help characterizing

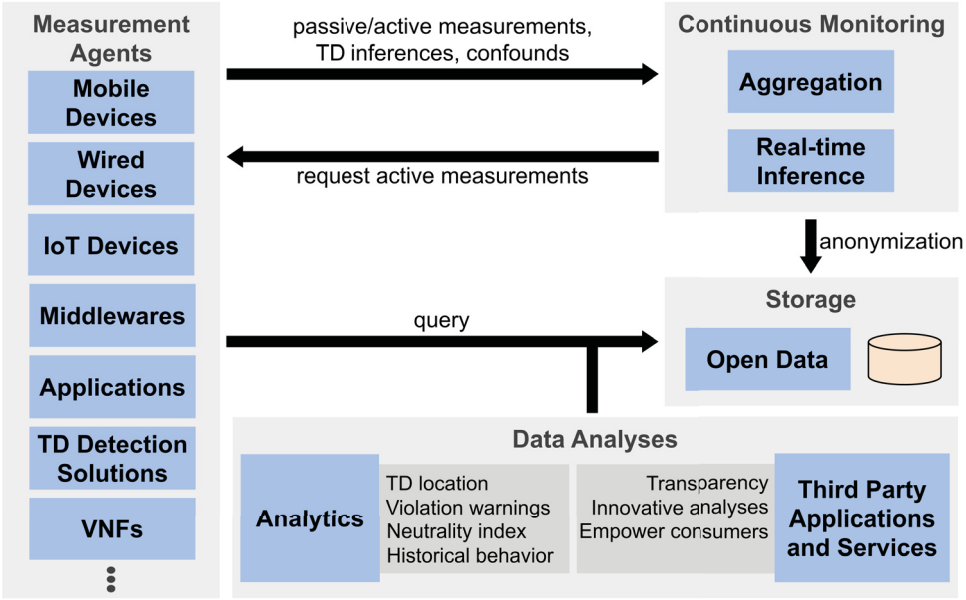


Figure 6.1: NN monitoring model.

them – such as geographic location, type of network (mobile or wired), ISP, application protocol, and time of day.

Different agents may have different sensing capabilities, due to differences in hardware or operating system features. If a measurement agent is embedded in a third-party application, for example, it may be able to report only measurements regarding the traffic of the application itself, since the application might not have enough permissions to measure all traffic that goes through the device on top of which it is running. On the other hand, if an agent is running in a personal computer with enough privileges, it may be able to make passive measurements regarding all the traffic in the device. Moreover, if a TD detection tool, such as those described in Chapter 3, is implemented as a measurement agent in our model, then the tool may report its TD detection results, instead of raw traffic measurements.

As measurements, inferences and confounds are reported by agents, they are aggregated and real-time TD inference is performed. If the presence of TD is detected, active measurements can be promptly requested to the appropriate agents for further investigating the suspicious case, and/or to locate where in the network the TD occurred. Note that active measurements may be requested to any agent, regardless of which agent reported the measurements that triggered the TD detection. Furthermore, there can also be measurement campaigns, in which active measurements are issued regardless of suspicions, configuring a more proactive approach instead of just reacting to potential cases of TD.

All data, both reported by the agents and inferred by the system, should be anonymized and stored in a database. This database should be publicly accessible through an Open Data API [85]. Furthermore, data should be distributed and replicated, in order to both increase its availability and protect it from any potential attack coming from those that might be compromised by the information. From the obtained data, the system can make more detailed and complex analyses. Participating applications and devices may benefit from these analyses at runtime, changing their behavior depending on how their traffic is being treated, for example. It is also possible for third parties to access the data and make their own use of it, expanding the capabilities of the model.

Examples of analyses that may be performed include: locating TD; warnings regarding NN violations; a "neutrality index", which indicates how neutral each ISP is; historical behavior of different ISPs or applications; and data mining, which can identify patterns regarding the deployment of traffic management techniques.

6.1.2 Discussion: Thoughts on the Model

Adopting a crowdsensing approach allows the model to take advantage of the large amount of devices already deployed in the wild. The model also allows any third-party application or service to benefit from the TD inference provided by the system. It is similar to a service-oriented approach, in which any agent can make use of the "TD inference service". Moreover, the data each agent contributes with is used to improve the overall effectiveness of the system for all participants. This aggregation of measurements from multiple vantage points may help not only to detect TD, but also to identify which ISP is doing that. Furthermore, this approach does not require control of end-hosts, since edge-devices and applications are external entities contributing willingly to the system.

The hybrid active/passive measurement strategy has a much lower overhead when compared to purely active strategies. By passively capturing measurements, it is possible to detect whether TD might be occurring and then trigger active measurements. Thus there is no need for generating a large amount of artificial traffic in order to saturate the network before taking measurements. Continuous monitoring also enables the detection of dynamic behaviors – such as an ISP employing TD only on specific periods of the day. The historical data obtained allows for deeper analyses regarding traffic management policies and TD patterns.

The proposed model makes no assumption regarding the network, TD mechanisms employed, applications being discriminated, or characteristics of the participating devices and applications. The measurement agents may be embedded in anything, such as edge-devices or even another system, which may be connected to any type of network (mobile or wired). This agnostic approach makes the model future-proof given the evolution of networks, devices and protocols. Specific characteristics of the agents and the network are reflected by the confounds during aggregation and real-time inference, and later analyses may also be performed considering such specific properties. Furthermore, any of the existing solutions for detecting TD may be employed, both for issuing active measurements and for the continuous monitoring, enabling the detection of several different types of TD.

By adopting the Open Data paradigm, this model not only helps ensuring innovation by monitoring NN compliance, but also creates new possibilities on its own for new innovative solutions. Third parties can create applications and services that make unforeseen uses of the data obtained by the system. Therefore, the crowdsensing approach allied with Open Data enables any consumer and/or innovator to contribute with a more transparent and innovative Internet.

6.2 AS-LEVEL ROUTING

In this section we present an overview of the AS-level routing properties that are assumed by the strategy for locating TD presented in this chapter. Data packets sent from one end-host in the Internet to another may traverse several ASes. The sequence of traversed ASes is called an AS-level path, which in this work we simply call a *path*. The path traversed by packets is defined by the ASes themselves. When a packet arrives, the AS

must decide to which neighbor AS the packet should be forwarded. This decision depends on the packet final destination and on the traffic exchange agreements the AS has with its neighbors.

The relationships between ASes can be abstracted into three types [54]: (i) customer-to-provider ($c2p$), or provider-to-customer ($p2c$) in the opposite direction; (ii) peer-to-peer ($p2p$); and (iii) sibling-to-sibling ($s2s$). An AS connects to another AS in order to gain access to other parts of the Internet which are not reachable from its own network or through its customers. In a $c2p$ relationship, a customer AS pays a provider AS for transit services, i.e., for access to part of the Internet. In a $p2p$ relationship, two ASes mutually exchange traffic without payments. This exchange may only occur between the two ASes themselves and their customers. In a $s2s$ relationship, the two ASes belong to the same organization, thus exchange traffic freely.

The Gao-Rexford model is widely accepted for describing paths in the Internet [55]. According to this model, a path between two ASes is defined as a sequence of ASes in which for every AS providing transit (a transit provider), there is a customer AS adjacent to the transit provider. Therefore, there is always an AS paying for the transit service. A path must thus have the following pattern: zero or more $c2p$ links, followed by zero or one $p2p$ link, followed by zero or more $p2c$ links. Moreover, any number of $s2s$ links may appear in the path. This pattern configures the *valley-free* property. A path that follows this property is a *valley-free path*, and a path that do not follow the property is a *valley path*.

Figure 6.2 shows an extract of a real AS-level topology with the corresponding relationships, inferred by CAIDA [154]. In the figure, the path *Copel* $\rightarrow$ *RNP* $\rightarrow$ *UFPR* is valley-free, since the transit provider (*RNP*) is being paid by its customer (*UFPR*). However, *Copel* $\rightarrow$ *Sercomtel* $\rightarrow$ *Level 3* is a valley path, since no one is paying the transit provider *Sercomtel*.

There may exist several possible valley-free paths between any two ASes. Depending on the traffic exchange agreements in place, any of the possible paths may be the actual path traversed by traffic [155]. Furthermore, the actual path traversed may change over time [109]. For instance, in the topology shown in Figure 6.2, paths *Sercomtel* $\rightarrow$ *Copel* $\rightarrow$ *Level 3* and *Sercomtel* $\rightarrow$ *ALGAR* $\rightarrow$ *Level 3* are both valley-free. In this particular case, the AS *Sercomtel* may prefer to exchange traffic with *ALGAR* through the $p2p$ link, since it would be cheaper than using the $c2p$ link with *Copel*.

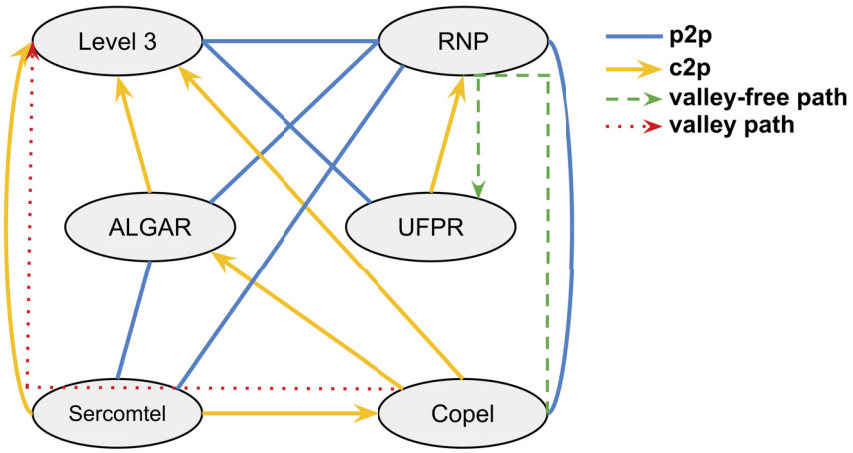


Figure 6.2: Extract of a real AS-level topology with the corresponding relationships, as well as valley and valley-free paths.

6.3 THE PROPOSED STRATEGY FOR LOCATING TD

In this section, we propose a strategy for combining several TD detection measurements in order to locate TD in the Internet. The main idea is to first rule out the ASes that are not employing TD until a remaining AS is left that can be identified as the source of TD. Our proposal takes advantage of inter-domain routing policies, in particular the valley-free property, to carefully select measurement points that help identify these TD sources.

The rest of this section is organized as follows. First, we describe our assumptions in Subsection 6.3.1. Next, we present our notation in Subsection 6.3.2. Subsection 6.3.3 describes then how valley-free paths between ASes are obtained. Finally, we describe the proposed strategy in Subsection 6.3.4.

6.3.1 Assumptions

Our proposal relies on five assumptions. We assume that the AS-level topology of the Internet is known, along with the relationships between ASes. Several datasets that infer the AS-level topology of the Internet are available, mainly based on BGP routing tables. In this work, we employ the AS relationships inferred by CAIDA in their AS Rank project [154]. We also assume that the valley-free property is valid, which is considered a fundamental BGP routing policy [156].

The availability of an end-to-end TD detection solution for checking the presence of TD between two specific end-hosts is also assumed. Traffic monitoring tools that employ statistical properties to detect TD are well known, several solutions exist as we describe in Chapter 3. We proposed a holistic TD detection solution described in Chapter 5. Another assumption is that we are able to execute a TD detection solution from/to some set of ASes – the so called *measurement ASes*. This can be done by having access to end-hosts connected to those ASes, or by having an accessible Virtualized Network Function (VNF) deployed within their networks.

Finally, we assume that if an AS discriminates some type of traffic, this discrimination will occur regardless of the origin and/or destination of the traffic. Therefore, if an AS discriminates a specific application, all traffic from that application will be affected, regardless of where it is coming from or going to. Note that in this work we consider only TD triggered by application, not TD triggered by path.

6.3.2 Notation

Before describing our proposal for locating TD, we first present our notation. We model the AS-level topology of the Internet as a directed graph $N = (A, L, f)$. Let A be the set of ASes in the network. Let L be the set of links between ASes. Let $R = \{c2p, p2c, p2p, s2s\}$ be the set of possible relationships between ASes. Let $f : L \rightarrow R$ be the function that maps a link $l \in L$ to the corresponding relationship $r \in R$.

A path $p = \{u, ..., v\}$ is a sequence of ASes connecting u and v. Let $P_{u,v}$ be the set of all paths between ASes u and v. Furthermore, L_p is the sequence of links of a path $p \in P_{u,v}$, and $R_p = \{f(l) : l \in L_p\}$ is the sequence of relationships between the corresponding ASes. A path $p \in P_{u,v}$ is valley-free if R_p follows the valley-free property as described in Section 6.2. The set of all valley-free paths between two ASes u and v is denoted by $V_{u,v} \subseteq P_{u,v}$.

We define three possible behaviors for a given AS regarding TD: *neutral*, *discriminatory*, and *unknown*. A *neutral* AS is not employing TD, while a *discriminatory* AS is.

An *unknown* AS has no inferred behavior. Let I_u be the inferred behavior of AS $u \in A$. Similarly, a pair of ASes (u, v), $u, v \in A$ may be *neutral* or *discriminatory*. For a *neutral* pair of ASes no end-to-end TD was detected between them, but if TD has been detected, then the pair is *discriminatory*. Let $I_{u,v}$ be the inferred behavior of the pair of ASes (u, v).

6.3.3 Searching Valley-free Paths

The proposed strategy relies on checking valley-free paths between ASes. In order to search for these paths we employ a modified breadth-first search, which discards paths that contain "valleys". In order to keep the search feasible, parameter σ is employed to establish a limit of the maximum path size with respect to the corresponding shortest valley-free path. In other words, we always search for valley-free paths with sizes that are at most σ links larger than the shortest valley-free path. Note that in the Internet real paths employed are often larger than the shortest possible path, as we observed in the experiments described in Section 6.4.

Let $p \in V_{u,v}$ be the shortest valley-free path between ASes u and v. We define $V^{\sigma}_{u,v} = \{p' : p' \in V_{u,v}, |p'| \leq |p| + \sigma\}$ as the set of valley-free paths between ASes u and v with size not larger than the size of the shortest path plus σ.

6.3.4 Locating TD

The proposed strategy for locating TD in the Internet consists of 5 steps: initialization, AS pair selection, TD detection, inference, and completion. An overview of our strategy is shown in Figure 6.3. In the initialization, our solution receives as input the AS-level topology graph, the pair of end-hosts between which TD will be located, and the set of measurement ASes. Based on the input, the initial set of suspects is created, i.e., the ASes suspected to be *discriminatory*. In the next step, AS pair selection, the pairs of measurement ASes from which probes will be issued are selected. The probes are effectively issued in the TD detection step, and results are examined in the inference step. If TD is located, or all suspects are classified as *neutral*, or if there is no more AS pair to be measured, the TD locating process finishes on the completion step, returning the output. If none of these conditions are met, the process returns to the AS pair selection step. We further describe each step below.

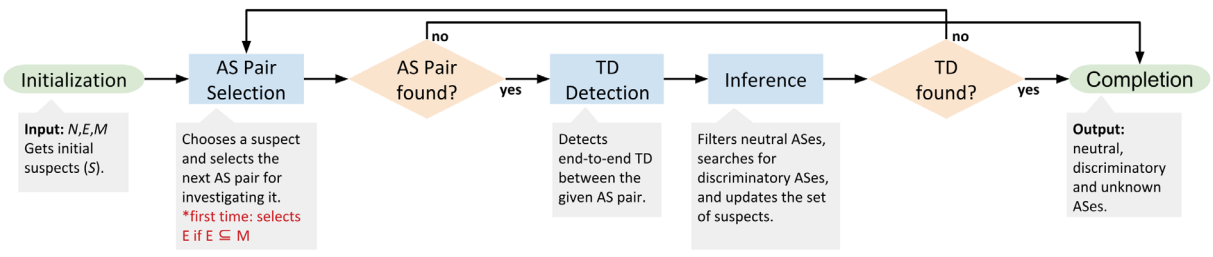


Figure 6.3: An overview of the proposed strategy for locating TD.

Initialization: The proposed strategy receives as input: (i) the graph $N = (A, L, f)$ representing the AS-level topology of the Internet; (ii) a pair $E = (e_1, e_2)$, $e_1, e_2 \in A$ of initial ASes between which TD will be located; and (iii) a set $M \subseteq A$, which are the ASes available to perform measurements from.

In this step, the set of suspects S is initialized. These are all the ASes in the valley-free paths between the initial pair E ($V^{\sigma}_{e_1,e_2}$). If an AS is doing TD, it is one of these ASes. Thus the behavior of all ASes in S is initialized as *unknown*.

Figure 6.4 shows an example of an initial pair $E = (e_1, e_2)$ and all the possible valley-free paths $V^{\sigma}_{e_1,e_2}$, $\sigma = 0$. In this example, there are two possible paths between the initial pair: $\{e_1, c_1, c_3, e_2\}$, and $\{e_1, c_2, c_3, e_2\}$. Since we do not know which path would be effectively traversed by traffic transmitted between e_1 and e_2, we consider all the possible paths. In this example, the set of initial suspects would be all the ASes shown in the figure, i.e., $S = \{e_1, e_2, c_1, c_2, c_3\}$ – any one of them could be the responsible for TD.

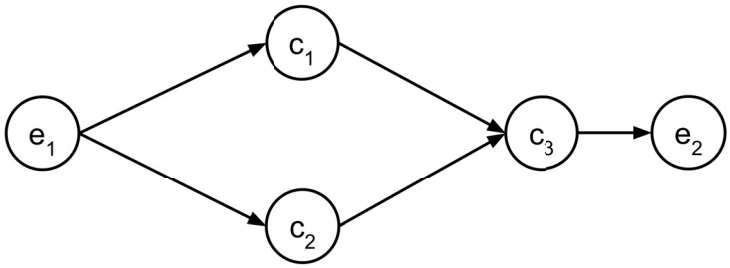


Figure 6.4: Example of an initial pair $E = (e_1, e_2)$, and the possible valley-free paths between e_1 and e_2 ($V^{\sigma}_{e_1,e_2}$).

AS Pair Selection: In this step a pair of measurement ASes that will help infer the behavior of a suspect is selected. Thus, this step starts by selecting one suspect AS $s \in S$ to be investigated. Then, a search is executed for a measurement pair $W = (u, v)$, $u, v \in M$ for which *all* valley-free paths $p \in V^{\sigma}_{u,v}$ traverses the suspect s, i.e., $\forall p \in V^{\sigma}_{u,v}, s \in p$. The strategy is to filter the *neutral* ASes until only the *discriminatory* AS is left. Therefore, we look for pairs for which the possible paths are guaranteed to traverse a particular AS, since that may either eliminate or confirm the suspicions about that AS, as we describe below in the inference step. The first time this step is performed, if the initial pair $E = (e_1, e_2)$ is available for measurement ($E \subseteq M$) then E is selected.

The suspect $s \in S$ to be investigated in this step is the one that appears less times in the paths between *discriminatory* pairs. Remember, the idea is to identify and eliminate neutral ASes. We take all discriminatory pairs $W' = (u, v)$, $u, v \in M$, $I_{u,v} = \textit{discriminatory}$ that were already measured in the TD detection step. Then, we count how many times each suspect s is present in all the possible paths $V^{\sigma}_{u,v}$. The suspect that appears less times is selected to be investigated. Let i be the selected suspect. The rationale is that suspects that appear less times in the paths between ASes for which TD was detected are less likely to be *discriminatory*. Thus they are selected first. However, if no *discriminatory* pair has been found yet, the first suspect in S is selected.

After selecting the suspect i, we search for another pair of measurement ASes $W = (u, v)$, $u, v \in M$ that has not been selected previously and satisfies the criterion above, i.e., $\forall p \in V^{\sigma}_{u,v}, i \in p$. We limit this search with parameter δ, which sets the maximum valley-free distance from i up to the limit at which measurement ASes are checked. Therefore, the proposed strategy tries to form an AS pair W starting from the measurement ASes closer to i, up to the measurement ASes that are at distance δ to i. The valley-free property makes this search computationally feasible, since it limits the possible paths between ASes.

For instance, if i itself is available for measurement ($i \in M$), then pairs using i (distance 0) and each measurement AS up to distance δ will be formed. But if $i \notin M$,

then we try to form a pair W first with measurements ASes at distance 1 from i, then up to distance 2, and so on, up to distance δ.

Figure 6.5 shows an example using the same initial pair E from Figure 6.4. In this example, $\delta = 2$. There are four measurement ASes within distance 2 of the suspect i (the one chosen to be investigated): $m_1, m_2, m_3, m_4 \in M$. The pair (m_1, m_2) follows the criteria described above and could be selected for investigating i, since all possible paths between them traverses i. However, pair (m_3, m_4) would not be selected by our strategy, since there is a possible path between them that does not traverse i, which is $\{m_3, c_4, e_2, m_4\}$.

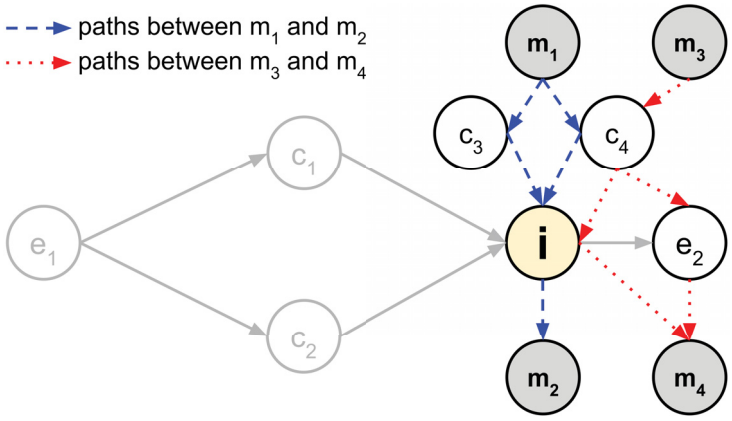


Figure 6.5: Example of AS pair selection for investigating i: pair (m_1, m_2) can be selected, while pair (m_3, m_4) can not.

After all possible pairs for a suspect i have been already checked, the next suspect is chosen. If there are no more suspects, the TD locating process is finished.

TD Detection: In this step, the presence of TD between the selected measurement pair $W = (u, v)$, $u, v \in M$ is assessed, following an end-to-end fashion. As mentioned above, we assume a solution for detecting TD between two end-hosts in the Internet is being used. There are several such proposals in the literature, as we describe in Chapter 3, and we proposed another solution described in Chapter 5. We assume that one of these solutions is employed in this step: it is executed on end-hosts connected to the ASes u and v in order to detect TD between these two ASes. If TD is detected between u and v, we set the corresponding behavior $I_{u,v}$ to *discriminatory*. Otherwise $I_{u,v} = neutral$. The type of TD that can be located by our proposal is dictated by which TD detection solution is employed, since different solutions detect different types of TD, as mentioned in Chapter 3. However, this proposal considers only TD triggered by application.

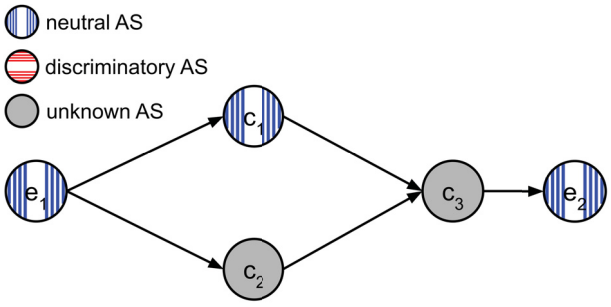
Inference: In this step, the inferred behaviors of the AS pairs measured in the TD detection step are combined. The idea is to filter the suspects, eliminating *neutral* ASes until there is only one suspect remaining, that could thus be deemed the responsible for employing TD. The rationale is that while there are two or more *unknown* ASes in the same set of paths $V^{\sigma}_{u,v}$, it is not possible to infer which one is practicing TD, since we do not know which of them were actually traversed by the TD detection traffic.

Inference is done in three parts. First, the *neutral* pairs of ASes are examined, to search for *neutral* ASes. Then, the *discriminatory* pairs of ASes are examined to search for *discriminatory* ASes. Finally, the set of suspects S is updated.

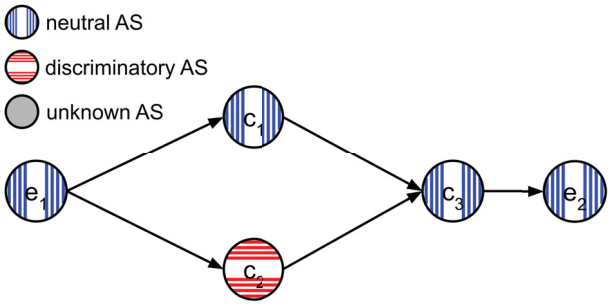
In the first part, for each *neutral* pair $W = (u, v)$, $u, v \in M$, $I_{u,v} = neutral$, measured in the TD detection step, we search for the set $T \subset A$ of ASes that are present in all valley-free paths $p \in V^{\sigma}_{u,v}$. These ASes are guaranteed to have been traversed by the TD detection traffic, since they are in all possible paths. If any of these ASes was employing TD, $I_{u,v}$ would have been set to *discriminatory*, given the assumptions. We thus set the behavior of all ASes $a \in T$ to *neutral* – this includes at least u, v, and the suspect i for which the pair W was selected in the AS pair selection step. For instance, in the example shown in Figure 6.5, if TD was not detected for the pair (m_1, m_2) ($I_{m_1,m_2} = neutral$), then i would be inferred as *neutral* ($I_i = neutral$).

Then, for each *discriminatory* pair $W' = (u, v)$, $u, v \in M$, $I_{u,v} = discriminatory$, we take all valley-free paths $p \in V^{\sigma}_{u,v}$ and remove the *neutral* ASes from those paths, resulting in the set of paths $V'^{\sigma}_{u,v}$. If there is a single AS d left in all non-empty paths $p' \in V'^{\sigma}_{u,v}$, then the inferred behavior of d is set to *discriminatory*. The rationale is that all other suspects were found to be *neutral*, so the only remaining AS that could have been responsible for TD is d. If there is more than one AS left in the paths $V'^{\sigma}_{u,v}$, their behavior is set to *unknown*.

Figure 6.6 shows two examples of possible inference situations, using the same initial pair E as in Figure 6.4. Let us suppose in these examples that TD was detected between the initial pair E. In Figure 6.6(a), ASes e_1, e_2 and c_1 were inferred *neutral*. Since there are two other ASes, c_2 and c_3, through which traffic may have traversed, it is not possible to know which one of them is responsible for TD. Therefore, both c_2 and c_3 remain *unknown*. However, let us suppose then that c_3 was inferred to be *neutral*. As shown in Figure 6.6(b), the only remaining *unknown* AS would be c_2, thus we infer $I_{c_2} = discriminatory$.



(a) More than one *unknown* ASes remains, thus it is not possible to infer the *discriminatory* AS.



(b) Only one *unknown* AS remains, thus it is possible to infer the *discriminatory* AS.

Figure 6.6: Examples of inference between the *discriminatory* pair E.

We then update the set of suspects S, removing all ASes that were inferred *neutral*, and adding the new *unknown* ASes found in the valley-free paths between *discriminatory*

pairs. Next, if an AS in the paths between the initial pair E is inferred to be *discriminatory*, then TD is found and the TD locating process finishes. Moreover, if all ASes in the paths between E are inferred to be *neutral*, then there is no TD to be found and the process finishes. Otherwise, if TD was not found and there are still *unknown* ASes among those between E, the process returns to the AS pair selection step.

Completion: The TD locating process may complete under three conditions: (i) a *discriminatory* AS between the initial pair E is found; (ii) all ASes between E are classified as *neutral*; or (iii) there are no more measurement AS pairs available. In the first two cases, the process is considered to have finished successfully, while in the last case the process did not succeed. The output consists of three sets: *neutral* ASes, *discriminatory ASes*, and *unknown ASes*.

6.4 EVALUATION: AS-LEVEL GRAPH AND PATHS

In this section, we first briefly describe in Subsection 6.4.1 the AS-level topology graph we employ in the evaluations presented in this chapter, as well as the dataset from which the graph was built. Then we present in Subsection 6.4.2 an experiment executed on the PlanetLab global testbed, which had the purpose to check the assumptions related to Internet routing; this experiment also confirms the limitations of *traceroute*-like techniques.

6.4.1 AS-level Topology Graph

The AS-level topology graph employed in our evaluations was built from the dataset published by CAIDA within their AS Rank project [154]. This dataset contains, among other data, information about the relationship between numerous ASes, inferred based on BGP data [54]. However, some ASes in the dataset have no relationship with other ASes. We thus ignored those ASes in our evaluations. We employed the October 2018 dataset, which contains 86622 different ASes, from which 24815 have no inferred relationships with other ASes. 61807 ASes are thus considered in our evaluations.

We employ in our evaluations the betweenness centrality metric, which measures to which extent a vertex is present in the shortest paths between all other pairs of vertices. To be precise, the betweenness of a vertex is the sum of the fractions of shortest paths between all other pairs of vertices in which the vertex is present [157]. We also computed a property that we call *valley-free betweenness centrality*, computed taking into account only the shortest valley-free paths. The rationale is that, since our proposal searches for measurement ASes that are in paths traversing certain ASes (the suspects), the betweenness centrality may be a good indication of how effective an AS is to be used for measurements – ASes with higher betweenness belong to more paths, therefore are more likely to be selected as a measurement AS. In this chapter, we present results for both the valley-free betweenness centrality and the standard betweenness centrality metrics.

6.4.2 AS-level Paths in the Internet

In order to evaluate the assumptions regarding the AS-level paths and also to assess the limitations of *traceroute*-like techniques, we conducted an experiment on the PlanetLab global testbed. In this experiment, we measured the AS-level paths between 29 PlanetLab hosts and a large amount of Internet IP prefixes. We employed the list of Internet prefixes and corresponding ASes published in May 2018 by CAIDA [158]. Several ASes control

more than one prefix. In these cases, one prefix was chosen for each AS. Furthermore, some ASes from the prefix list were not present in the AS-level topology graph (described above), and were thus discarded. The resulting list contained 60578 prefixes/ASes.

From each PlanetLab host, we continually measured the paths to all prefixes from our list using the *traceroute* tool. The experiment took place from January 10, 2019 to February 1, 2019, totaling 22 days of measurements. For each measurement obtained, we mapped the IP addresses to the corresponding ASes, using the list of prefixes from CAIDA. Thus, we converted the host-level paths acquired by *traceroute* to AS-level paths. However, it is common for some hosts not to reply *traceroute* probes, or to reply with an invalid IP address. In such cases, we cannot know that the corresponding AS is in the path, unless another host within the same network replies to another probe during the same measurement.

We then classified all paths measured as *valley*, *valley-free*, or *unknown*. Paths that follow the valley-free property in the graph are classified as *valley-free*, and otherwise they are classified as *valley*. A portion of the measured paths presented measurement errors, as described above. These errors resulted in incomplete paths: for some hosts of these paths the corresponding ASes were missing. Whenever ignoring these errors caused the resulting path to be valley-free, then it was classified as *valley-free*: in those cases, we considered that another host of the same AS replied correctly. Otherwise, paths are classified as *unknown*, since we failed to obtain the complete set of ASes and thus cannot know the actual classification. We excluded from our results the paths that contained links not in the graph.

A total of 75,597,104 *traceroute* measurements were issued, but 1,801,089 were excluded due to missing links (2.38%). From the remaining 73,796,015 measurements, 55.34% (40,837,151, more than half) resulted in *unknown* paths, which clearly shows the limitation of measuring paths with the *traceroute* tool. 44.31% (32,703,036) of the measurements resulted in *valley-free* paths, the vast majority of measurements that were not *unknown*, while 0.35% (255,828) of the measurements resulted in *valley* paths. The *valley-free* paths reached 48283 different ASes (79.7% of all prefixes measured).

We also evaluated the sizes of the measured valley-free paths, taking into account parameter σ. We compared the size of measured valley-free paths with the size of the corresponding shortest paths in the graph. In our experiment 55.78% of valley-free measurements corresponded to shortest paths, while 31.87% traversed paths with one more edge in comparison with the corresponding shortest paths, and 10.34% were two edges larger.

6.5 EVALUATION: LOCATING TD

In this section we present results from the empirical evaluation of the proposed strategy to locate the source of TD in the Internet. We simulated our strategy for locating TD under several different scenarios. In each scenario, different sets of initial pairs (the two ASes between which TD should be located) and measurement ASes (the ASes available for measurement) are employed. These simulations have three main goals: (i) to evaluate if the proposed strategy is capable of locating TD; (ii) to identify which ASes are better for issuing measurements from, in different situations; and (iii) to identify between which pairs of ASes it is easier to locate TD. We employ three main criteria for comparing sets of measurement ASes: (i) the success rate, which expresses the portion of the simulations within a scenario in which TD was successfully located; (ii) the average number of measurements across

all simulations within a scenario; and (iii) the number of ASes available for measurement. The optimal set of measurement ASes is the one that achieves the largest success rate, issuing the least amount of probes, and containing the least amount of ASes available for measurement. The rationale is that it may not be feasible to have access to a large number of different ASes. Furthermore, issuing a large amount of probes presents an overhead to the network.

The rest of this section is organized as follows. We first describe the setup of our simulations in Subsection 6.5.1: we describe the implementation, execution, as well as the sets of measurement ASes and initial pairs employed. Next, we describe the parameters employed in the simulations, and how we chose their values in Subsection 6.5.2. Then, Subsection 6.5.3 presents results comparing several different sets of measurement ASes, while Subsection 6.5.4 compares different sets of initial pairs. We then present results with different assumptions than the results presented in the other subsections. The results presented in Subsection 6.5.5 do not assume that the initial pairs are available for measurement, while the results presented in Subsection consider paths larger than the shortest paths. Finally, we discuss the obtained results in Subsection 6.5.7.

6.5.1 Simulations Setup

We defined several simulation scenarios. In each simulation scenario, a given set $Z = \{(u_1, v_1), (u_2, v_2), ..., (u_n, v_n)\}$, $u_i, v_j \in A$ of initial pairs, and a given set $M \subseteq A$ of measurement ASes are received as input. In each experiment for a given scenario, several simulations are executed. For each initial pair $E = (e_1, e_2)$, $E \in Z$ we take all ASes present in the possible valley-free paths between E ($V^{\sigma}_{e_1,e_2}$). Let X_E be the set of such ASes. For each AS $k \in X_E$, we execute a simulation in which k is the AS responsible for TD. The simulation is successful if TD is located. Furthermore, we also execute a simulation with no AS employing TD, in which case the simulation is successful if all ASes $u \in X_E$ are classified as *neutral*. Therefore, a scenario results in $\sum_{E \in Z}(|X_E| + 1)$ simulations.

Each simulation receives as input (N, E, M), i.e., graph N, an initial pair of ASes E, and a set of measurement ASes M. We employ the AS-level topology graph built from the CAIDA dataset, as described in Section 6.4. We assume at first that on each simulation the ASes in E are also available for measurement, in addition to M – for that particular simulation. We also present results without this assumption later in Subsection 6.5.5. Furthermore, to run a simulation, an end-to-end TD detection solution is required, as per our assumptions. We simulate the TD detection with an "oracle" detector, which we describe below. Thus, our simulations employ an actual implementation of the proposed strategy (written in C++), and make use of a simulated TD detector (the oracle) instead of generating real measurement traffic. The rationale is that our goal is to evaluate only the proposal for locating TD. We do not have access to a large number of ASes for issuing measurements. Therefore, simulating the TD detection allows us to try to locate TD between any pair of ASes in the Internet.

The oracle detector receives as input two ASes u and v, $u, v \in A$, and returns the inferred behavior $I_{u,v}$, which may be *neutral* or *discriminatory*. The oracle works by checking if AS $k \in X_E$ (the AS responsible for TD in each simulation) is in any valley-free path between u and v ($V^{\sigma}_{u,v}$). If k is not present in any path $p \in V^{\sigma}_{u,v}$, then the oracle returns *neutral*, since traffic between u and v does not traverse k and thus cannot be discriminated by k. Otherwise, if k is in at least one path p, the oracle assumes the worst case, which corresponds to traffic traversing the path containing k, and thus returns as

output the classification *discriminatory*. In the case of simulations with no TD, the oracle always returns *neutral*.

The sets Z and M were built based on metrics extracted from the graph, as well as on the classification of ASes available on the PeeringDB website [159]. PeeringDB is an online database where operators contribute information about their networks. According to the authors in [160], the amount of ASes registered on the website as transit, access and content providers is representative of the corresponding sets of these types of ASes in the Internet. We obtained the list of all ASes of these types from PeeringDB in June 20th, 2019. Furthermore, we listed ASes from the graph based on their degree, betweenness centrality, and valley-free betweenness centrality. Table 6.1 shows the sets of ASes selected. The columns of the table indicate for each set: name, description, and number of ASes. The first three sets were taken from the PeeringDB website. The last three sets consist of the n ASes with the largest values for the corresponding metrics. The values of n we employed were: 10, 50, 100, 500, and 1000.

Table 6.1: Selected Sets of ASes

Name	**Description**	**Size**
pdb-access	Access providers from PeeringDB	5263
pdb-content	Content providers from PeeringDB	1462
pdb-transit	Transit providers from PeeringDB	2293
degree-eq-1	ASes with degree 1 in the graph	21220
degree-le-2	ASes with degree ≤ 2 in the graph	41247
degree-top-n	ASes with the largest degree	n
vfbet-top-n	ASes with the largest valley-free betweenness centrality	n
bet-top-n	ASes with the largest betweenness centrality	n

We created six sets of initial pairs, shown in Table 6.2, using the sets of ASes described above. Each of these sets contains 1000 different pairs of ASes. The table also shows the total number of simulations executed on scenarios employing each set. The set *pdb-a2a* contains 1000 pairs randomly selected from the ASes in the set *pdb-access*, i.e., from all the possible pairs between access providers (from PeeringDB), we randomly picked 1000 pairs. This set represents a common situation in the Internet: two end-hosts, connected to access providers, communicating with each other, such as in a P2P application. Similarly, the sets *pdb-c2c* and *pdb-t2t* are composed with ASes from the sets *pdb-content* and *pdb-transit*, respectively. Moreover, the *pdb-a2c* set contains 1000 pairs randomly selected in such a way that one of the ASes in each pair is from the *pdb-access* set and the other from the *pdb-content* set. This represents another common situation: an end-user accessing a content provider, such as a video streaming service. Similarly, the sets *pdb-a2t* and *pdb-c2t* are composed with access/transit providers and content/transit providers, respectively.

6.5.2 Parameters

In our simulations, we employed two extra parameters for selecting AS pairs: *mp* and *mt*. Parameter *mp* is the maximum number of AS pairs that can be selected to investigate a suspect. Thus, if *mp* AS pairs have been already checked to investigate some suspect, that suspect will no longer be investigated. *mt* is the maximum number of times the proposed solution tries to form a pair with a given measurement AS when investigating a suspect.

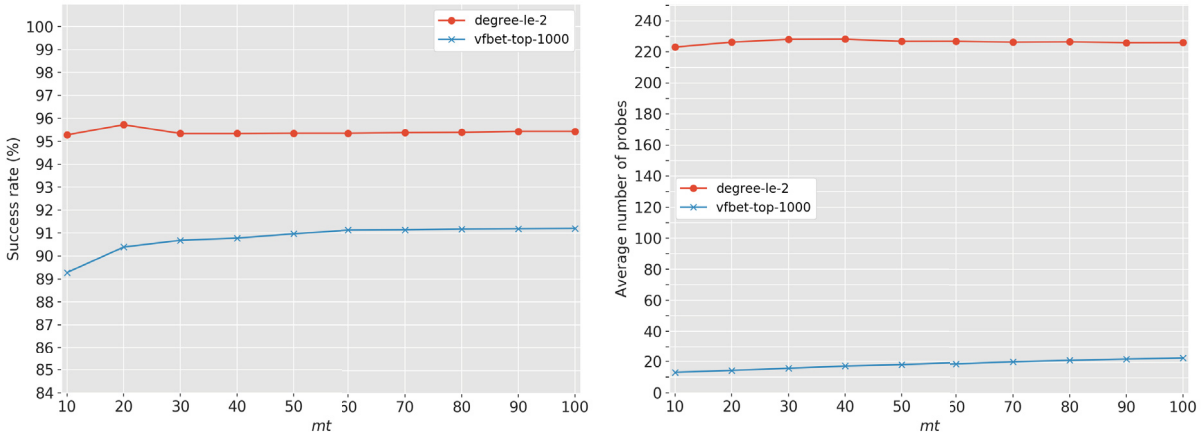
Table 6.2: Sets of Initial Pairs

Name	Pair Composition	Size	Simulations
pdb-a2a	Access providers	1000	7818
pdb-c2c	Content providers	1000	7229
pdb-t2t	Transit providers	1000	7168
pdb-a2c	Access and content providers	1000	7807
pdb-a2t	Access and transit providers	1000	7609
pdb-c2t	Content and transit providers	1000	7295

Therefore, if for mt times the paths between pairs containing a same AS do not all traverse the suspect, we no longer try to form measurement pairs using that AS for the suspect under investigation. These parameters limit the search space to select AS pairs, making it feasible to execute a large number of simulations.

In order to choose the values for the parameters mp and mt, we run several simulations employing different values for them. In these simulations, we employed a set of initial pairs containing 1000 pairs selected randomly from all ASes in the graph. We employed two different sets of ASes as the measurement ASes M: *degree-le-2* and *vfbet-top-1000*. As we describe later in this section, these two sets presented the best results overall.

First, we ran several sets of simulations employing a fixed large value for mp, and several different values for mt. We employed $mp = 100$, and mt ranged from 10 to 100, in increments of 10. For each value of mt, 8479 simulations were executed. Figure 6.7 shows the results obtained in these simulations. The success rate achieved by each set of measurement ASes for each value of mt is shown in Figure 6.7(a), while Figure 6.7(b) shows the average number of probes issued when using each set and for each value of mt. It is possible to see that both the success rate and the average number of probes did not vary much as the value of mt increased. We chose $mt = 20$ for our simulations, which is the value for which the success rate had the largest increment for both sets of measurement ASes. Therefore, we discard a measurement AS after 20 attempts when searching for AS pairs for each suspect.

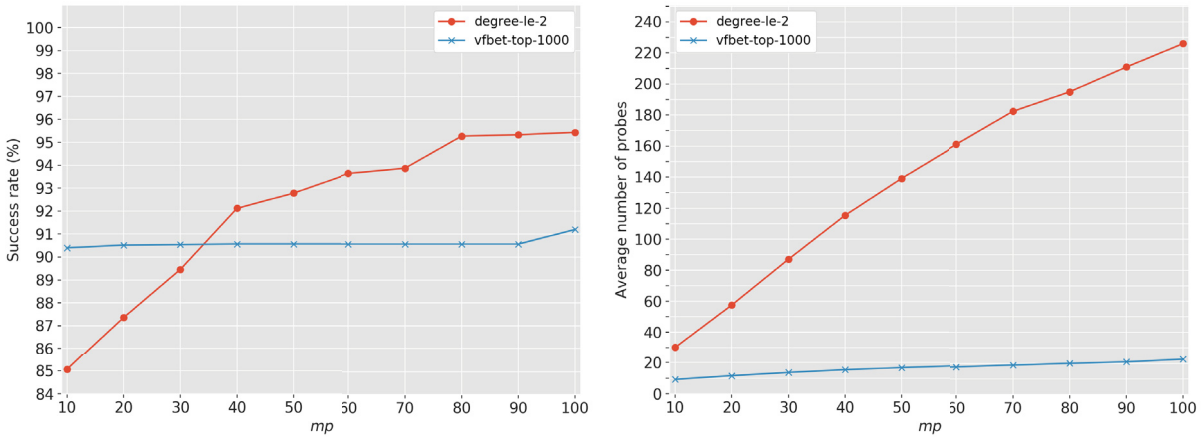


(a) Success rate

(b) Average probes

Figure 6.7: Success rates and average probes achieved by the sets of measurement ASes *degree-le-2* and *vfbet-top-1000*, for $mp = 100$ and mt ranging from 10 to 100.

Next, we ran simulations with $mt = 100$, and mp ranging from 10 to 100. Figure 6.8 shows the results obtained in these simulations. The success rate achieved by each set of measurement ASes for each value of mp is shown in Figure 6.8(a), while Figure 6.8(b) shows the average number of probes issued when using each set and for each value of mp. The success rate and average probes for the *vfbet-top-1000* did not increase much as the value of mp increased. However, for the *degree-le-2* set, both the success rate and average probes increased significantly. We chose $mp = 40$, since larger values would significantly increase the search space, and thus also the execution times, but did not achieve significantly better results. Therefore, in our simulations, up to 40 AS pairs are selected for each suspect.



(a) Success rate

(b) Average probes

Figure 6.8: Success rates and average probes achieved by the sets of measurement ASes *degree-le-2* and *vfbet-top-1000*, for $mt = 100$ and mp ranging from 10 to 100.

In addition to parameters mp and mt, the proposed strategy for locating TD also employs two other parameters: δ and σ. We set $\delta = 2$, thus only measurement ASes up to 2 hops away from the suspects are considered. Larger values would significantly increase the search space, since a large portion of the graph would be in distance 3 or more from the suspects. Furthermore, as we observed in the results presented later in this section, measurement ASes farther from the suspects are rarely selected. Figure 6.9 shows the CDF of the valley-free distances for all pairs of ASes in the graph. The figure shows, for each value of distance, the rate of pairs of ASes that are up to that distance from each other. For instance, about 5% of all pairs of ASes in the graph are up to 2 hops away from each other. For distance up to 3, the rate raises to about 35% of AS pairs.

We employ $\sigma = 0$ in most results presented in this section, thus we examine only the shortest valley-free paths between ASes. We do, however, present results for $\sigma = 1$ in Subsection 6.5.6, since paths one link larger than the shortest path are common in the Internet, according to the experiments described in Section 6.4.

6.5.3 Results: Comparing Measurement ASes

We first describe results for the comparison of the three metrics: degree, betweenness and valley-free betweenness centrality. These metrics were employed to build measurement sets M *degree-top-n*, *bet-top-n* and *vfbet-top-n*, respectively – $n \in \{10, 50, 100, 500, 1000\}$. Figure 6.10 shows the success rate achieved for each size and type of set, on scenarios with Z = *pdb-a2a*. For all sizes, sets *degree-top-n* achieved the smallest success rates,

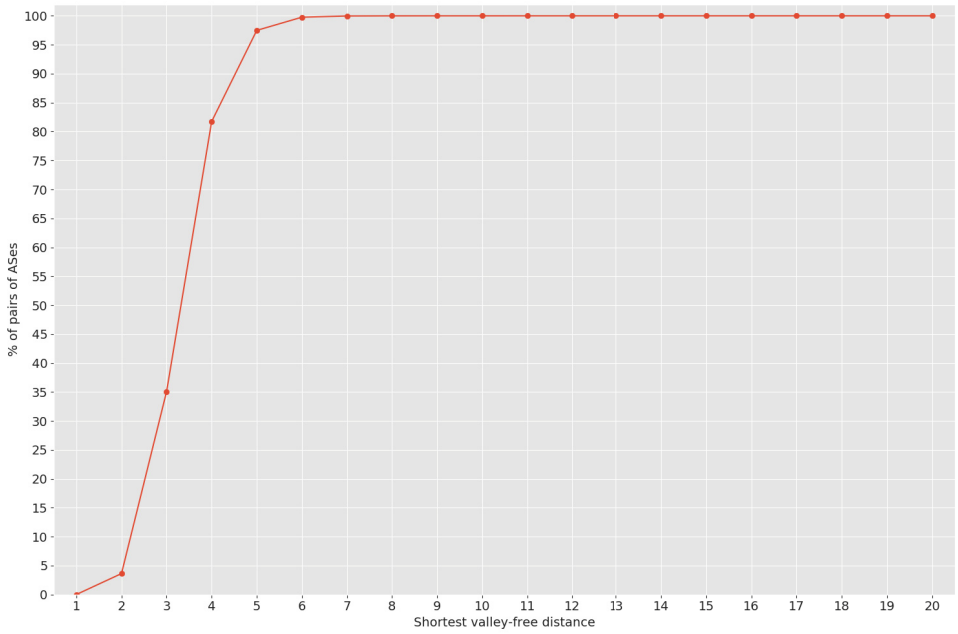


Figure 6.9: CDF of the valley-free distances.

while *vfbet-top-n* achieved the highest, ranging from 29% for *vfbet-top-10* to 93% for *vfbet-top-1000*. Since the *vfbet-top-1000* set achieved the best success rate, we will show no more results for the other sets in this work.

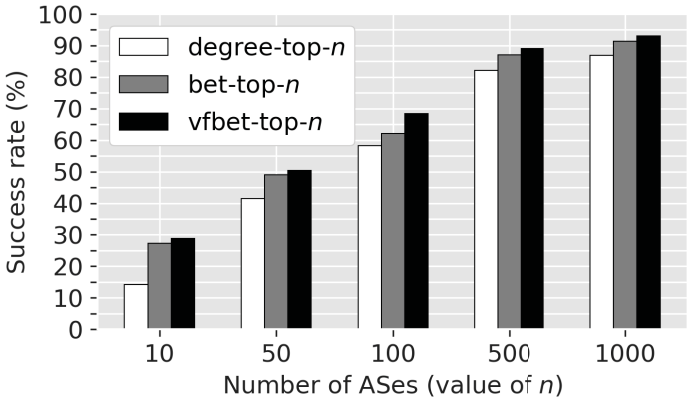
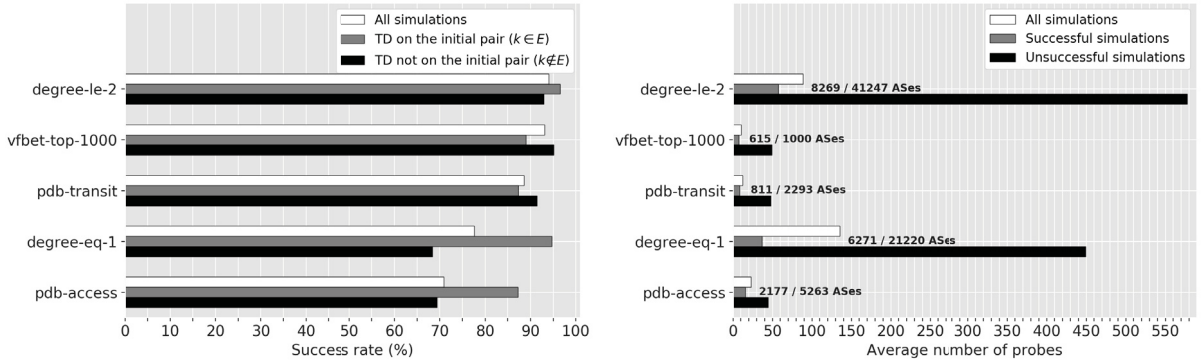


Figure 6.10: Success rates for Z = *pdb-a2a*, and varying sizes of *degree-top-n*, *vfbet-top-n* and *bet-top-n* as M.

ASes in the *vfbet-top-n* sets are generally closer to the suspects when compared to ASes in *degree-top-n* and *bet-top-n* sets. For instance, the average valley-free distance to the suspects from the measurement ASes selected from *vfbet-top-1000* was 0.79, which was slightly smaller than for the other sets: for *degree-top-1000* it was 0.87, and 0.85 for *bet-top-1000*. There are usually less paths and less ASes between the selected measurement pairs from *vfbet-top-1000*, and thus the *discriminatory* AS k appears less often in these paths. When k is not in a path, no TD is detected for that pair, and thus the suspect is inferred as *neutral*. Otherwise, another pair has to be selected for the same suspect, since the previous measurement pair did not help filtering the suspect.

Next, we evaluate the following sets of ASes as the measurement set M: *degree-eq-1, degree-le-2, pdb-access, pdb-transit*, and *vfbet-top-1000*. Figure 6.11 shows results for each of these sets in scenarios with Z = *pdb-a2a*. Figure 6.11(a) shows the success rates achieved by each set M considering all simulations, simulations in which the *discriminatory* AS k was in the initial pair E, and simulations in which k was not in E. Figure 6.11(b) shows the average number of probes (requests to the oracle) for all simulations, including those that were successful and those that did not succeed. The values beside each set of bars in the Figure indicate the number of different ASes selected for measurement across all simulations for the corresponding set M, as well as the total number of measurement ASes available.



(a) Success rate

(b) Average probes

Figure 6.11: Success rates and average probes for different sets M, and Z = *pdb-a2a*.

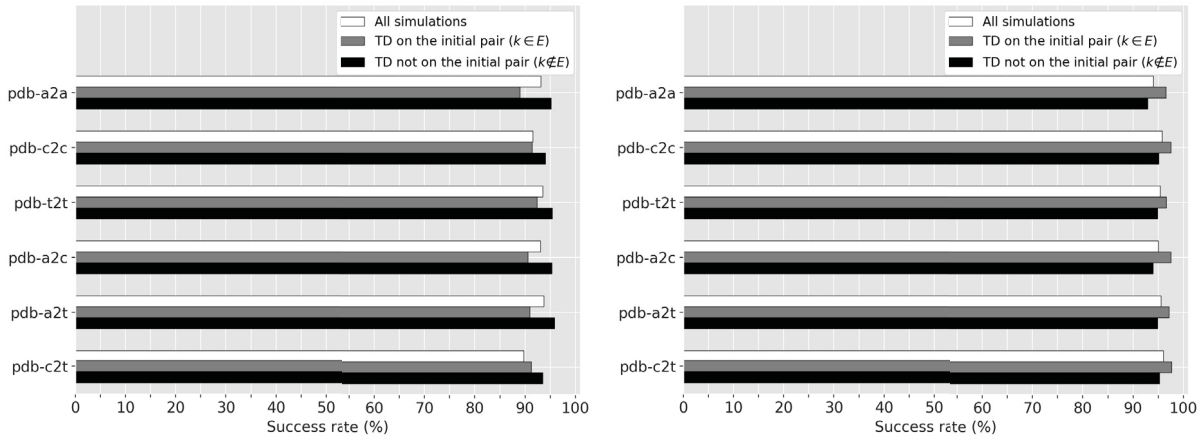
Results show that the *degree-le-2* and *vfbet-top-1000* sets achieved the best success rates, 94% and 93%, respectively. However, *degree-le-2* employed a significantly larger number of probes on average. This happens because the AS pairs selected for measurement from *degree-le-2* are usually farther from each other than the AS pairs selected from *vfbet-top-1000*. The average valley-free distance between selected AS pairs from *degree-le-2* was 2.01, while it was 1.48 for pairs from *vfbet-top-1000*. The average valley-free distances to the suspects were 1.8 and 0.79 for the same sets, respectively. Therefore, larger numbers of pairs were needed in the scenarios employing *degree-le-2*, since several of the selected pairs do not help filtering the suspects in the inference step of our strategy: the *discriminatory* AS k is present more often in the paths between the selected pairs, as described previously. 8269 different ASes (from a total of 41247) were selected for measurement from the *degree-le-2* set, and 615 (from a total of 1000) from *vfbet-top-1000*. These observations can be further confirmed with the results for the simulations in which no TD was present. The success rates for sets *degree-le-2* and *vfbet-top-1000* on those simulations were 94% and 91%, and the average numbers of probes were 5.27 and 5.15, respectively. Since there were no *discriminatory* ASes in these simulations, the selected pairs always filtered the corresponding suspects.

Furthermore, *pdb-transit* achieved a slightly smaller success rate than *vfbet-top-1000* (88%), with a similar amount of probes. However, more ASes were employed for the measurements considering all simulations (811 from a total of 2293). Sets *degree-eq-1* and *pdb-access* achieved the smallest success rates, 77% and 71%, respectively. However, *degree-eq-1* employed, on average, significantly more probes than *pdb-access*. Moreover, 6271 ASes (from a total of 21220) were selected for measurement set *degree-eq-1*, while 2177 were selected (from a total of 5263) for the *pdb-access* measurement set. The larger amount

of probes and selected measurement ASes for *degree-eq-1* happened due to the same reason described above for *degree-le-2*. Results also show that for all sets of measurement ASes, the average number of probes employed in unsuccessful simulations is significantly larger than those of successful simulations. This is due to the termination conditions we adopted: all the possible measurement AS pairs for all suspects are selected in every simulation that does not succeed.

6.5.4 Results: Comparing Initial Pairs

We now present results comparing different sets Z of initial pairs. We present results for the sets of measurement ASes *degree-le-2* and *vfbet-top-1000*, which presented the largest success rates and contain ASes at different parts of the Internet – edge (*degree-le-2*) and core (*vfbet-top-1000*). Figure 6.12 shows the success rates (for all simulations, $k \in E$, and $k \notin E$) for the sets of initial pairs *pdb-a2a, pdb-c2c, pdb-t2t, pdb-a2c, pdb-a2t*, and *pdb-c2t*. Figure 6.12(a) shows the results for $M =$ *vfbet-top-1000*, while Figure 6.12(b) for $M =$ *degree-le-2*. Furthermore, Figure 6.13 shows the average number of probes (for all simulations, then only those that were successful, and then to unsuccessful simulations) for each set of initial pairs. Figure 6.13(a) shows the results for $M =$ *vfbet-top-1000*, while Figure 6.13(b) for $M =$ *degree-le-2*.



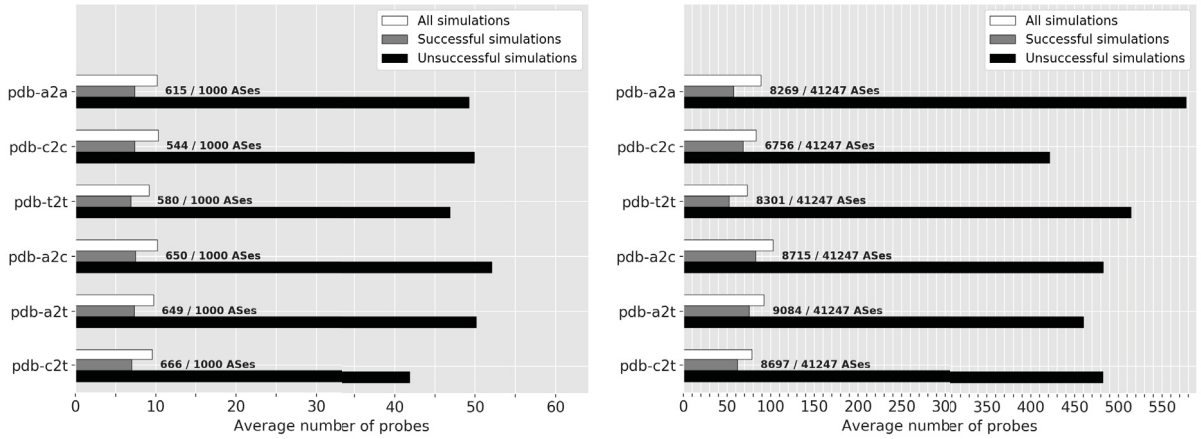
(a) $M =$ vfbet-top-1000 (b) $M =$ degree-le-2

Figure 6.12: Success rates for different initial pair sets Z, with $M =$ vfbet-top-1000 and $M =$ degree-le-2.

It is possible to conclude that both measurement sets had similar success rates for all sets of initial pairs. The success rates for *vfbet-top-1000* ranged from 89% to 93%, while the success rates for scenarios with *degree-le-2* ranged from 94% to 96%. The main difference between the two sets was that scenarios with *degree-le-2* employed significantly more probes on average, ranging from 73.12 to 102.48. The number of different ASes selected for measurement from *degree-le-2* ranged from 6756 to 9084 (from a total of 41247). For scenarios with *vfbet-top-1000*, the average number of probes ranged from 9.16 to 10.28, and the number of ASes selected for measurement ranged from 544 to 666 (from a total of 1000).

6.5.5 Results: $E \not\subset M$

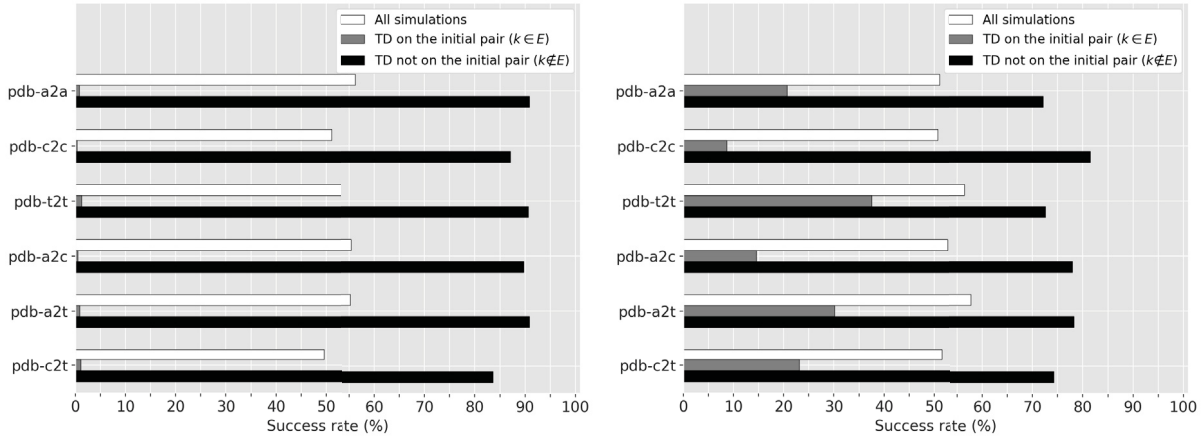
We also simulated scenarios not considering that the initial pair E is available for sending probes. In this way we are checking whether it is possible to detect TD between two AS



(a) M = vfbet-top-1000 (b) M = degree-le-2

Figure 6.13: Average number of probes for different initial pair sets Z, with M = vfbet-top-1000 and M = degree-le-2.

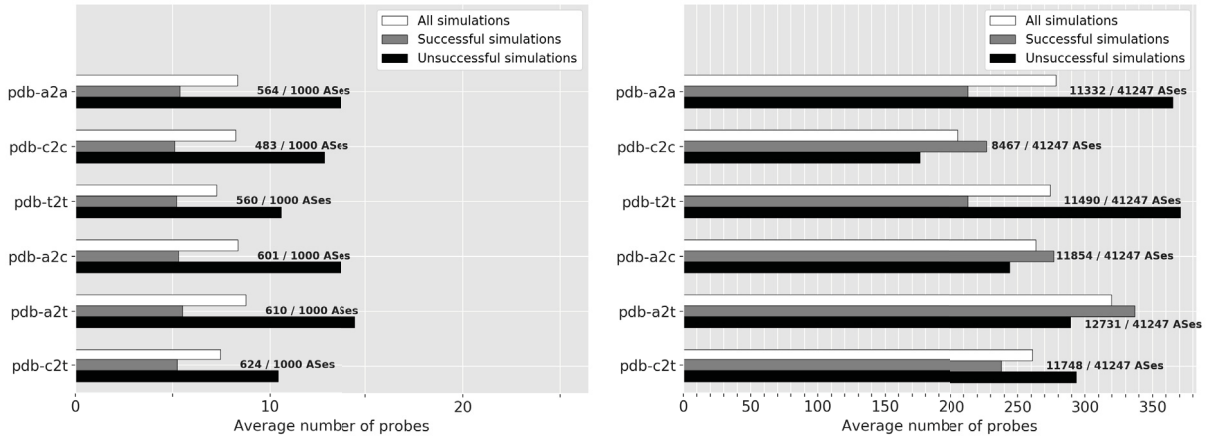
pairs so that no measurement host is available in any of those ASes. In these scenarios, only the ASes in M are available for measurement – and in case the ASes in the initial pair are present in M, we remove them from the set for that simulation scenario to ensure they are not available. Figure 6.14 shows the success rates for the sets of measurement ASes *vfbet-top-1000* (6.14(a)) and *degree-le-2* (6.14(b)), on scenarios with different sets Z. Similarly, Figure 6.15 shows the average number of probes for the sets of measurement ASes *vfbet-top-1000* (6.15(a)) and *degree-le-2* (6.15(b)).



(a) M = vfbet-top-1000 (b) M = degree-le-2

Figure 6.14: Success rates for different sets Z, with M = vfbet-top-1000 and M = degree-le-2, and $E \not\subset M$.

Results show that the success rates for both sets of measurement ASes were similar. The success rates for all simulations on scenarios with *vfbet-top-1000* ranged from 49% to 56%, while the success rate for *degree-le-2* ranged from 50% to 57%. As expected, the success rates for both sets were significantly smaller than those of the scenarios previously presented (for $E \subset M$). However, for both sets, the success rates for the simulations in which $k \notin E$ were significantly higher than for simulations with $k \in E$. For *vfbet-top-1000*, the success rates when $k \notin E$ ranged from 83% to 90%, and for *degree-le-2* ranged from 72% to 81%. When $k \in E$, the success rates ranged from 0% to 1% for *vfbet-top-1000*, and from 8% to 37% for *degree-le-2*. We explain these results below.



(a) M = vfbet-top-1000 (b) M = degree-le-2

Figure 6.15: Average number of probes for different sets Z, with M = vfbet-top-1000 and M = degree-le-2, and $E \not\subset M$.

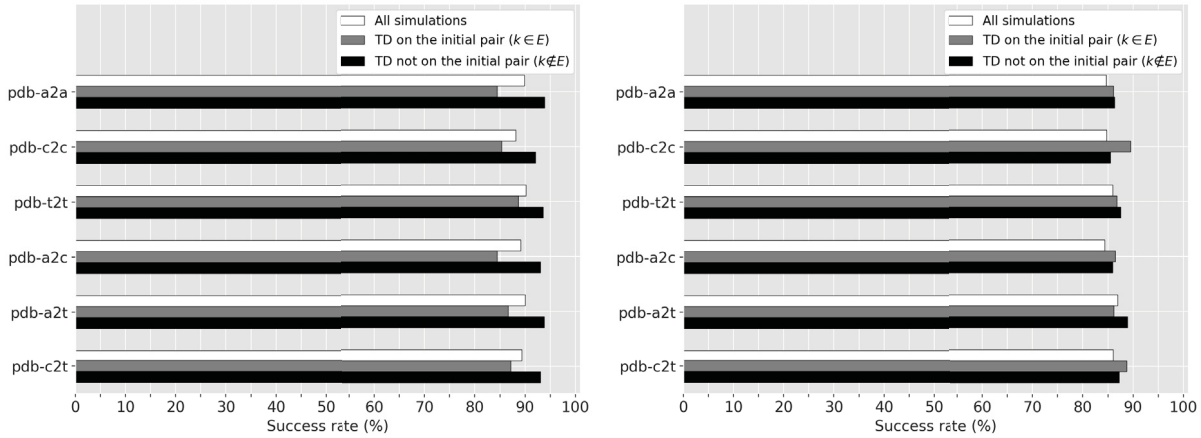
Due to the valley-free property, it is possible that there are no paths between ASes of the Internet core that traverse ASes on the edge of the Internet (or closer to the edge). ASes on the core, such as the ASes in *vfbet-top-1000*, are mostly connected to other ASes through *p2p* or *p2c* relationships – they are on the top of the Internet hierarchy (Tiers 1 and 2). For instance, only 1.8% of the relationships from ASes in *vfbet-top-1000* to other ASes are *c2p*. Therefore, the paths between these ASes usually consist of other ASes with the same characteristics. If a path between two such ASes traverses an edge AS it would violate the valley-free property, since at some point there would be a *p2c* link to the AS on the edge, followed by a *c2p* link going back to an AS on the core – i.e., a "valley". In this set of simulations, since the initial pair of ASes is not available for measurement, our strategy needs at least one measurement pair for which the paths traverse through the *discriminatory* AS k: new suspects are then found, potentially better positioned so it is possible to measure and find them. However, it is often not possible to find such measurement pair when $k \in E$. The ASes in *degree-le-2* are on the edge of the Internet, so it was possible to find paths traversing some of the ASes in the initial pairs, hence the higher success rates. The success rates for sets Z containing transit providers (*pdb-t2t, pdb-a2t* and *pdb-c2t*) were higher, which supports our claims.

Furthermore, the average number of probes in unsuccessful simulations was significantly smaller for $E \not\subset M$, when compared to the results presented previously in this section. However, the average number of probes in successful simulations is similar. For instance, let us take M = *vfbet-top-1000* and Z = *pdb-a2a*. The average number of probes in successful simulations for this configuration and $E \subset M$ was 7.39, as can be observed in Figure 6.13, while in unsuccessful simulations the average was 49.22. For $E \not\subset M$ (Figure 6.15), the average in successful simulations was 5.36, while the average in unsuccessful simulations was 13.72. The reason for this behavior is the same as described above: in the unsuccessful simulations, our strategy was able to find a much lower number of suitable AS pairs for issuing probes from when $E \not\subset M$. In the successful cases, a similar number of AS pairs was necessary.

6.5.6 Results: $\sigma = 1$

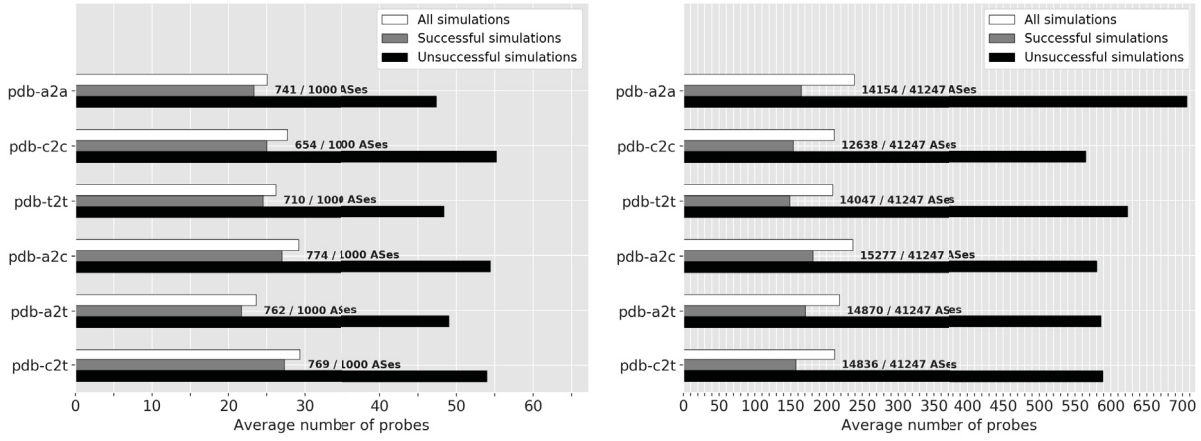
In the experiments previously described in Section 6.4, 55.78% of the measured valley-free paths had the same size of the corresponding shortest valley-free path in the graph, while 31.87% of the measured valley-free paths were one link larger than the shortest path. These represent 87.65% of all valley-free paths observed in the experiments. Therefore, we executed several simulations with $\sigma = 1$ to check if our proposal is capable of locating TD with a larger amount of possible paths between end-hosts.

Figure 6.16 shows the success rates for the sets of measurement ASes *vfbet-top-1000* (6.16(a)) and *degree-le-2* (6.16(b)), on scenarios with different sets Z. Similarly, Figure 6.17 shows the average number of probes for the sets of measurement ASes *vfbet-top-1000* (6.17(a)) and *degree-le-2* (6.17(b)).



(a) M = vfbet-top-1000 (b) M = degree-le-2

Figure 6.16: Success rates for different initial pair sets Z and $\sigma = 1$, with M = vfbet-top-1000 and M = degree-le-2.



(a) M = vfbet-top-1000 (b) M = degree-le-2

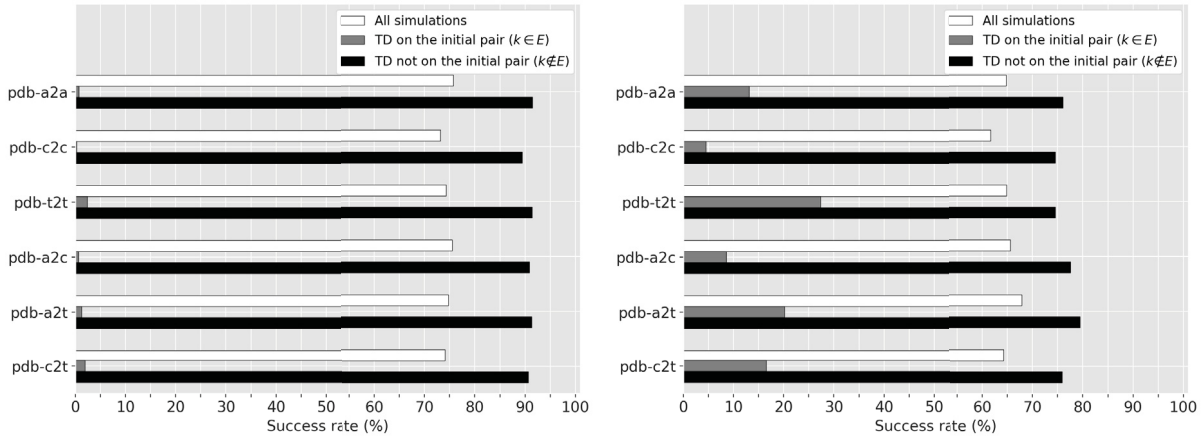
Figure 6.17: Average number of probes for different initial pair sets Z and $\sigma = 1$, with M = vfbet-top-1000 and M = degree-le-2.

For the *vfbet-top-1000* set, the success rates ranged from 88% to 90% for all simulations. These values were similar to the success rates for $\sigma = 0$ and the same set (Figure 6.12), which ranged from 89% to 93%. For the *degree-le-2* set, the success rates for

$\sigma = 1$ ranged from 84% to 87%. For $\sigma = 0$, the success rates ranged from 94% to 96% (Figure 6.12). It is also possible to observe that the average number of probes increased significantly for both sets in comparison with the results presented previously in Figure 6.13.

In all results presented in previous subsections, the success rates were always slightly larger for *degree-le-2*. However, for $\sigma = 1$, the success rates are slightly larger for the *vfbet-top-1000* set. Since ASes in *vfbet-top-1000* are generally closer to each other, the amount of possible paths between measurement ASes increases much more for *degree-le-2* then for the *vfbet-top-1000* set when $\sigma = 1$.

We also present results for $\sigma = 1$ and $E \not\subset M$, i.e., not considering that the initial pair is available for measurement. Figure 6.18 shows the success rates for the sets of measurement ASes *vfbet-top-1000* (6.18(a)) and *degree-le-2* (6.18(b)), on scenarios with different sets Z and $E \not\subset M$.



(a) M = vfbet-top-1000

(b) M = degree-le-2

Figure 6.18: Success rates for different initial pair sets Z and $\sigma = 1$, with M = vfbet-top-1000 and M = degree-le-2, and $E \not\subset M$.

Surprisingly, the success rates for both sets of measurement ASes were larger than those of the results presented previously in Figure 6.14 (for $\sigma = 0$). The success rates of all simulations for *vfbet-top-1000* ranged from 73% to 75% (49% to 56% in previous results). For the *degree-le-2* set, the success rates ranged from 62% to 68% (50% to 57% in previous results). The reason for this behavior is that since with $\sigma = 1$ there are more possible paths between measurement ASes, it is easier to find a measurement pair for which the paths contain the discriminatory AS k. When such pair is found, new suspects start to be investigated, which are better positioned (relative to the valley-free property) than the initial suspects, as we explained previously in Subsection 6.5.5: when $\sigma = 0$, it is less likely that k will be in the paths between the measurement ASes.

6.5.7 Discussion

We highlight that our simulations never produced false-negatives (a *discriminatory* AS k was never inferred as *neutral*) nor false positive results (*neutral* ASes were never inferred as *discriminatory*). However, if TD is mistakenly detected or if traffic traverses valley paths, then false-positives or false-negatives might happen. Furthermore, the oracle always assumed the worst case, i.e., traffic would always follow the path containing the

discriminatory AS k. However, if the actual path does not traverse k, fewer probes might be necessary to locate TD, since suspects might be filtered earlier.

Furthermore, results clearly show that the valley-free betweenness centrality is a good metric for selecting measurement ASes. Our results show that having few measurement ASes (1000 from the *vfbet-top-1000* set) on the core of the network led to similar success rates as having a larger number of measurement ASes on the edge (41247 from the *degree-le-2* set). Moreover, using ASes on the core resulted in much less probes. ASes on the core are generally closer to a larger portion of the network, while ASes on the edge are often farther away. Therefore, to achieve good success rates using ASes on the edge, there should be a much larger number of them available for measurement at several parts of the network, in order to "cover" several vantage points. Finally, results show that it is possible to locate TD between any two ASes, even if we do not have access to them for issuing probes. However, locating TD that is happening on the core of the Internet is easier then locating TD on the edge.

In comparison with the existing solutions for locating TD in the Internet, our proposal is able to locate TD without relying on *traceroute*-like techniques, while making more realistic assumptions. NetPolice, ChkDiff, and NeutMon all employ *traceroute*-like techniques to measure the exact host-level path from/to end-hosts, in order to locate where in the path TD took place. The experiments described in Section 6.4 show that these techniques may not produce reliable results. Instead of relying on such techniques, we consider all possible paths the traffic may take, and make inferences based only on ASes that were surely traversed.

Furthermore, the TD solution based on network tomography combines measurements taken from several different end-hosts to infer exactly in which host TD occurred. However, the authors assume complete knowledge of the network host-level topology, as well as knowledge of the exact path traffic traverses between end-hosts. These assumptions are not realistic in the Internet, since the Internet host-level topology is not only hard to obtain, it is also constantly and rapidly changing, and operators are not usually willing to provide details about their networks. We assume knowledge of the AS-level topology, which is feasible to be obtained in the Internet [54]. Furthermore, we do not assume which exact path traffic actually traverses – we consider all the possible paths given the routing restrictions. Moreover, we also provided a metric (the valley-free betweenness centrality) for choosing good measurement points, which is not addressed by other works.

7 CONCLUSION

As the number of Internet users reaches the 4 billion mark, and a myriad of Internet services become available, governments worldwide are deploying NN regulations. In several countries TD is illegal. However, regulations are not enough to ensure ISP compliance. Solutions to monitor and enforce NN compliance are necessary. We argue that even without regulations, maintaining the transparency on traffic management practices on the Internet is important by itself and can lead to a more competitive market, and foster innovation. Detecting TD on the Internet, however, is still a challenge. An even harder problem is to locate where within the network TD is occurring. Furthermore, studying NN in the context of emerging technologies, such as IoT, is still an under-explored topic.

In this thesis, we first described the current state of the art regarding the problem of detecting and locating TD on the Internet. We defined the problems, and surveyed existing solutions, highlighting the techniques employed by each one, how they implement these techniques, and what types of TD each solution is capable of detecting. We also presented open challenges, and proposed a taxonomy for the different aspects of TD and its detection. From the open challenges we identified in the current state of the art, in this thesis we focused on investigating TD on contexts not yet explored (such as IoT), as well as on proposing solutions for detecting and locating TD that can be employed in any of such contexts, taking advantage of both existing and emerging technologies.

For investigating TD on IoT, we first described common IoT traffic patterns and discussed how TD may impact those patterns. We then presented simulation results showing how different TD scenarios affected each traffic pattern. We concluded that even a small prioritization may introduce a significant difference between different traffic priorities. This difference might greatly influence the QoE perceived by end-users of IoT applications and devices.

Next, we proposed a solution for detecting TD between two end-hosts in the Internet, based on the techniques employed by the several existing solutions. A novel idea introduced by our proposal is combining several different metrics to enable the detection of more types of TD. In order to evaluate this proposal, we executed several simulations. Results show that the proposed strategy of combining several metrics was capable of correctly detecting TD under several conditions.

A general model for continuously monitoring NN in the Internet was then proposed. This model is a first step towards a more capable and future-proof solution that takes advantage of emerging technologies to monitor NN. The goal is to unify the solutions for detecting TD into a single framework, while enabling new devices and applications to participate in the system. With this model in mind, we then proposed a strategy to identify which AS is practicing TD in the Internet. The goal of this proposal is twofold: (i) to enable the implementation of the proposed model for monitoring NN, by combining TD detection results and/or measurements from several different vantage points; and (ii) to address the problem of locating TD in the Internet.

Our strategy for locating TD identifies which AS between two end-hosts is employing TD, without explicitly measuring the path the traffic takes – such as with *traceroute*-like techniques. The proposed strategy investigates several ASes suspected of being responsible for TD until only the AS that is actually discriminating traffic remains. In order to investigate the suspects, we take advantage of the valley-free property of AS-level paths in the

Internet to carefully select measurement points between which all possible valley-free paths traverse the suspects. End-to-end measurements and/or TD detection results between the selected measurement points then help infer the behavior of the suspects. Our proposal is an alternative to the *traceroute*-based solutions previously proposed for locating TD in the Internet, since we do not rely on path discovery measurements, only on end-to-end measurements.

To evaluate our proposals for locating TD, we first performed an experiment on PlanetLab, in which we executed *traceroute* a large number of times for determining the paths from a set of end-hosts to several Internet prefixes. Results show that *traceroute*-like techniques indeed may not be reliable, and that the vast majority of the paths that were successfully measured do follow the valley-free property. We then executed simulations for evaluating our strategy for locating TD. We defined several scenarios, varying the location of TD and the measurement points employed. We draw four main conclusions from the results obtained on these simulations: (i) few measurement ASes on the core of the network achieve similar results to those that employ larger amounts of measurement ASes on the edge; (ii) it is possible to locate TD between any two ASes in the Internet, even if they are not accessible for issuing probes from; (iii) due to the valley-free property, it is easier to locate TD on the core than on the edge; and (iv) the valley-free betweenness centrality is a good metric for selecting measurement ASes.

Future work includes effectively implementing and evaluating the proposed general model for monitoring NN in the Internet: e.g., a crowdsourcing system, in which participating users rely on the system to monitor whether they are being victims of TD, but also allow their devices to be used as measurement points for other users. Another direction is the development of a hybrid version of our proposal for locating TD using *traceroute*-like techniques: if the exact path between ASes can be obtained, our proposal does not need to consider all possible paths. Furthermore, our proposals consider only TD triggered by application. Detecting and locating TD triggered by path is still an under-explored topic. Finally, another research direction is to design a system that, after locating which AS is discriminating traffic, deviates traffic through a path known to be fully neutral, circumventing the discriminatory AS.

REFERENCES

[1] K. G. Coffman and A. M. Odlyzko. *Internet Growth: Is There a "Moore's Law" for Data Traffic?*, pages 47–93. Springer, 2002.

[2] Barbara van Schewick and David Farber. Point/Counterpoint: Network Neutrality Nuances. *Communications of the ACM*, 52(2):31–37, February 2009.

[3] R. Ravaioli, G. Urvoy-Keller, and C. Barakat. Towards a General Solution for Detecting Traffic Differentiation at the Internet Access. In *International Teletraffic Congress (ITC)*, pages 1–9, September 2015.

[4] Milton L. Mueller and Hadi Asghari. Deep packet inspection and bandwidth management: Battles over BitTorrent in Canada and the United States. *Telecommunications Policy*, 36(6):462–475, 2012.

[5] Fei-Yang Ling, Shou-Lian Tang, Miao Wu, Ya-Xian Li, and Hui-Ying Du. Research on the net neutrality: The case of Comcast blocking. In *International Conference on Advanced Computer Theory and Engineering (ICACTE)*, volume 5, pages V5–488–V5–491, August 2010.

[6] Hassan Habibi Gharakheili, Arun Vishwanath, and Vijay Sivaraman. Perspectives on Net Neutrality and Internet Fast-Lanes. *SIGCOMM Computer Communication Review*, 46(1):64–69, January 2016.

[7] James Kendrick. T-Mobile Germany Blocks iPhone Skype Over 3G and WiFi. `https://gigaom.com/2009/04/06/t-mobile-germany-blocks-iphone-skype-over-3g-too`, 2009. Accessed in August 28th, 2019.

[8] Natasha Lomas. Verizon Accused Of Net Neutrality Foul By Zero-Rating Its Go90 Mobile Video Service. `https://techcrunch.com/2016/02/07/verizon`, 2016. Accessed in August 28th, 2019.

[9] Alan Joch. Debating net neutrality. *Communications of the ACM*, 52(10):14–15, October 2009.

[10] Ministry of Internal Affairs and Communications. Report from Panel on Neutrality of Networks. `http://www.soumu.go.jp/main_sosiki/joho_tsusin/eng/Releases/NewsLetter/Vol18/Vol18_23/Vol18_23.html`, 2008. Accessed in August 28th, 2019.

[11] Norwegian Communications Authority. Net neutrality. `http://eng.nkom.no/technical/internet/net-neutrality/net-neutrality`. Accessed in August 28th, 2019.

[12] Canadian Radio-television and Telecommunications Commission. Review of the Internet traffic management practices of Internet service providers. `http://www.crtc.gc.ca/eng/archive/2009/2009-657.htm`, 2009. Accessed in August 28th, 2019.

[13] Subsecretaría de Telecomunicaciones. Consagra el Principio de Neutralidad en la Red para los Consumidores y Usuarios de Internet. http://www.leychile.cl/Navegar?idNorma=1016570, 2010. Accessed in August 28th, 2019.

[14] El Congreso de Colombia. Ley 1.450 de 2011: por la cual se expide el Plan Nacional de Desarrollo, 2010-2014. http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=43101, 2011. Accessed in August 28th, 2019.

[15] La Comisión de Regulación de Comunicaciones. Resolución 3.502 de 2011: por la cual se establecen las condiciones regulatorias relativas a la neutralidad en Internet, en cumplimiento de lo establecido en el artículo 56 de la Ley 1450 de 2011. http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=45061, 2011. Accessed in August 28th, 2019.

[16] Korea Communications Commission. Annual Report 2011. http://eng.kcc.go.kr/download.do?fileSeq=35215, 2012. Accessed in August 28th, 2019.

[17] Presidência da República. Lei 12965. http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2014/lei/l12965.htm, 2014. Accessed in August 28th, 2019.

[18] Presidência da República. Decreto 8771. http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2016/Decreto/D8771.htm, 2016. Accessed in August 28th, 2019.

[19] Secretaria de Comunicaciones y Transportes. Ley Federal de Telecomunicaciones y Radiodifusión. http://www.sct.gob.mx/fileadmin/Comunicaciones/LFTR.pdf, 2014. Accessed in August 28th, 2019.

[20] Federal Communications Commission. FCC Adopts Strong, Sustainable Rules to Protect the Open Internet. https://www.fcc.gov/document/fcc-adopts-strong-sustainable-rules-protect-open-internet, 2015. Accessed in August 28th, 2019.

[21] Telecom Regulatory Authority of India. Prohibition of Discriminatory Tariffs for Data Services. https://main.trai.gov.in/sites/default/files/Regulation_Data_Service.pdf, 2016. Accessed in August 28th, 2019.

[22] Body of European Regulators for Electronic Communications. BEREC Guidelines on the Implementation by National Regulators of European Net Neutrality Rules. http://berec.europa.eu/eng/document_register/subject_matter/berec/regulatory_best_practices/guidelines/6160, 2017. Accessed in August 28th, 2019.

[23] Jon Crowcroft. Net Neutrality: The Technical Side of the Debate: a White Paper. *SIGCOMM Computer Communication Review*, 37(1):49–56, 2007.

[24] S. Dustdar and E. P. Duarte. Network Neutrality and Its Impact on Innovation. *IEEE Internet Computing*, 22(6):5–7, Nov 2018.

[25] Tim Berners-Lee. Long Live the Web. *Scientific American*, 303(6):80–85, 2010.

[26] Hong Guo and Robert F. Easley. Network Neutrality Versus Paid Prioritization: Analyzing the Impact on Content Innovation. *Production and Operations Management*, 25(7):1261–1273, 2016.

[27] R. T. B. Ma. Pay or Perish: The Economics of Premium Peering. *IEEE J. Sel. Areas Commun.*, 35(2):353–366, Feb 2017.

[28] H. Schulzrinne. Network Neutrality Is About Money, Not Packets. *IEEE Internet Computing*, 22(6):8–17, Nov 2018.

[29] Johannes M. Bauer and Günter Knieps. Complementary innovation and network neutrality. *Telecommunications Policy*, 42(2):172–183, 2018.

[30] Yiannis Yiakoumis, Sachin Katti, and Nick McKeown. Neutral net neutrality. In *ACM SIGCOMM*, pages 483–496. ACM, 2016.

[31] Body of European Regulators for Electronic Communications. What are specialised services and how are they relevant to the Regulation? `http://berec.europa.eu/eng/netneutrality/specialised_services`. Accessed in August 28th, 2019.

[32] S. Jordan. Some Traffic Management Practices Are Unreasonable. In *International Conference on Computer Communications and Networks*, pages 1–6, August 2009.

[33] Scott Jordan and Arijit Ghosh. A Framework for Classification of Traffic Management Practices As Reasonable or Unreasonable. *ACM Transactions on Internet Technology*, 10(3):12:1–12:23, October 2010.

[34] Ryan Knutson and Shalini Ramachandran. Netflix Throttles Its Videos on AT&T, Verizon Networks. `http://www.wsj.com/articles/netflix-throttles-its-videos-on-at-t-verizon-phones-1458857424`, March 2016. Accessed in August 28th, 2019.

[35] P. Maille, G. Simon, and B. Tuffin. Toward a net neutrality debate that conforms to the 2010s. *IEEE Communications Magazine*, 54(3):94–99, March 2016.

[36] Body of European Regulators for Electronic Communications. Summary of BEREC positions on net neutrality. `http://berec.europa.eu/eng/document_register/subject_matter/berec/opinions/1128-summary-of-berec-positions-on-net-neutrality`, 2012. Accessed in August 28th, 2019.

[37] Mukarram Bin Tariq, Murtaza Motiwala, and Nick Feamster. NANO: Network Access Neutrality Observatory. In *7th ACM Workshop on Hot Topics in Networks (Hotnets-VII)*, 2008.

[38] P. Kanuparthy and C. Dovrolis. DiffProbe: Detecting ISP Service Discrimination. In *IEEE INFOCOM*, pages 1–9, March 2010.

[39] Partha Kanuparthy and Constantine Dovrolis. ShaperProbe: End-to-end Detection of ISP Traffic Shaping Using Active Methods. In *Internet Measurement Conference*, pages 473–482. ACM, 2011.

[40] Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, and Ramesh Govindan. An Internet-Wide Analysis of Traffic Policing. In *ACM SIGCOMM*, pages 468–482. ACM, 2016.

[41] G. Lu, Y. Chen, S. Birrer, F. E. Bustamante, and X. Li. POPI: A User-Level Tool for Inferring Router Packet Forwarding Priority. *IEEE/ACM Transactions on Networking*, 18(1):1–14, February 2010.

[42] Ying Zhang, Zhuoqing Morley Mao, and Ming Zhang. Detecting Traffic Differentiation in Backbone ISPs with NetPolice. In *Internet Measurement Conference*, pages 103–115. ACM, 2009.

[43] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna P. Gummadi, Ratul Mahajan, and Stefan Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *USENIX Conference on Networked Systems Design and Implementation*, pages 27–27. USENIX Association, 2010.

[44] 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 *Internet Measurement Conference*, pages 239–251. ACM, 2015.

[45] U. Weinsberg, A. Soule, and L. Massoulie. Inferring traffic shaping and policy parameters using end host measurements. In *IEEE INFOCOM*, pages 151–155, April 2011.

[46] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network Neutrality Inference. *SIGCOMM Computer Communication Review*, 44(4):63–74, October 2014.

[47] Robert Beverly, Steven Bauer, and Arthur Berger. *The Internet Is Not a Big Truck: Toward Quantifying Network Neutrality*, pages 135–144. Springer, 2007.

[48] E. Gregori, V. Luconi, and A. Vecchio. NeutMon: Studying neutrality in European mobile networks. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 523–528, April 2018.

[49] T. Garrett, L. E. Setenareski, L. M. Peres, L. C. E. Bona, and E. P. Duarte. Monitoring Network Neutrality: A Survey on Traffic Differentiation Detection. *IEEE Communications Surveys Tutorials*, 20(3):2486–2517, thirdquarter 2018.

[50] T. Garrett, S. Dustdar, L. C. E. Bona, and E. P. Duarte Jr. Traffic differentiation on internet of things. In *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, March 2018.

[51] T. Garrett, S. Dustdar, L. C. E. Bona, and E. P. Duarte Jr. Ensuring Network Neutrality for Future Distributed Systems. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 1780–1786, June 2017.

[52] E. Gregori, V. Luconi, and A. Vecchio. Studying Forwarding Differences in European Mobile Broadband with a Net Neutrality Perspective. In *European Wireless Conference*, pages 1–7, May 2018.

[53] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding Traceroute Anomalies with Paris Traceroute. In *Internet Measurement Conference*, pages 153–158. ACM, 2006.

[54] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and K. C. Claffy. AS Relationships, Customer Cones, and Validation. In *Internet Measurement Conference*, IMC '13, pages 243–256. ACM, 2013.

[55] Phillipa Gill, Michael Schapira, and Sharon Goldberg. A Survey of Interdomain Routing Policies. *SIGCOMM Computer Communication Review*, 44(1):28–34, December 2013.

[56] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *SIGCOMM Computer Communication Review*, 33(3), 2003.

[57] M. E. Tozal. The Internet: A system of interconnected autonomous systems. In *Annual IEEE Systems Conference (SysCon)*, pages 1–8, April 2016.

[58] Jan Krämer, Lukas Wiewiorra, and Christof Weinhardt. Net neutrality: A progress report. *Telecommunications Policy*, 37(9):794–813, 2013.

[59] R. Adams. Active Queue Management: A Survey. *IEEE Communications Surveys Tutorials*, 15(3):1425–1476, October 2013.

[60] Tim Wu. Network Neutrality, Broadband Discrimination. *Journal of Telecommunications and High Technology Law*, 2:141–176, 2003.

[61] Robert W. Hahn and Scott Wallsten. The Economics of Net Neutrality. *The Economists' Voice*, 3(6):1–7, 2006.

[62] Paul Ganley and Ben Allgrove. Net neutrality: A user's guide. *Computer Law & Security Review*, 22(6):454–463, 2006.

[63] A. Dainotti, A. Pescape, and K. C. Claffy. Issues and future directions in traffic classification. *IEEE Network*, 26(1):35–40, January 2012.

[64] Charles V Wright, Fabian Monrose, and Gerald M Masson. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research*, 7(Dec):2745–2769, 2006.

[65] Gregory Detal, Benjamin Hesmans, Olivier Bonaventure, Yves Vanaubel, and Benoit Donnet. Revealing Middlebox Interference with Tracebox. In *Internet Measurement Conference*, pages 1–8. ACM, 2013.

[66] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero. A survey on the contributions of Software-Defined Networking to Traffic Engineering. *IEEE Communications Surveys Tutorials*, 19(2):918–953, 2017.

[67] Ying Zhang, Z. Morley, and Mao Ming Zhang. Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs. In *ACM Workshop on Hot Topics in Networks*, pages 121–126, 2008.

[68] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. Detecting Network Neutrality Violations with Causal Inference. In *International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 289–300. ACM, 2009.

[69] Judea Pearl Sander Greenland, James M. Robins. Confounding and Collapsibility in Causal Inference. *Statistical Science*, 14(1):29–46, 1999.

[70] Nicholas P Jewell. *Statistics for epidemiology*. Chapman & Hall/CRC, 2004.

[71] Maria Silvia Abba Legnazzi, Cristina Rottondi, and Giacomo Verticale. Secure and Differentially Private Detection of Net Neutrality Violations by Means of Crowdsourced Measurements. *Wireless Personal Communications*, Sep 2018.

[72] Cynthia Dwork. *Differential Privacy*, pages 338–340. Springer US, 2011.

[73] Gottfried E Noether. *Introduction to statistics: the nonparametric way*. Springer, 2012.

[74] A. Coates, A. O. Hero III, R. Nowak, and Bin Yu. Internet tomography. *IEEE Signal Processing Magazine*, 19(3):47–65, 2002.

[75] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. Detecting Bittorrent Blocking. In *Internet Measurement Conference*, pages 3–8. ACM, 2008.

[76] V. Bashko, N. Melnikov, A. Sehgal, and J. Schönwälder. BonaFide: A traffic shaping detection tool for mobile networks. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 328–335, May 2013.

[77] D. R. Whitney H. B. Mann. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[78] Riccardo Ravaioli, Chadi Barakat, and Guillaume Urvoy-Keller. Chkdiff: Checking Traffic Differentiation at Internet Access. In *ACM Conference on CoNEXT Student Workshop*, pages 57–58. ACM, 2012.

[79] Marta Carbone and Luigi Rizzo. Dummynet Revisited. *SIGCOMM Computer Communication Review*, 40(2):12–20, 2010.

[80] David Choffnes. Wehe: Check Your ISP for Net Neutrality Violations. `https://dd.meddle.mobi/`. Accessed in August 28th, 2019.

[81] Fangfan Li, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. A Large-scale Analysis of Deployed Traffic Differentiation Practices. In *ACM Special Interest Group on Data Communication*, SIGCOMM '19, pages 130–144. ACM, 2019.

[82] Özgü Alay, Andra Lutu, Miguel Peón-Quirós, Vincenzo Mancuso, Thomas Hirsch, Kristian Evensen, Audun Hansen, Stefan Alfredsson, Jonas Karlsson, Anna Brunstrom, Ali Safari Khatouni, Marco Mellia, and Marco Ajmone Marsan. Experience: An Open Platform for Experimentation with Commercial Mobile Broadband Networks. In *International Conference on Mobile Computing and Networking (MobiCom)*, pages 70–78. ACM, 2017.

[83] Ionut Trestian, Rahul Potharaju, and Aleksandar Kuzmanovic. WindRider - A Mobile Network Neutrality Monitoring System. http://networks.cs.northwestern.edu/mobile-neutrality. Accessed in August 28th, 2019.

[84] Constantine Dovrolis, Krishna Gummadi, Aleksandar Kuzmanovic, and Sascha D. Meinrath. Measurement Lab: Overview and an Invitation to the Research Community. *SIGCOMM Computer Communication Review*, 40(3):53–56, June 2010.

[85] D. Miorandi, I. Carreras, E. Gregori, I. Graham, and J. Stewart. Measuring net neutrality in mobile Internet: Towards a crowdsensing-based citizen observatory. In *IEEE International Conference on Communications Workshops (ICC)*, pages 199–203, June 2013.

[86] M. Weber, V. Svedek, Z. Jukic, I. Golub, and T. Zuljevic. Can HAKOMetar be used to increase transparency in the context of network neutrality? In *International Conference on Telecommunications (ConTEL)*, pages 309–316, June 2013.

[87] M. Yuksel, K. K. Ramakrishnan, S. Kalyanaraman, J. D. Houle, and R. Sadhvani. Quantifying Overprovisioning vs. Class-of-Service: Informing the Net Neutrality Debate. In *International Conference on Computer Communications and Networks*, pages 1–8, August 2010.

[88] G. Hourton, G. Del Canto, J. Bustos, and F. Lalanne. Crowd-measuring: Assessing the quality of mobile Internet from end-terminals. In *International Conference on Network Games, Control and Optimization (NetGCooP)*, pages 145–148, November 2012.

[89] Zachary S. Bischof, John S. Otto, and Fabián E. Bustamante. Up, Down and Around the Stack: ISP Characterization from Network Intensive Applications. *SIGCOMM Computer Communication Review*, 42(4):515–520, September 2012.

[90] Mario A. Sánchez, John S. Otto, Zachary S. Bischof, and Fabián E. Bustamante. Dasu - ISP Characterization from the Edge: A BitTorrent Implementation. *SIGCOMM Computer Communication Review*, 41(4):454–455, August 2011.

[91] M. Aida, N. Miyoshi, and K. Ishibashi. A scalable and lightweight QoS monitoring technique combining passive and active approaches. In *IEEE INFOCOM*, volume 1, pages 125–133, March 2003.

[92] Ookla. The World Standard in Internet Metrics. https://www.ookla.com. Accessed in August 28th, 2019.

[93] Broadband Speed Checker. The UK's No.1 Broadband Speed Test. http://www.broadbandspeedchecker.co.uk. Accessed in August 28th, 2019.

[94] NIC.br. Sistema de Medição de Tráfego Internet (SIMET). http://simet.nic.br. Accessed in August 28th, 2019.

[95] J. Bustos-Jiménez, V. Ramiro, F. Lalanne, and T. Barros. Adkintun: SLA Monitoring of ISP Broadband Offerings. In *International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 1445–1449, March 2013.

[96] Javier Bustos-Jiménez and Camila Fuenzalida. All Packets Are Equal, but Some Are More Equal Than Others. In *Latin America Networking Conference (LANC)*, pages 5:1–5:8. ACM, 2014.

[97] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Accurate and Efficient SLA Compliance Monitoring. *SIGCOMM Computer Communication Review*, 37(4):109–120, August 2007.

[98] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Multiobjective Monitoring for SLA Compliance. *IEEE/ACM Transactions on Networking (TON)*, 18(2):652–665, April 2010.

[99] X. Ta and G. Mao. Online End-to-End Quality of Service Monitoring for Service Level Agreement Verification. In *IEEE International Conference on Networks*, volume 2, pages 1–6, September 2006.

[100] R. Serral-Gracia, Y. Labit, J. Domingo-Pascual, and P. Owezarski. Towards an Efficient Service Level Agreement Assessment. In *IEEE INFOCOM*, pages 2581–2585, April 2009.

[101] Tongqing Qiu, Jian Ni, Hao Wang, Nan Hua, Y. Richard Yang, and Jun Jim Xu. Packet Doppler: Network Monitoring Using Packet Shift Detection. In *ACM CoNEXT Conference*, pages 3:1–3:12. ACM, 2008.

[102] René Serral-Gracià, Marcelo Yannuzzi, Yann Labit, Philippe Owezarski, and Xavi Masip-Bruin. An efficient and lightweight method for Service Level Agreement assessment. *Computer Networks*, 54(17):3144–3158, 2010.

[103] Z. Jukic, M. Weber, V. Svedek, M. Vukovic, D. Katusic, and G. Jezic. Technical aspects of network neutrality. In *International Conference on Telecommunications (ConTEL)*, pages 405–410, June 2011.

[104] Javier Bustos-Jiménez, Gabriel Del Canto, Sebastián Pereira, Felipe Lalanne, José Piquer, Gabriel Hourton, Alfredo Cádiz, and Victor Ramiro. How AdkintunMobile Measured the World. In *ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp)*, pages 1457–1462. ACM, 2013.

[105] F. Lalanne, N. Aguilera, A. Graves, and J. Bustos. Adkintun Mobile: Towards using personal and device context in assessing mobile QoS. In *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 49–54, August 2015.

[106] Andreas Sfakianakis, Elias Athanasopoulos, and Sotiris Ioannidis. CensMon: A Web censorship monitor. In *USENIX Workshop on Free and Open Communications on the Internet*, 2011.

[107] Arturo Filasto and Jacob Appelbaum. OONI: Open Observatory of Network Interference. In *USENIX Workshop on Free and Open Communications on the Internet*, 2012.

[108] Tim Hwang. Threat Modeling: Herdict: A Distributed Model for Threats Online. *Network Security*, 2007(8):15–18, August 2007.

[109] Shinyoung Cho, Rishab Nithyanand, Abbas Razaghpanah, and Phillipa Gill. A Churn for the Better: Localizing Censorship Using Network-level Path Churn and Network Tomography. In *International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, pages 81–87. ACM, 2017.

[110] Giuseppe Aceto and Antonio Pescapé. Internet Censorship detection: A survey. *Computer Networks*, 83:381–421, 2015.

[111] Electronic Frontier Foundation. Switzerland Network Testing Tool. `https://www.eff.org/pages/switzerland-network-testing-tool`. Accessed in August 28th, 2019.

[112] Charles Reis, Steven D. Gribble, Tadayoshi Kohno, and Nicholas C. Weaver. Detecting In-flight Page Changes with Web Tripwires. In *USENIX Symposium on Networked Systems Design and Implementation (NDSI)*, 2008.

[113] Network Neutrality Squad. NNSquad Network Measurement Agent (NNMA). `https://www.nnsquad.org/agent.html`. Accessed in August 28th, 2019.

[114] Nicholas Weaver, Robin Sommer, and Vern Paxson. Detecting Forged TCP Reset Packets. In *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2009.

[115] Mohan Dhawan, Justin Samuel, Renata Teixeira, Christian Kreibich, Mark Allman, Nicholas Weaver, and Vern Paxson. Fathom: A Browser-based Network Measurement Platform. In *ACM Conference on Internet Measurement Conference (IMC)*, pages 73–86. ACM, 2012.

[116] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu. Characterizing Residential Broadband Networks. In *SIGCOMM Conference on Internet Measurement (IMC)*, pages 43–56. ACM, 2007.

[117] Ratul Mahajan, Ming Zhang, Lindsey Poole, and Vivek Pai. Uncovering Performance Differences Among Backbone ISPs with Netdiff. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[118] Zachary S. Bischof, John S. Otto, Mario A. Sánchez, John P. Rula, David R. Choffnes, and Fabián E. Bustamante. Crowdsourcing ISP Characterization to the Network Edge. In *SIGCOMM Workshop on Measurements Up the Stack (W-MUST)*, pages 61–66. ACM, 2011.

[119] Mario A. Sánchez, John S. Otto, Zachary S. Bischof, David R. Choffnes, Fabián E. Bustamante, Balachander Krishnamurthy, and Walter Willinger. Dasu: Pushing Experiments to the Internet's Edge. In *USENIX Conference on Networked Systems Design and Implementation*, pages 487–500. USENIX Association, 2013.

[120] SamKnows. The global platform for internet measurement. `https://www.samknows.com`. Accessed in August 28th, 2019.

[121] Demetris Antoniades, Evangelos P. Markatos, and Constantine Dovrolis. *MOR: Monitoring and Measurements through the Onion Router*, pages 131–140. Springer, 2010.

[122] V. Bajpai and J. Schönwälder. A Survey on Internet Performance Measurement Platforms and Related Standardization Efforts. *IEEE Communications Surveys Tutorials*, 17(3):1313–1341, 2015.

[123] K. V. Vishwanath and A. Vahdat. Swing: Realistic and Responsive Network Traffic Generation. *IEEE/ACM Transactions on Networking*, 17(3):712–725, June 2009.

[124] F. Michaut and F. Lepage. Application-oriented network metrology: metrics and active measurement tools. *IEEE Communications Surveys Tutorials*, 7(2):2–24, 2005.

[125] Simone Basso, Michela Meo, and Juan Carlos De Martin. Strengthening Measurements from the Edges: Application-level Packet Loss Rate Estimation. *SIGCOMM Computer Communication Review*, 43(3):45–51, July 2013.

[126] V. Raida, P. Svoboda, and M. Rupp. Lightweight Detection of Tariff Limits in Cellular Mobile Networks. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7, Sep. 2018.

[127] H. Lan, W. Ding, and J. Gong. Useful Traffic Loss Rate Estimation Based on Network Layer Measurement. *IEEE Access*, 7:33289–33303, 2019.

[128] J. C. De Martin and A. Glorioso. The Neubot project: A collaborative approach to measuring internet neutrality. In *IEEE International Symposium on Technology and Society*, pages 1–4, June 2008.

[129] S. Basso, A. Servetti, and J. C. De Martin. The network neutrality bot architecture: A preliminary approach for self-monitoring of Internet access QoS. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 1131–1136, June 2011.

[130] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: Illuminating the Edge Network. In *SIGCOMM Conference on Internet Measurement (IMC)*, pages 246–259. ACM, 2010.

[131] V. Garcez Schaurich, M. Barbosa de Carvalho, and L. Zambenedetti Granville. ISPANN: A Policy-Based ISP Auditor for Network Neutrality Violation Detection. In *IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 1081–1088, May 2018.

[132] Wilfried Mayer, Thomas Schreiber, and Edgar Weippl. A Framework for Monitoring Net Neutrality. In *International Conference on Availability, Reliability and Security (ARES)*, pages 17:1–17:10. ACM, 2018.

[133] Don Batory. Feature Models, Grammars, and Propositional Formulas. In *International Conference on Software Product Lines*, pages 7–20. Springer-Verlag, 2005.

[134] Yu-Chung Cheng, Urs Hölzle, Neal Cardwell, Stefan Savage, and Geoffrey M. Voelker. Monkey See, Monkey Do: A Tool for TCP Tracing and Replaying. In *USENIX Annual Technical Conference (ATEC)*, 2004.

[135] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios. *Computer Networks*, 56(15):3531–3547, 2012.

[136] Zoraida Frias and Jorge Pérez Martínez. 5G networks: Will technology and policy collide? *Telecommunications Policy*, 42(8):612–621, 2018.

[137] L. G. M. Ballesteros, Ó. Álvarez, and J. Markendahl. Quality of Experience (QoE) in the smart cities context: An initial analysis. In *International Smart Cities Conference (ISC2)*, pages 1–7, Oct 2015.

[138] I. Lovrek, A. Caric, and D. Lucic. Future network and future internet: A survey of regulatory perspective. In *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 186–191, Sep. 2014.

[139] Alex Maltinsky, Ran Giladi, and Yuval Shavitt. On Network Neutrality Measurements. *ACM Transactions on Intelligent Systems and Technology*, 8(4):56:1–56:22, May 2017.

[140] Vijay Sivaraman, Sharat Chandra Madanapalli, Himal Kumar, and Hassan Habibi Gharakheili. OpenTD: Open Traffic Differentiation in a Post-Neutral World. In *ACM Symposium on SDN Research (SOSR)*, pages 119–126. ACM, 2019.

[141] D. Andreoletti, S. Giordano, C. Rottondi, M. Tornatore, and G. Verticale. To be Neutral or Not Neutral? The In-Network Caching Dilemma. *IEEE Internet Computing*, 22(6):18–26, Nov 2018.

[142] Pio Baake and Slobodan Sudaric. Net neutrality and CDN intermediation. *Information Economics and Policy*, 46:55–67, 2019.

[143] Jan Krämer and Martin Peitz. A fresh look at zero-rating. *Telecommunications Policy*, 42(7):501–513, 2018.

[144] Ioannis Avramopoulos, J Rexford, D Syrivelis, and S Lalis. Counteracting discrimination against network traffic. Technical report, Tech. Rep. TR-794-07, Princeton University Computer Science, 2007.

[145] M. B. de Carvalho, V. G. Schaurich, and L. Z. Granville. Considering Jurisdiction When Assessing End-to-End Network Neutrality. *IEEE Internet Computing*, 22(6):27–34, Nov 2018.

[146] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surveys Tuts.*, 17(4):2347–2376, Fourthquarter 2015.

[147] K. U. Rehman Laghari and K. Connelly. Toward total quality of experience: A QoE model in a communication ecosystem. *IEEE Commun. Mag.*, 50(4):58–65, April 2012.

[148] Dong-Hee Shin. Conceptualizing and measuring quality of experience of the Internet of Things: Exploring how quality is perceived by users. *Information & Management*, February 2017.

[149] Muhammad Zubair Shafiq, Lusheng Ji, Alex X. Liu, Jeffrey Pang, and Jia Wang. A First Look at Cellular Machine-to-machine Traffic: Large Scale Measurement and Characterization. *SIGMETRICS Perform. Eval. Rev.*, 40(1):65–76, June 2012.

[150] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke. Middleware for Internet of Things: A Survey. *IEEE Internet Things J.*, 3(1):70–95, Feb 2016.

[151] N. Nikaein, M. Laner, K. Zhou, P. Svoboda, D. Drajic, M. Popovic, and S. Krco. Simple Traffic Modeling Framework for Machine Type Communication. In *Int. Symp. Wireless Communication Systems*, pages 1–5, Aug 2013.

[152] András Varga and Rudolf Hornig. An Overview of the OMNeT++ Simulation Environment. In *Int. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 60:1–60:10. ICST, 2008.

[153] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Characterizing and Classifying IoT Traffic in Smart Cities and Campuses. In *IEEE Conf.Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017.

[154] CAIDA AS Rank. `http://as-rank.caida.org/`, 2018. Accessed in August 28th, 2019.

[155] M. E. Tozal. Enumerating single destination, policy-preferred paths in AS-level Internet topology maps. In *IEEE 37th Sarnoff Symposium*, pages 25–30, September 2016.

[156] V. Giotsas and S. Zhou. Valley-free violation in Internet routing – Analysis based on BGP Community data. In *IEEE International Conference on Communications (ICC)*, pages 1193–1197, June 2012.

[157] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[158] CAIDA. Routeviews Prefix to AS mappings Dataset for IPv4 and IPv6. `http://www.caida.org/data/routing/routeviews-prefix2as.xml`, 2018. Accessed in August 28th, 2019.

[159] PeeringDB. `http://www.peeringdb.com`. Accessed in August 28th, 2019.

[160] Aemen Lodhi, Natalie Larson, Amogh Dhamdhere, Constantine Dovrolis, and K. C. Claffy. Using peeringDB to Understand the Peering Ecosystem. *SIGCOMM Computer Communication Review*, 44(2):20–27, April 2014.

APPENDIX A – PUBLICATIONS

We list below the publications obtained during the development of this thesis.

1. **Thiago Garrett**, Luis C. E. Bona, Elias P. Duarte Jr., "Exploiting AS-level Routing Properties to Infer the Source of Traffic Differentiation in the Internet," *IEEE International Conference on Computer Communications (INFOCOM)*, Beijing, China, 2020. *(Submitted)*

2. **Thiago Garrett**, Ligia E. Setenareski, Leticia M. Peres, Luis C. E. Bona, Elias P. Duarte Jr., "Monitoring Network Neutrality: A Survey on Traffic Differentiation Detection," *IEEE Communications Surveys & Tutorials*, IEEE, 2018.

3. **Thiago Garrett**, Schahram Dustdar, Luis C. E. Bona, Elias P. Duarte Jr., "Traffic Differentiation on Internet of Things," *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, Bamberg, Germany, 2018.

4. **Thiago Garrett**, Schahram Dustdar, Luis C. E. Bona, Elias P. Duarte Jr., "Ensuring Network Neutrality for Future Distributed Systems," *IEEE International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, USA, 2017.

5. **Thiago Garrett**, Luis C. E. Bona, Elias P. Duarte Jr., "Improving the Performance and Reproducibility of Experiments on Large-scale Testbeds with k-cores," *Computer Communications*, Elsevier, Vol. 110, pp. 35-47, 2017. *(Related to the Master Dissertation)*

6. **Thiago Garrett**, Luis C. E. Bona, Elias P. Duarte Jr., "Explorando Propriedades do Roteamento na Internet para a Localização de Diferenciação de Tráfego," *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, Gramado, Brazil, 2019. *(In portuguese)*

7. **Thiago Garrett**, Schahram Dustdar, Luis C. E. Bona, Elias P. Duarte Jr., "Uma Avaliação do Impacto da Diferenciação de Tráfego na Internet das Coisas," *Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF-SBRC)*, Campos do Jordão, Brazil, 2018. *(In portuguese)*

8. Ligia E. Setenareski, **Thiago Garrett**, Leticia Peres, Luis C. E. Bona, Elias P. Duarte Jr., "Fiscalização da Neutralidade da Rede: Conceitos e Técnicas," *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 1-50, Belém, Brazil, 2017. *(In portuguese)*