Efficient Network traffic classifier: Composition approach.

by

Hossein Doroud

A dissertation submitted by in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Telematic Engineering

Universidad Carlos III de Madrid

Advisor:

Prof. Andrés Marín López

Tutor:

Prof. Andrés Marín López

October 2019

Acknowledgements

This thesis is the culmination of my journey of Ph.D which was just like climbing a high peak step by step accompanied with encouragement, hardship, trust, and frustration. When I found myself at top experiencing the feeling of fulfilment, I realised though only my name appears on the cover of this dissertation, a great many people including my family members, my friends and colleagues have contributed to accomplish this huge task.

Firstly, I would like to express my sincere gratitude to my advisor Prof. Andrès Marìn Lòpez for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

My sincere thanks also goes to Prof. Antonio Pescapè, Dr. Narseo Vallina-Rudriguez, Prof. Adam Wolisz and Prof. Anatolij Zubow, who provided me an opportunity to join their team as intern, and who gave access to the laboratory and research facilities. Without they precious support it would not be possible to conduct this research.

Finally, I acknowledge the people who mean a lot to me, my parents, Nazir and Iran, for showing faith in me and giving me liberty to choose what I desired. I salute you all for the selfless love, care, pain and sacrifice you did to shape my life. Although you hardly understood what I researched on, you were willing to support any decision I made. I would never be able to pay back the love and affection showered upon by my parents. Also I express my thanks to my brother Reza, and sisters Lina, Bahareh for their support and valuable prayers.

I owe thanks to a very special person, my Wife, Elnaz for her continued and unfailing love, support and understanding during my pursuit of Ph.D degree that made the completion of thesis possible. You were always around at times I thought that it is impossible to continue, you helped me to keep things in perspective. I greatly value her contribution and deeply appreciate her belief in me.

PUBLISHED AND SUBMITTED CONTENT

- H. Doroud, G. Aceto, W. D. Donato, E. A. Jarchlo, A. M. Lopez, C. D. Guerrero, and A. Pescape, "Speeding-Up DPI Traffic Classification with Chaining," 2018 IEEE Global Communications Conference (GLOBECOM), 2018.
    - DOI: 10.1109/GLOCOM.2018.8648137
    - As the main author, the idea, implementation and evolution were my contribution and I wrote the paper in collaboration and under supervision of the participated authors.
    - All content of the paper is used in the thesis and distributed in:
        - Introduction
        - Chapter 2
        - Chapter 6
    - The material from this source included in this thesis is not singled out with typographic means and references.

- F. Michclinakis, H. Doroud, A. Razaghpanah, A. Lutu, N. Vallina-Rodriguez, P. Gill, and J. Widmer, "The Cloud that Runs the Mobile Internet: A Measurement Study of Mobile Cloud Services," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018.
    - DOI: 10.1109/INFOCOM.2018.8485872
    - As the second author, my responsibility was to carrying out passive measurement and write the corresponded section of the paper under supervision and in collaboration with the rest of the authors.
    - The content of the paper is used partially in this thesis. Chapter 4 represent my research work which I contributed with this paper.
    - The material from this source included in this thesis is not singled out with typographic means and references.

- F. Michelinakis, H. Doroud, A. Razaghpanah, A. Lutu, N. Vallina-Rodriguez P. Gill & J. Widmer, (2017). Content Distribution Networks in the mobile age.
    - URL: http://eprints.networks.imdea.org/id/eprint/1745
    - As the second author, my responsibility was to carry out passive measurement corresponded to the relevant section of the poster under supervision and in collaboration with the rest of the authors.
    - The content of the poster is used partially in this thesis. Chapter 4 represent my research work which I contributed with this poster.
    - The material from this source included in this thesis is not singled out with typographic means and references.

OTHER RESEARCH MERITS

- E. A. Jarchlo, X. Tang, H. Doroud, V. P. G. Jimenez, B. Lin, P. Casari, and Z. Ghassemlooy, "Li-Tect: 3-D Monitoring and Shape Detection Using Visible Light Sensors," IEEE Sensors Journal, vol. 19, no. 3, pp. 940–949, 2019.
- E. A. Jarchlo, S. M. Kouhini, H. Doroud, G. Maierbacher, M. Jung, B. Siessegger, Z. Ghassemlooy, A. Zubow, G. Caire, "Flight: A Flexible Light Communications network architecture for indoor environments", Conference Proceeding Contel2019

UNIVERSIDAD CARLOS III DE MADRID

Abstract

Telematic Engineering

Doctor of Philosophy

Efficient Network traffic classifier: Composition approach.

by Hossein Doroud

Internet Service Providers (ISP) are eagerly looking for obtaining metadata from the traffic that they carry. The obtained metadata is a valuable asset for ISPs to enhance their functionality and reduce their operational cost. Classifying a network traffic based on the application (app) that generates the traffic is vital for today's ISPs and network providers. They use Network Traffic Classification (NTC) to improve many aspects of their network like security and resource allocation. In addition, NTC enables the ISPs to offer new services to their customers and end users.

However, NTC faces a big challenge due to the high dynamic Internet ecosystem. Thousands apps are published daily[1] and NTC needs to be updated with their footprint. Moreover some of the existing apps do not follow IANA[2] port number assignment list to use port number which provides more complexity to the ecosystem. Besides, encryption is a trend to secure end-to-end communication and it makes performing NTC hard for those classifiers who relay on information in users payload. Last but not least the volume of traffic that NTC has to investigate is drastically increasing. Therefore, NTC should be fast enough to do the classification on-line which is an essential requirement for many NTC applications. in this thesis, I propose Chain as a novel algorithm to do NTC. Chain sequentially investigates different aspects of a network traffic and brings a significant improvement in tradeoff between classification performance and speed. Besides, it shows a great flexibility to adopt to the new network traffic due to its modularity design. I have implemented Chain in Traffic Identification Engine (TIE) [3] platform and have evaluated its performance with data set [4] which is published by CBA research group at Technical University of Catalunya. Following I have developed a platform named `GTEngin` to collect ground truth driven from mobile apps and then I have reevaluated the performance of my proposal with the new ground truth. In addition, I participated in an investigation carrying out on mobile Internet to study the possibility of improving my proposal performance in mobile ecosystem.Consequently, I leverage the result of the investigation and measure the enhancement of my proposal performance which achieved accordingly.

Contents

List of Figures

List of Tables

List of Abbreviations

NTC	**N**etwork **T**raffic **C**lassification
app	**app**lication
DPI	**D**eep **P**acket **I**nspection
ML	**M**achine **L**earning
CSP	**C**loud **S**ervice **P**rovider
LPI	**L**ightweight **P**acket **I**nspection
DNS	**D**omain **N**etwork **S**ervice
TIE	**T**raffic **I**dentification **E**ngine
ID	**I**ntrusion **D**etection
IoT	**I**nternet **o**f **T**hing

Chapter 1

Introduction

Network Traffic Classification (NTC) is a process of classifying network's traffic according to its corresponding applications (generally known as the app) which generates the traffic. There are several defined practical applications of NTC for Internet Service Providers (ISP), governmental agencies and end users which are described below:

1. NTC enables an ISP to detect the emerging application traffic requirements and develop its infrastructure accordingly.
2. Besides, ISPs can do traffic engineering and improve their Quality-of-Service.
3. Therefore, they can offer new services to their customers with a better network service pricing.
4. NTC can be used in the core of an automated intrusion detection system [5], in order to detect patterns of denial of service attacks or the users' suspicious activities and worms .
5. In addition, NTC can assist the automated intrusion detection system to reallocate the network resources dynamically based on the customers' priorities [6].
6. NTC is a fundamental component of ISP-based Lawful Interception (LI) solutions [7], since the governments enforce ISPs to provide LI of IP data traffic [8]. Therefore, the ISPs can report network activities of a specific user in a specific time upon the government's request.

1.1 Background

Many research works have been done to participate in NTC evolution, since two decades ago.[9] The majority of the works have been focused on an approach corresponding to one of the three main aspects of a flow. There is a brief introduction to each one of them in the following paragraphs .

1.1.1 Port-based Approach

Port-based is the first approach since early days of Internet. It classifies flows based on their destination's port number. Although it is a fast and a low memory/process consumption approach, it suffers from high false positive ratio due to many apps which do not have a registered port number like `Team Fortress 2` [10] or use 'well known' port numbers such as using ICMP proxies to bypass firewalls or even registration at hotels' hotspots [11]. This limits port-based classifiers accuracy from 30%

up to 70%[12]. Nevertheless, there are still some apps such as Email that have to use registered port number to communicate with different servers. Therefore, the port number is still valuable to classify specific apps.

1.1.2 Payload-based Approach

Payload-based classifier or Deep Packet Inspection (DPI) is another approach which implies string matching on packet payload to find apps' signature. It is known as the most accurate approach in the field and many publications ground truth are based on that[13][14][15][16]. Despite its high accuracy, it has three main drawbacks: (i) It is expensive in terms of memory and process consumption because it needs to store apps' signature in a data structure and also it requires checking each byte of packet payload exhaustively to find signature. Therefore, it is not practical to be used online for a high speed link. Many research works [17][18] [19] developed DPI implementation on hardware like NetFPGA [20] to meet the high speed link requirement for NTC. (ii) Looking inside the user data rises privacy concerns. (iii) It is incapable against encrypted flows and applications with no existing signature data such as evolving, polymorphic, and zero-day malware.

Although both port-based and DPI approaches have their limitations, they are still widely used in practice [21] as adopted by commercial products (e.g., Cisco NBAR[1], Ipoque PACE[2]) mainly to implement application-level firewalls. Specifically, many commercial solutions apply DPI to encrypted HTTPS traffic by adopting a man-in-the-middle [22] technique based on trusted SSL certificates (e.g., SonicWall DPI-SSL[3], A10 Thunder SSLi[4]). Within this approach, a DPI classifier is placed in the middle of communication of an end user and its corresponded server. Both end points establishes a SSL [23] tunnel to the DPI classifier and the classifier forward the traffic to either side. Therefor, the DPI classifier can decrypt the traffic and do its classification while the communication is protected by the encryption.

In addition, J.Sherry *et al.* proposed `BlindBox` [24] to use DPI directly on the encrypted traffic. `BlindBox` encrypts DPI rules with the end points public key and considers the encrypted rules to detect different apps.

However the other limitations of DPI still apply to these cases.

1.1.3 Statistical-based Approach

To overcome the above mentioned issues, recognizing apps based on the flow statistical fingerprints by means of Machine Learning (ML) has been the dominant trend in the field of NTC [7], [25]. ML classifiers automatically build a model to map flow-level statistical features to apps during a training phase and then use the model to classify new flows. As the resulting model is much smaller than the signatures used by DPI approaches and, its computational complexity is way lower than regular expression matching, it is more efficient in terms of memory and processing usage. In addition, since it does not require access to packets payload, it does not rise any privacy concern and remains practical against encrypted flows. However, real life tests show that it can recognise a few traffic categories and it is not accurate for mission critical applications.[26]

[1] http://bit.ly/cisco-nbar
[2] http://i-en.co.th/ipoque-pace/
[3] http://bit.ly/sonic-wall-dpi-ssl
[4] http://bit.ly/a10-thunder-ssli

1.1.4 Pros and Cons

Each approach offers several advantages while suffers from some drawbacks. Table 1.1 illustrates pros and cons of the classifiers developed based on the main approaches principle. The table compares compares the approaches considering the parameters mentioned below:

- *Accuracy* of classification
- *Cost* of classification in term of memory and processing consumption
- *Privacy* violation of users
- *Updating* a classifier to recognise traffic generated by new apps

TABLE 1.1: Strengths and weaknesses of classification relaying on the main approaches

	Port Number	**ML**	**DPI**
Accuracy	Low	Moderate	High
Cost	Cheap	Cheap	Expensive
Privacy	Safe	Safe	Violate
Updating	easy	Moderate	complex

Besides Accuracy and Cost parameters which have been discussed earlier, I would like to focus on Privacy and Updating aspects of different NTC approaches.

The user sensitive data is wrapped in packet's payloads and a DPI classifier relies on the payload information to perform the classification. Therefore, DPI is the only NTC approach which has access to the user's private data. However, there are several DPI based classifiers such as `PortLoad`[27] and `Libprotoident` [14] which intend to respect the user's privacy by limiting their inspections to just a small portion of the first payloads.

Many new apps are developed and installed by users daily all around the world. Consequently, a classifier is required to update constantly to recognise the new traffic types. Updating a port based classifier is as easy as adding a new entry to the table which maps a given port number to its corresponding app. However, the updating procedure is more complex for a classifier based on ML or DPI. A ML based classifier needs an accurate training dataset to update its model. A DPI classifier requires to be reinforced with the app signature of a new app to recognise its traffic. To this end, it is essential to collect a representative traffic set of the new app and extract a unique string from the traffic as the app signature. Following, signature database of the classifier should be updated with the new signature.

1.2 Objective

Considering the vital role of NTC in toddy's networks, I focus my effort to improve NTC in my PhD thesis. My initial hypothesis is that NTC can be enhanced by a composition of well-known approaches.

The objectives in order to validate my hypothesis are the following:

- Propose and implement Chain classifier
- Develop a platform to build a reliable ground truth

- Study environmental parameters that affect NTC performance

I designed a research plan consist of the following milestones to achieve the targeted objectives.

- Reviewing literature
- Designing and proposing my classifier named Chain
- Implementing Chain
- Evaluating Chain performance with publicly available ground truth (fixed traffic)
- Comparing Chain performance with its competitors
- Developing a platform to collect ground truth from mobile apps
- Collecting a first hand and fairly representative ground truth from the most popular mobile apps (mobile traffic)
- Evaluating the performance of Chain and its competitors with the ground truth collected from mobile apps
- Measuring the dependency of android apps on CSPs in the mobile internet ecosystem
- Studying the impact of traffic generated from communication between CSPs and mobile apps on Chain

Within the thesis, I explain the state-of-the-art classifier following the well-known approaches in Chapter II. I introduce related works combining different classification approaches and introduce my proposal (Chapter III). In addition, Chapter III contains performance evaluation of the proposal and a comparison between Chain and state of the art. Chapter IV introduces GTEngin platform and illustrates the proposed classifier performance in mobile ecosystem. I study mobile ecosystem to improve my proposal performance in Chapter V. Chapter VI illustrates the performance enhancement achieved in the mobile internet. Finally, there is a discussion session which discuss pros and cons of the proposed classifier and draw conclusions and future directions (Chapter VI).

Chapter 2

The State of the art

I select a set of classifiers as competitors to my proposal. These classifiers are either state of the art instances of the well-known approaches, or their properties are investigated comprehensively in many research works. The following chapter introduces the selected classifiers and reports their performance based on the ground truth which is summarised in table 3.1.

2.1 nDPIng

`nDPIng` [28] is the next generation of `nDPI` [26] which involved many data structure modifications to report different levels of a classification result in a consistent format. `nDPI` is developed to be: (i) reliable (ii) extensible to accommodate new protocols (iii) integratable with applications and kernel which are developed under open source license (iv) capable to extract basic network metrics and metadata such as network latency and DNS query/response. `nDPI` relays on the host name extracted from server certificate to classify an encrypted traffic. *e.g.*, an encrypted flow with server "api.twitter.com" is classified as `Twitter`. Although research works [29][30] confirmed the high accuracy of `nDPI`, its output has a limited usage. The output is a mixture of label from different levels which supposed to introduce a flow in the most detail way. Therefore, the output can be content type (Flash, MPEG, *etc*.), service providers (Twitter, Facebook, *etc*.), application protocol (DNS, HTTP, *etc*.) or even IP protocol (UDP, TCP, *etc*.). Consequently, `nDPI` can label a flow with its content type and labels the other based on its application protocol. `nDPIng` modifies `nDPI` data structure to return all the detected characteristics of an classified flow to provide more practical classification results which can be used by different network applications targeting a specific characteristic of flows [31].

2.2 C4.5

C4.5 is a decision tree algorithm which is developed to build a classifier based on nominal attributes. However, the continuous attributes are still applicable as C4.5 algorithm can convert them into the nominal by performing a binary split based on a threshold. The algorithm considers all possible thresholds for a continuous attribute and calculates its corresponded gain. The threshold with the highest gain will be the final candidate to perform the binary split. Therefore, it is compatible with both nominal and continues value attributes. C4.5 selects the attribute with the highest gain as a root of the tree and expends the three based on the other attributes information [32].

P. Perera *et al.* [33] and M. Shafiq *et al.* [34] measured the performance of different ML algorithms such as Bayes Net [35] and C4.5. Their studies confirmed the superiority of C4.5 among the selected ML algorithms.

In addition, [36] reported the high accuracy (97%) of C4.5 to detect P2P traffic based on the first 5 packets of a flow. Following, Y. Zhang *et al.* [37] skipped 10 to 1000 packets from the beginning of flows to study its impact on C4.5 performance by considering P2P traffic. The experiment showed that C4.5 is robust against such a distortion.

2.3 CoralReef

Using the port numbers to classify network traffic is the most straightforward approach. However, it is essential to relay on a up-to-date mapping table which reflects the latest modification of IANA [2] list to achieve the maximum classification gain reachable by a port based classifier. `CoralReef` is a network monitoring suit developed by CAIDA [38] to analyse network traffic. The database of `CoralReef` contains the most up-to-date list of port numbers assigned by IANA. Several research works [39][27][40] relay on `CoralReef` to measure the different aspects of port-based classifiers.

2.4 Testbed

`NeTraMark` [40] [41] and `Traffic Identification Engine(TIE)` are two platforms which are introduced to the community of NTC to implement and compare classifiers. Both of the testbeds intend to provide comparability, reproducibility, extensibility, synergy, and flexibility/ease-of-use to the researchers. Both of them offer several ready to use classifiers based on different classification approaches in format of plugins. Besides, the testbeds permit their users to implement their own classifier as a new plugin.

Although `NeTraMark` offers a user-friendly GUI, it does not support online classification. Furthermore, there is no other support to maintain it up to date neither from its development team, nor from the researcher's community. On the other hand, `TIE` has been involved in many enhancements since its creation date on 2009 by its original developers team as well as research communities such as `Universidad de la Republica of Montevideo` [42] and `University of Brescia` [43]. In addition, `TIE` has the capability of online classification functionality. Therefore, I consider it as the testbed for implementation and evaluation of my proposal and the state of the arts.

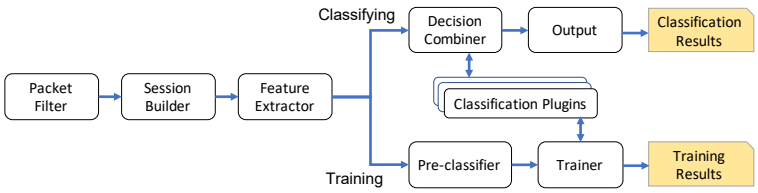

FIGURE 2.1: `TIE` Architecture[41].

`TIE` classifies network traffic in three different levels of granularity: app group, app and sub app or content. *E.g.*, `TIE` classifies a flow generated from Skype to

Conferencing, Skype and Image category as the group, app and content label respectively.

Also it is possible to run TIE depending on the traffic source and operation concern in offline, online and cycling mode. The latest mode performs the classification periodically in a user define interval time.

Figure 2.1 illustrates the block diagrams of TIE components. A user can run TIE either to train a machine-learning classifier, or to do the classification. TIE processes packets in five different stages which the last two stages vary depending on the objective of the user.

Packet filtering is the first stage of TIE process which captures or reads a raw packet (link layer frame) by means of Libpcap library[44]. In addition, it has the capability of filtering the data by both Berkeley Packet Filters [45] and user-space filtering rules like targeting traffic within a specified time interval.

Session builder stage splits traffic to sessions based on their five tuples and a user-define timeout (source and destination IP address and port number as well as transport protocol with 60 sec timeout) or their host. Host sessions consist of all income and outcome traffic of a host.

Feature extractor is invoked by Session builder upon arrival of a new packet and is in charge of extracting features which are required by classification plugins.

Considering classification objective, Decision combiner is the fourth stage of TIE process. This stage checks the availability of all features required for a list of selected classifier plugins to invoke them and collect their result consequently. Following, Decision combiner combines the obtained results based on different configurable combination rules. Finally, Decision combiner labels a session based on the outcome of the combination rule with a percentage of confidence which shows the reliability of the classification result.

TIE defines plugin as a format with standard interface for integration of new classification techniques. Therefore, Classification plugins are the classifiers which are encapsulated in TIE plugin format. The plugins are loaded dynamically on run time.

Output generator is responsible for creating an output file consists of the sessions information and their corresponded classification results.

However, the forth TIE stage would be Pre-classifier if the objective of process was focused on training. Pre-classifier loads labels corresponded to each flow from a ground-truth file.

Finally, Trainer triggers the signature collection function implemented in classifier plugins and permits the plugins to collect the information of each session in order to train themselves.

2.5 Overall Performance

Table 2.1 reports the measurement performance of state-of-the-art classifiers in terms of Recall and Precision (equation 3.4.1). I conduct the measurement on a server equipped with Intel(R) Xeon(R) CPU E5-2430@2.20GHz processor and 15 GB of RAM, running Linux Ubuntu 14.04.5 LTS. TIE is considered as the testbed of the measurement and the ground truth which is provided by Carela-Español *et al.* [4].

Considering the measurement outcome, nDPIng gains the highest Precision with a negligible cost of reduction in Recall in the most classes. Although C4.5 does not perform as well as nDPIng, it can be considered as a classifier for non-critical applications. CoralReef shows the lowest performance among the other classifiers.

Although it reaches the highest Precision and Recall in a couple of the app classes, it fails to classify the rest of the classes.

The measurement aims to illustrate an overview of the classifiers performance. There are more details about the classifier performance and the structure of the ground truth which can be found in chapter 3.

TABLE 2.1: Performance of state-of-the-art classifiers in terms of Precision (P) and Recall (R) percentages.

	CoralReef		**C4.5**		**nDPIng**	
	P	R	P	R	P	R
FTP (Control)	100	100	89	81	100	77
FTP (Data)	0	0	66	80	72	69
Flash	0	0	78	38	89	74
Emule	0	0	89	91	92	49
Bittorrent	100	3	86	96	100	94
Web (HTTPS)	100	100	92	96	100	94
Web (HTTP)	100	100	97	99	100	94

Chapter 3

Chain: A Network Traffic Classifier

This chapter provides a comprehensive explanation of Chain classifier's functionality and its implementation. Besides, I evaluate the performance of the proposed classifier and compare its results with the well-known approaches.

However, I consider similar classifiers in prior to my proposal and introduce them briefly in the Related Works section.

3.1 Related Works

There is a tradeoff between performance and resource consumption in the NTC field. In order to tackle this issue, a trend has arisen from multiple research studies: the proposal of various *combinations* of different approaches. Alcock et al. [14] proposed `libprotoident`, which is a Lightweight Packet Inspection (LPI) implemented as a C library. It inspects the first four payload bytes and the size of the first payload-bearing packet of each flow in both directions and takes into account also the destination port number. Several protocol identification rules are checked against all or part of extracted information in order to classify application flows. In this manner a significant amount of processing and memory consumption is saved in compared to DPI. However, the possible benefit of using port number information is not gained comprehensively, because it is exploited only for applications using well-defined ports such as port number 53 for Domain Network Service (DNS). In addition, since detection rules are checked before port numbers, the performance gain is sub-optimal and neglecting flow-level statistical information makes it still ineffective on encrypted traffic.

Several research works [46]–[48] use multiple classifiers in parallel and combine the results with different algorithms in order to obtain a final result(figure 3.1).

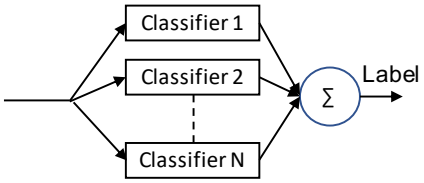

FIGURE 3.1: Multiple classifiers approach.

Specifically, Dainotti et al. [47] demonstrates how such approach leads to an improvement of the accuracy, but at the cost of increasing the processing costs linearly with the number of classifiers.

Foremski et al.[49] introduced the `Waterfall` multi-classifier, which consists of five cascaded classifiers (figure 3.2).

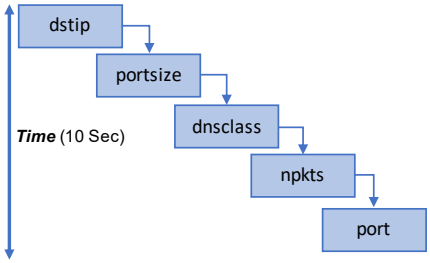

FIGURE 3.2: Multiple classifiers approach.

For each flow, classifiers criteria are checked against extracted features and more features get extracted until all classifiers are able to proceed with the classification. `Waterfall` relies on flow-level statistical information (i.e., packet size and inter-arrival time), destination port, and IP address, as well as DNS name associated with IP. Presented results show that `Waterfall` is quite accurate and it reduces processing consumption significantly, but due to the depth of the cascaded architecture it can introduce a significant delay for obtaining the classification result. Compared to `Waterfall`, my proposal is based upon a similar approach of cascading classifiers, but I limit the chain to just two stages, using different classification algorithms and features than `Waterfall` (ignoring IP address and adding a DPI-based stage). Moreover, I adopt a *principled approach* for the sequence of stages: while `Waterfall` classification sequence is not informed by decreasing complexity (the authors defer to future works the criterion for choosing a specific sequence of classifiers), I decide to employ the simplest (and fastest) methods first, as my goal is to improve the efficiency of DPI without compromising its state of the art classification performance.

3.2 How Chain Works

The overall idea proposed with Chain is a classifier architecture made of sequential stages, implemented with modules of increasing resource (time, computation, memory) cost. Each stage consists of one or more classifiers, which either identify flows precisely (with low rate of false positives), letting the flows out the chain, or passes them to the next stage. Therefore classification precision is the primary objective of each stage , and only low-confidence results will be passed on to subsequent(more resource-demanding) stages. In this work I propose a specific instance of the Chain architecture made of two stages. The first stage is a consensus-based multi-classifier, composed of a port-based classifier and a ML one. The result of this stage is a classification verdict only if both classifiers agree, otherwise the flow is passed on to the subsequent stage. The second stage is a DPI-based classifier. Figure 3.3 illustrates how network flows traverse Chain classifier components up to the end of the classification process.

Although port-based and ML classification approaches suffer from high rate of false positives, they are very fast classifiers. In considering them for the first stage, I am motivated by the expectation that the intersection of their false positives is relatively very small, because port numbers and flow-level statistic are two independent aspects of a flow and the concepts behind the two classifiers are highly different. I refer to the next section for the experimental validation of such expectation. Hence, the first stage of Chain exploits this property to quickly (in few packets) classify

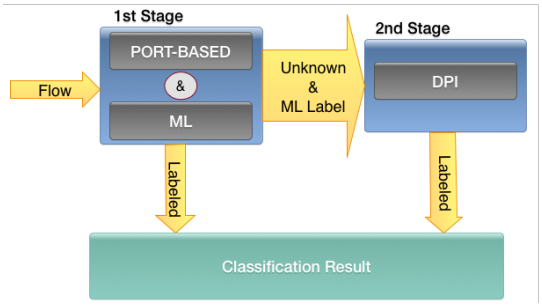

FIGURE 3.3: Chain classifier block diagram.

flows in case both classifiers agree. This way I am able to overcome the main weakness of port-based classifiers, i.e. the intentional adoption of well-known (usually unfiltered) ports to carry traffic of other applications.

The remaining unclassified (*Unknown*) traffic is passed to the second stage, implemented using a DPI approach, which is the most precise but most demanding in terms number of packets, memory, and processing. To improve Recall, the second stage considers the classification response of the ML module as *fall-back*, in case DPI is not able to issue a verdict. We have reported results proving the improvement in Table 3.2, in columns "DPI only" and "DPI+ML". Chain issues the verdict of the second stage without further investigation.

The following algorithm 1 explains the Chain classification procedure.

Algorithm 1 Chain classifier

```
1:  Port_label ← Port_based(Flow)
2:  while packets_arrived_in_each_directions ≤ 5 do
3:      Flow_features ← extract_features(packet)
4:  end while
5:  ML_label ← ML(Flow_features)
6:  if Port_label = ML_label then
7:      return Port_label                          ▷ Verdict of 1st Stage
8:  else                                           ▷ 2nd Stage
9:      repeat
10:         DPI_label ← DPI(Flow)
11:     until (packets_arrived ≤ 20 OR
12:             DPI_label ≠ Unknown)
13:     if DPI_label ≠ Unknown then
14:         return DPI_label
15:     else
16:         return ML_label
17:     end if
18: end if
```

3.3 Implementation

To implement a working prototype of Chain classifier I based it on the `Traffic Identification Engine (TIE)` [50], a well-known open source platform for implementing and evaluating network traffic classifiers. Due to its modularity design, `TIE`

provides a platform to fairly compare different NTC techniques, where new classification technique can be introduced as *plug-ins*, and *combined* according to a user-defined combination rule. Figure 3.4 shows TIE's logical blocks components. Some of the components are designed to be utilised just for scenarios [3]. TIE's combiner component is designed to combine several classification result in parallel scheme. Therefore I develop a new combiner specifically for Chain to follow its workflow. In addition, I develop and update the required plugins corresponded to the Chain modules. The modified and developed components are highlighted in the figure 3.4 with orange colure.

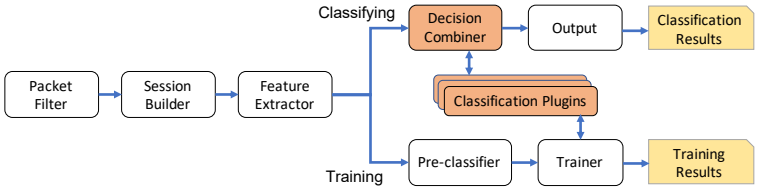

FIGURE 3.4: TIE Architecture.

The instance of Chain we propose in this work requires for its stages three classification techniques. As for the first stage, the port-based classifier is implemented via the `port` plug-in, which adopts the CoralReef [51] database. This is the only available plug-in that Chain uses without any modification. The ML classifier, instead, is an updated version derived from `TIEWeka`, a fork of `TIE` introducing a plug-in to stream flow features to Weka ML engine [52] and collects its results. As for the second stage, I developed a new plug-in based on `nDPIng` [28], which is the most advanced and up-to-date DPI algorithm publicly available, to the best of my knowledge.

The original `TIE` combiner invokes plug-ins and passes them the required subset of classification features without considering the results of other plug-ins (thus implementing a purely parallel architecture). Instead, Chain requires to invoke some plug-ins sequentially depending on the result of others. The TIE combiner has been modified significantly in order to implement the sequential Chain architecture and algorithm 1, instead of the original (purely parallel) one.

3.4 Evaluation

In order to investigate Chain classifier's performance, there is a need to implement and evaluate it. An existing NTC platform is selected and modified to implement Chain classifier. In addition, obtaining a ground truth is considered as a challenge for the evaluation. One of the main challenges which Network Providers establish strict rules for it is capturing traffic due to the privacy protection issue. The other challenge to face with is assigning labels to the captured traffic according to their app accurately.

3.4.1 Data and Tools

Carela-Español et.al [53] made an experiment to investigate the accuracy of ground truths obtained with DPI. Although it has been used on several research works [13], [49], [54]–[57], it reveals that DPI is unable to provide a reliable ground truth. Furthermore, using a ground truth generated with DPI would overestimate the precision in the case of Chain, since it includes a DPI stage. Based on these considerations,

TABLE 3.1: Structure of the ground truth in term of number of flows, number of packet and size of flows which are generated by different apps.

App	# Flow	# Packet	Size (byte)
HTTP	46336	4797697	4,091 M
HTTPS	8181	970354	728 M
Bittorrent	3187	677024	617 M
Emule	582	2003270	1,529 M
Flash	498	2676943	2,850 M
FTP-Data	366	988674	1,026 M
FTP-CTL	43	7146	184,351
Total	59193	12121108	10,841 M

I considered as ground truth the dataset provided by Carela-Español et. al [4], which contains three traffic traces labeled at their generation point. Each trace includes a PCAP file and a related info file including general metadata concerning the captured flows as well as the app labels. Table 3.1 reports a summary of the dataset.

The dataset has been filtered excluding unlabelled flows and divided in two sets to provide training and testing datasets for the ML classifier. Since flows were captured in sequences according to their app label, I have to modify the ordering of flows within the traces before the division. There are two rounds of evaluation used in this as the dataset is divided in two parts: the one used for training in the first round is used for testing in the second round and the other way round. In this way, both training and testing dataset contain an equal number of flow from each app. Therefore all the flows are used for the classification once. Moreover, to compensate for the unbalance among the classes, we adopted a classification comparison metric suitable for unbalanced datasets when comparing our proposal with the reference state-of-art.

I selected j48 (the Weka implementation of C4.5 decision tree) as ML algorithm because it demonstrated to achieve high performance [33][34]among the common ML algorithms in NTC field. Statistics (min, max, mean, standard deviation) of packet length and packet inter-arrival time of the first five packets in each direction are considered to train ML model and classify flows on-line. TIE has been used for features extraction of training data set which Weka uses it to build j48 ML model with its default parameter.

The evaluation is based on the well-known classification metrics:

$$Precision = \frac{Tp}{(Tp + Fp)} \quad Recall = \frac{Tp}{(Tp + Fn)}$$

$$Accuracy = \frac{Tp + Tn}{N} \quad F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

where Tp, Fp, Fn, and N stand for true positives, false positives, false negatives, and the total of samples, respectively. Finally, Area Under the Curve (AUC) [58] is used in classification analysis to determine which of the considered classifiers predicts better the classes (the higher the better, with 1 as maximum). prediction in compare with its competitors. AUC has been originally used in binary classification problems, but several works (e.g. [59]) extend the concept to multiclass problems via classifying each class versus all the others and averaging the AUCs derived for each class. I follow macro-average principle (equation 3.1) to measure the AUC of

each classifier to eliminate any bias toward classes with high number of instances. In equation 3.1, M is the number of app classes which are included in my dataset and AUC_i is AUC of $i-th$ class.

$$AUC = \frac{\sum_{i=0}^{M-1} AUC_i}{M} \tag{3.1}$$

The experiments were conducted on a server equipped with Intel(R) Xeon(R) CPU E5-2430@2.20GHz processor and 15 GB of RAM, running Linux Ubuntu 14.04.5 LTS.

3.4.2 Experimental Evaluation

My experiments shows that Chain successfully classifies 98% of the flows (recall) with 98% precision. It should be noted that 89% of the flows are classified by the first stage with 97% precision.

Table 3.2 reports the performance of the Chain classifier and its sub stages in more details. According to the table and focusing on Recall, although the first stage is unable to classify some apps (e.g., Bittorrent, for which only 2% of flows is classified, albeit correctly), it reaches high performance to classify a significant amount of flows generated by the other apps, such as Web and FTP (Control).

TABLE 3.2: Performance of Chain classifier (Overall) and its modules in terms of Precision (P) and Recall (R) percentages.

	1st-Stage		**2nd-Stage (DPI only)**		**2nd-Stage (DPI + ML)**		**Overall**	
	P	R	P	R	P	R	P	R
HTTP	100	99	74	1	45	1	99	100
HTTPS	100	89	100	8	85	8	99	96
Bittorrent	100	2	100	91	98	94	98	96
Emule	0	0	100	20	94	91	94	91
Flash	0	0	100	0	99	38	77	38
FTP-Data	0	0	100	9	95	80	95	80
FTP-CTL	100	74	100	7	43	7	90	81

According to the evaluation, considering the result of ML submodule in case of DPI substage failure (algorithm 1 at line 12) lead to significant improvement in the classification result of apps which Port substage classifies them with pure precision. My measurement shows that following this strategy enhances the average Precision and Recall of FTP (Data), Flash Content and Emule by 22% and 59% respectively. Consequently the overall Accuracy is increased by 2% and reaches up to 98%.

3.4.3 Experimental Comparison

Since the Chain classifier includes three well-known and state-of-the-art NTC approaches, I compare the performance of our proposed classifier with them individually: `CoralReef`, `j48` and `nDPIng`. Using the mentioned classifiers as competitors provide also the opportunity to compare both classification accuracy and time of stand-alone classifiers with their composition in the Chain schema.

Figures 3.5 and 3.6 show drastic fluctuations of Precision and Recall of stand-alone CoralReef classification results, confirming that in that form it is unpractical

for many real-life applications. However, it reaches the highest Precision and Recall for standard apps, which represent the majority of network traffic [60].

Comparing Precision of the considered classifiers in Figure 3.5 shows that j48 reaches the lowest precision after CoralReef. Also a closer look into the set of misclassified instanced which are classified by j48 and CoralReef with the same label, shows that their intersection is as low as 1.71%, confirming the expectation that motivated me in using their consensus as the first stage of Chain.

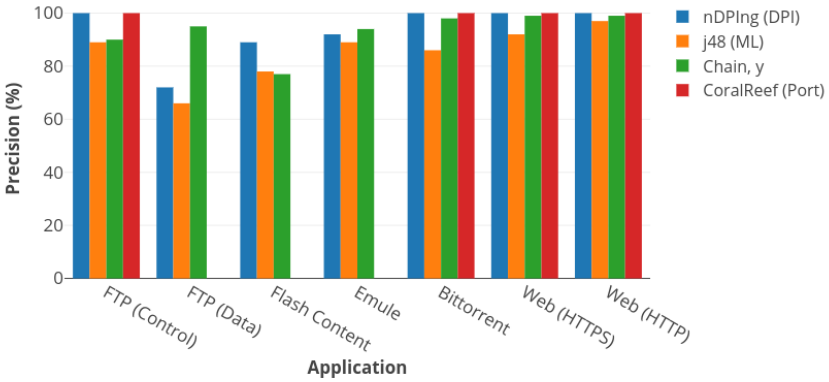

FIGURE 3.5: *Classification Precision of Chain and its competitors among different applications. The applications are ordered by increasing number of samples.*

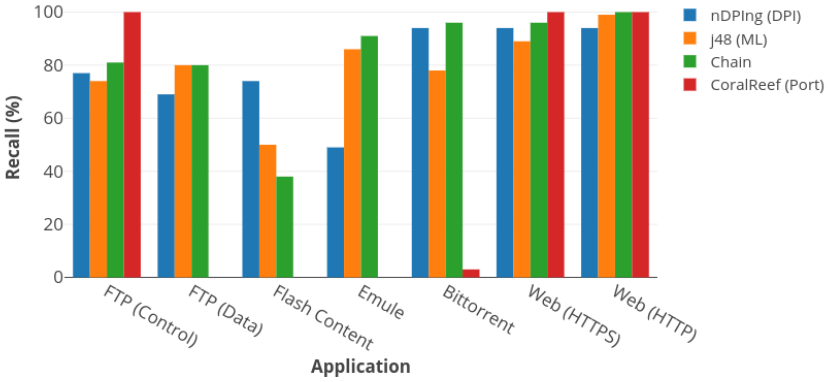

FIGURE 3.6: Classification Recall of Chain and its competitors among different applications. The applications are ordered by increasing number of samples.

As for nDPIng and Chain, there is a close competition between them to reach the highest overall Precision and overall Recall. Table 3.3 provides more details about such comparison, where both classifiers achieve approximately the same performance in terms of Precision and F1, with nDPIng being better by less than 2%, while Chain results superior in terms of Recall, Accuracy, and AUC (by 5%, 6%, and

7% respectively). This confirms the ability of Chain to keep classification performance comparable to stand-alone DPI (even slightly improving it for some metrics).

TABLE 3.3: Characterization of classification performance of Chain and nDPIng.

	Precision(%)	F1	Recall(%)	Accuracy(%)	AUC
Chain	98	0.98	98	98	0.91
nDPIng	99	0.96	93	93	0.89

As for the efficiency, I measure the overall time necessary to classify the whole dataset for all the considered approaches. I run the measurement 100 times for each classifier and report in Figure 3.7 mean and standard deviation of the resulting classification time. According to the results, as expected CoralReef and j48 respectively result the fastest classifiers, while nDPIng the slowest one. As expected, since Chain includes both CoralReef and j48 it results slower than them individually, but still 45% faster with respect to nDPIng. Such result demonstrates that Chain is able to improve the trade-off between performance and classification delay of DPI significantly.

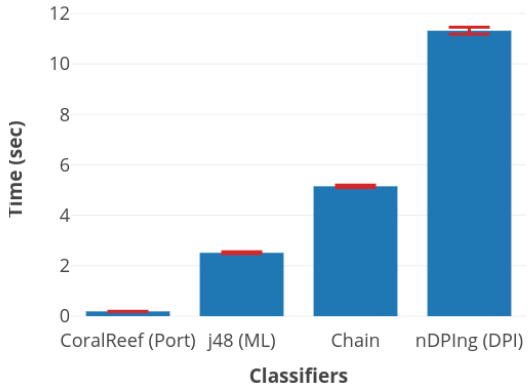

FIGURE 3.7: The average and standard deviation of classification time over 100 iterations.

Chapter 4

Mobile Traffic Classification

The previous chapter (chapter 3) introduced Chain and explained its functionality. Besides, I measured Chain performance and showed its superiority in compare with its competitors. However, the ground truth limits the performance evaluation to Internet traffic generated by fixed nodes or fixed traffic. Although fixed traffic contributes a considerable amount of network's traffic, the volume and number of mobile traffic [61] and apps [62] increases dramatically daily. The following chapter studies the classification performance of Chain and the instances of the well-known classification approaches in mobile Internet. Obtaining a fairly representative ground truth collecting from mobile apps is the first challenge which should be faced.

4.1 GTEngin: the Ground Truth Builder

In section 4 chapter 3 I mentioned that the privacy concern prohibits ISPs to share their users traffic with third parties. The nature of mobile devices which attaches them to their user personal life, includes private data to the mobile traffic more than ever. Therefore, ISPs do not publish their traffic publicly without any significant anonymization to preserve their users' privacy which make it unpractical for DPI performance evaluation.

Nevertheless, a ground truth from mobile Internet is an essential ingredient for studying the classifiers performance with mobile traffic. Therefore, it is necessary to develop a platform to generate and capture mobile traffic from different mobile apps on multiple mobile devices. Furthermore, the platform should label the captured traffic with their apps' name to be applicable for NTC.

I have proposed and implemented an architecture called GTEngin to answer the need based on inspiration from Ciuonzo *et al.* work [63]. Ciuonzo *et al.* structure supports just one android device while my proposal is scalable and can support multiple android devicess. `GTEngin` controls multiple android devices and run randomly different apps an each of them simultaneously. In parallel, `GTEngin` captures the generated network traffic from the router which connects the devices to the network. Also `GTEngin` records the network activity logs of the devices to extract the final label at the end of the process.

Figure 4.1 shows that `GTEngin` uses `Monkey Tool` [64] to play with an app in a device. `Monkey Tool` generates pseudorandom stream of events to manipulate an app which leads to network traffic generation.

A router with DDWRT [65] platform has the capability to accept remote command like `tcpdump` [66]. `GTEngin` runs `tcpdump` on the router at the beginning of traffic generation process to capture the traffic and forwards them to the local by means of `NetCat` [67].

I relay on "Network Log" to collect information about the process which generates the traffic. "Network Log" provides a unique opportunity to monitor all network activities running on an android devices. I record the information which is monitored by "Network Log" to label the captured traffic later. The source, destination IP and port number of a flow as well as its transport protocol are used as identifier to map a captured flow with its corresponded process's name existing in "Network Log" information.

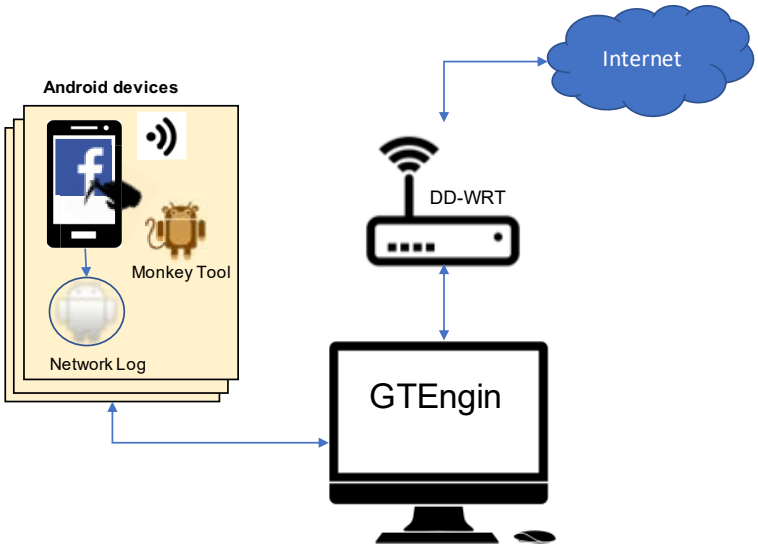

FIGURE 4.1: Diagram of GTEngin structure.

`GTEngin` process consist of the following stages:

- `GTEngin` runs `Network Log` on all the attached devices and executes `tcpdump` on the router in the initialisation phase. Besides, a `NetCat` connection is stablished between the local node and the router in order to forward the `tcpdump` (raw packets) output to the local node.
- `GTEngin` selects an app from a list of predefined apps for each of the devices and run them simultaneously on the devices. The process continues until all of the apps in the list get executed once on each device.
- `GTEngin` terminates `tcpdump` and `NetCat` connection. `GTEngin` saves captured traffic in a pcap file format locally.
- I label the captured file with the help of "Network Log" information after `GTEngin` accomplishes his task.

4.1.1 GTEngin Setup

Although, `GTEngin` can support multiple android devices, the available devices limit me to include just two devices in the process of traffic generation. The utilised devices are: Smartphone Samsung Galaxy S5 (SM-G900F) with android version 6.0.1 and tablet Samsung Galaxy Tab S2 with android version 7. Also I use a 13" MacBook pro with 2,5 GHz Core i5 and 8 GB DDR3 as a host for `GTEngin`. I select 15 android apps that are among the most popular apps reported by "ANDROIDRANK" [68] on October 2018. Table 4.1 contains the list of the selected apps.

TABLE 4.1: List of selected apps for traffic generation process

App	Packagename	Category
mytalkingtomfree	com.outfit7.mytalkingtomfree	Casual
spotify	com.spotify.music	Music & Audio
subwaysurf	com.kiloo.subwaysurf	Arcade
youtube (https)	com.google.android.youtube	Video Players & Editors
candycrushsaga	com.king.candycrushsaga	Casual
DU Battery	com.dianxinos.dxbs	Tools
instagram	com.instagram.android	Social
eightballpool	com.miniclip.eightballpool	Sports
snapchat	com.snapchat.android	Social
ucmobile (http)	com.UCMobile.intl	Communication
messenger	com.facebook.orca	Communication
facebook (https)	com.facebook.katana	Social
android.gms	com.google.android.gms	Tools
clashofclans	com.supercell.clashofclans	Strategy
clashroyale	com.supercell.clashroyale	Strategy
ucmobile (https)	com.UCMobile.intl	Communication

Multiple apps in the table 4.1 such as `facebook` and `instagram` fall in the social category and they require a registered user to login prior to be used. Therefore, I create distinct test profiles for each of them on each device and do the login before starting `GTEngin`.

Also, `GTEngin` provides the possibility to customise the sequence of events generated by `Monkey Tool`. I inclose `Monkey Tool` to operate in a selected app by skipping system key event such as pressing Home and Back bottom [64]. In addition, I run `GTEngin` five times and adjust the delay between the stream event to 250, 500, 750, 1000, 1500 milli seconds each contains 1000 events.

Following the mentioned setup, I succeed to collect 11,316 number of flows generated from the most popular android apps. Table 4.2 reports the structure of the ground truth that I have obtained at the end of the process.

TABLE 4.2: Structure of the ground truth in term of number of flows, number of packet and size of flows which are generated by different app.

App	# Flow	# Packet	Size (byte)
mytalkingtomfree	1477	29,909	11,2M
spotify	560	11,890	3,7M
subwaysurf	477	9,611	3,6M
youtube (https)	427	9,278	2,9M
candycrushsaga	360	7,310	1,4M
DU Battery	306	6,524	2M
snapchat	248	5,458	1,7M
instagram	236	4,726	1M
eightballpool	207	4,095	1,7M
messenger	182	3,630	0.7M
facebook (https)	150	3,080	0.5M
android.gms	120	2,661	0.8M
clashofclans	84	1,746	0.7M
clashroyale	73	1,445	0.5M
ucmobile (https)	50	1,079	0.2M
ucmobile (http)	43	1,181	0.08M
Total	5,000	103,623	32.68M

4.2 Mobile Traffic & Chain Performance

I use the same configuration as section 3.4.1 to evaluate the performance of the proposed classifier and its competitors with the ground truth collecting from mobile apps.

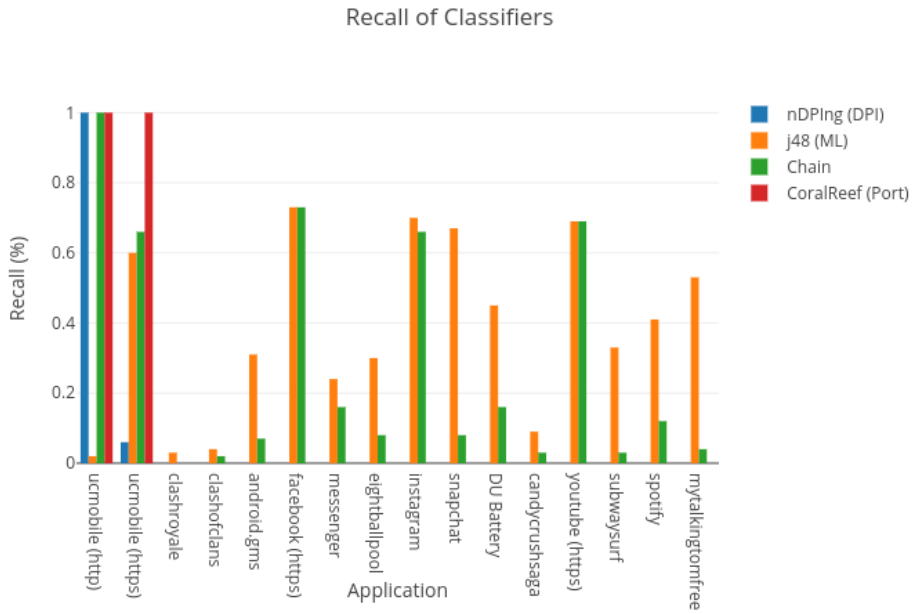

FIGURE 4.2: Classification Recall of Chain and its competitors among different applications. The applications are ordered by increasing number of samples.

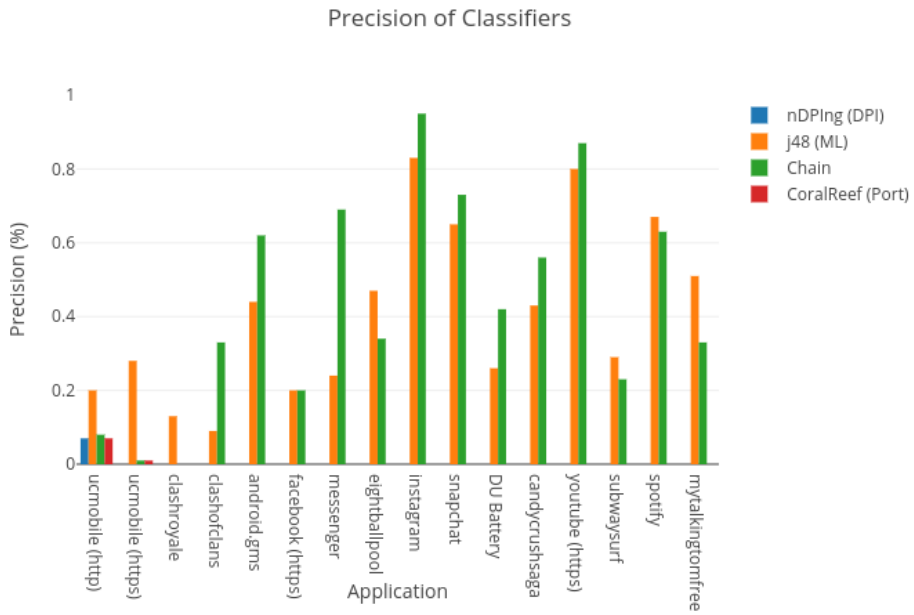

FIGURE 4.3: *Classification Precision of Chain and its competitors among different applications. The applications are ordered by increasing number of samples.*

Figures 4.2 and 4.3 illustrate performance of the classifiers to classify different apps considering Recall and Precision metrics. As it is expected, the `CoralReef` 's performance which relays on the port number information, is unacceptable. `CoralReef`

identifies flows just from two apps with low precision. According to the figures nDPIng also performs as unacceptable as CoralReef. Outdated signature database is the reason behind of nDPIng 's bad performance. In fact, nDPIng apps' signatures are not updated since 2014 and apps have been involved in many changes since then. *E.g.*, Razaghpanah and *et al.* studied TLS[69] usage in android apps and reported that 84% of the apps relay on default OS APIs to encrypt the user communications [70]. On the other hand, android has canceled SSL[71] support and SSL was substituted by TLS since android 6 [72]. This explains the low performance of nDPIng to detect HTTPS traffics.

In addition, there are apps in the ground truth which were developed after 2014 and its signature is missing from nDPIng signature database. Figure 4.4 is a heat map representation of confusion matrix of nDPIng classification result. The heat map shows that nDPIng fails to identify the majority of the flows.

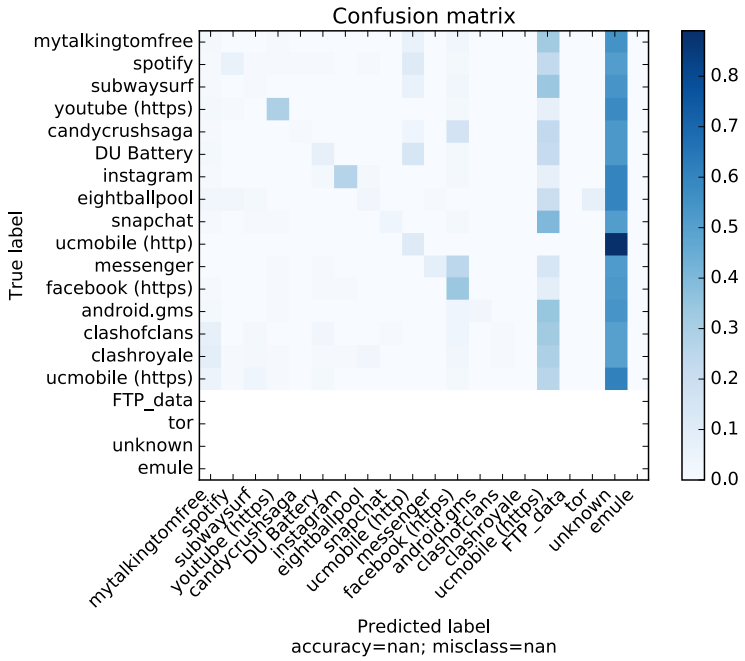

FIGURE 4.4: nDPIng confusion matrix

However, figures 4.2 and 4.3 illustrate that j48 outperform among all the classifiers. In fact, j48 adopts to the new environment faster than nDPIng and training its model with a new ground truth is much easier than to update nDPIng signature database. However, it classifies just 45% of flows with 49% precision which is not sufficient for many real life applications.

Considering the figures, Chain follows j48 performance in most of classes. However, Chain reaches the highest or equal precision in compare with j48 in the majority of the classes. Although Chain relays on nDPIng as its core module, its performance is not limited within Chain signature database. In fact, the *fall-back* 1 mechanism bypasses the nDPIng drawback. Therefore, Chain is more flexible than nDPIng to adopt to the mobile traffic thanks to its modularity design.

There is an inconsistency of classification performance measured in this chapter and chapter 3. Performance of most of the classifiers dropped drastically when they classify mobile traffic. This motivates me to take a closer look in to the issue by studying mobile Internet ecosystem. Next chapter is dedicated to study the relationship between mobile apps and cloud services as one of the key players in the mobile Internet ecosystem.

Chapter 5

Mobile Ecosystem

Mobile applications typically connect to two types of on-line services: first-party services controlled by application developers, and third-party services integrated in mobile apps for advertising and tracking purposes [73], or to embed other services like online payment and weather reports [74]. These third-party services may also rely on CSPs for outsourcing their cloud infrastructure. Vallina-Rodriguez *et al.* have shown that third-party traffic dominates all app traffic in the mobile ecosystem [73], demonstrating the importance of studying this type of mobile traffic in addition to first-party traffic. Besides, connecting to third party services generates network traffic which is not statistically unique to the app. Therefore, these traffic can be considered noise for NTC relaying on ML approach. Consequently, it is essential to identify and filter out the third party traffic before doing NTC.

The Flipboard app leverages Facebook's Graph API, which is hosted in Facebook's own cloud infrastructure, for user login and possibly for advertising purposes. Figure 5.1 shows the Flipboard reliance to CSPs as an example. Using the method which is described in the following sections, allows me to know that the Flipboard app communicates with 5 different FQDNs, and that each contacted domain is hosted in a different CSP.

The Flipboard is a typical example of mobile app to illustrate the tight relationship between a mobile app and CSPs.

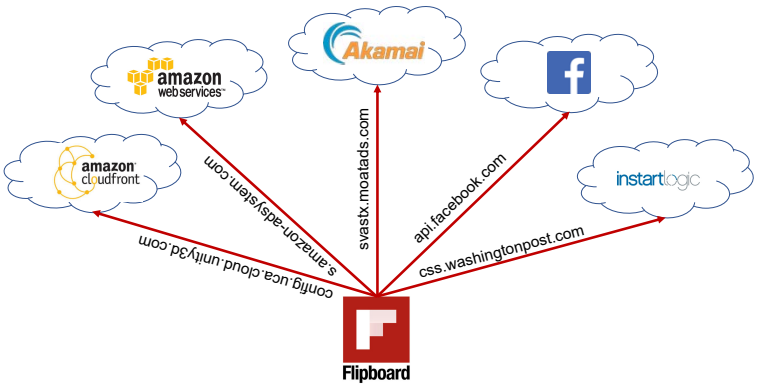

FIGURE 5.1: A simplified case of the Flipboard app demonstrates the network domains reached by the app (each red arrow represents a traffic flow to a domain).

5.1 Datasets

Lumen Privacy Monitor is a mobile application, available for free on Google Play [75], that aims to promote mobile transparency and user awareness. Users use Lumen to identify data leaks and the presence of online tracking services on their apps. Lumen

runs locally on the device and intercepts all network traffic without requiring root permissions by using the Android VPN permission. In a nutshell, Lumen inserts itself as a middleware between apps and the network interface. By operating locally on the device, Lumen is able to correlate disparate and rich contextual information such as process IDs, with flows. For example, Lumen can match DNS queries to outgoing flows and the process owning the socket. This feature makes Lumen a unique mobile vantage point to understand how mobile apps communicate with online services with real user input and network-stimuli "in the wild". I leverage the data provided by Lumen to identify the main third-party services providing support to mobile apps and mobile-related online services.

Since October 2015, Lumen has been installed by more than 1,600 real users. This user base has allowed me to have access to over 5,319,150 anonymous network flows corresponding to 8,281 different mobile apps connecting to 36,918 unique FQDNs and 135,325 unique IP addresses. The dataset contains both IPv4 and IPv6 flows (7.5 % of total flows). Finally, I discard 1.5 % of the flows connecting to IP addresses reserved by IANA for private networks and CGNs [76], [77] which may be associated to P2P traffic and home networking activity. 97% of the DNS responses were provided by Google's DNS.

The analysis of mobile traffic generated by real human action poses ethical challenges. In order to preserve user privacy, Lumen performs its flow processing and analysis on the device, only sending anonymised data such as protocols used, domains contacted, and type of privacy leaks identified to Lumen's servers for research purposes. No traffic payload and any form of identifier is collected. I also exclude browser traffic as it may reveal user activity patterns, hence likely easing de-anonymisation. Due to these precautions, the Lumen developers institutional IRB has considered this project as "non-human research subject" as they analyse the behaviour of software, not people.

To complement my measurements and datasets, I use a number of external resources. I use Maxmind GeoLite2 database geo-IP database for IP geolocation. I also leverage `censys.io` IP scans to contextualise and validate observation observed in certain IP machines. Furthermore I use the domain-CDN mapping provided by CDNFinder [78] and the wptagent project [79] as a starting point to identify `CSP`-hosted domains.

5.2 Method

To the best of my knowledge, no study has exhaustively analyzed the close relationships that exist between mobile applications and `CSP` services. The first problem I encounter in my effort is knowing which domains are hosted in third-party `CSP`s.

5.2.1 Leveraging reverse DNS lookups

Finding `CSP`s from `PTR` records: In order to identify the `CSP` providing support to a given FQDN, I analyse the `PTR` records associated with each one of the IPv4 and IPv6 destination IP addresses provided by the Lumen dataset. I run reverse DNS lookups to extract the `PTR` record, if available. This method allows me to map FQDNs to `CSP`s via their `PTR` records. In total, my 18,272 FQDNs map to 2,995 second-level `PTR` records. I could not harvest any `PTR` records for 37% of `FQDN`. I first check the presence of the gathered `PTR` records on well-known CDN-domain maps like CDNFinder [78]. However, these public maps are CDN-specific, they are incomplete (*e.g.*, they do

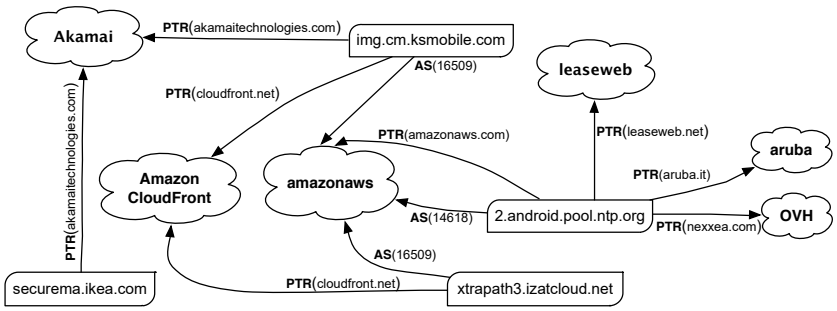

FIGURE 5.2: Combination of ASN and PTR approaches to detect CSPs

not consider unpopular CSPs like cdnetworks) and they contain domains such as doubleclick.com–arguably a CDN provider. I sanitised these public mappings to obtain a list of 145 second-level domains associated to 82 CDN providers.

Finding CSPs using URL classifiers: To extend their coverage to any type of CSP provider as well as marginal CSPs, and also to minimise the false positive rate, I leverage both McAfee [80] and OpenDNS [81] domain classifiers. First, I extract the categories assigned by those services to well-known CDN providers: mainly "Internet Services" and "Content Server" and then I search for other domains, ignored by CNDFinder, falling in these categories. Unfortunately, URL classifiers provide vague categories (*e.g.*, ad networks are also classified as "Internet Services") which can introduce false positives in my list.

Manual Inspection

The scale of the set of PTR records does not allow me sanitising all of them manually to identify such errors. As a result, I limit my manual inspection–which relies on the information provided by WHOIS queries and censys.io [82]– to 248 popular domains reached by at least three mobile apps and those containing the keywords "host" and "cdn" on their names. The combination of these methods has allowed me to identify 194 second-level PTR records associated with 125 third-party CSPs. Only 43 of them were present on CDNFinder. my CSP domain list is publicly available at A.1 and publicly at [83].

5.2.2 Leveraging AS information

Following reverse DNS lookup approach, CSPs provide services to 39 % of FQDN and their 54,996 IPs are spread among 217 Autonomous Systems. Caida classifies 74 % of the ASs as "Transit/Access".

Further investigation by considering ipinfo.io information shows IP blocks of 33 (15 %) of these Autonomous System are entirely dedicated to a given CSP. Therefore I consider FQDNs that resolve to one of the late AS are served by its corresponded CSP. This approach ables me to analysis FQDN without any PTR record and to increase 14% the identification coverage of FQDN associated to CSPs. Table 5.1 reports Caida classification result of the AS and indicates more than half of them are Content. A list of ASs and their corresponded CSP is available at A.2 and publicly at [83].

Figure 5.2 shows various relationship between CSPs and FQDNs. Also it explains the combination of approaches to detect CSPs.

AS Type	Volume (%)
Content	58.82
Transit/Access	20.58
Enterpise	17.64
N/A	2.94

TABLE 5.1: *Classification of 33 ASs which entirely belong to a given CSP according to Caida database*

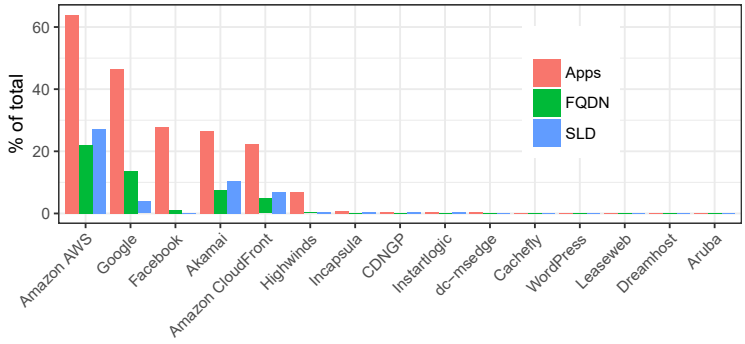

FIGURE 5.3: Top 15 CSP by app usage. We show their domain penetration for reference, both at the FQDN-level and second-level domains (SLD).

5.2.3 Method Limitations

Many CSPs offer several services such as CDN or cloud services to their customers. The main challenge that my technology faces with is to distinguish between CDN and cloud services. However this the first study to look at CSP space and it can be consider as research field for future work.

5.3 CSP Presence on Mobile Apps

I represent in a graph the relationships found between mobile applications, FQDN and PTR records as recorded by Lumen and my reverse DNS lookups. This representation allows me to have a comprehensive view of how mobile applications leverage online third-party service providers, and also identify those online services implementing multi-CDN strategies. 85 % of the apps in the Lumen dataset connect at least to one of the 55 CSPs. However, five CSPs play a central role on running mobile Internet services. As we can see in Figure 5.3, over 64 % of the apps connect at least to one service hosted on Amazonaws. 49 CSPs have a marginal presence in this market, receiving connections from less than 1% of the apps.

Figure 5.3 concludes that Amazon dominates the market by Amazonaws and "Amazon CloudFront". Also 28 % of APKs communicate with facebook. According to the hosting infrastructure analyse"Facebook " CSP supports only its domains as well as its affiliates such as "Instagram". In addition considering that request to FB's Graph API is resolve to the same domain (fbcdn.net) I can conclude many APKs contact facebook domain for non CSP services.

Difference of CDNs coverage and diversity of CSP services in general is a strong motivation for CSP customers to enlist multiple CSP to enhance the reliability and scale of their services. According to Table 5.2 "Facebook" is a popular example of such a customer that utilises "Akamai" next to its own developed CSP infrastructure.

#app (%)	Domain	IP(%)	CSP
29.43	googlesyndication.com	0.81	Google
28.47	doubleclick.net	1.44	Google
26.84	facebook.com	0.59	Facebook Akamai
25.77	crashlytics.com	1.41	amazonaws
24.96	googleadservices.com	0.48	Google

TABLE 5.2: *Top 5 domain based on their app penetration*

num-CSP	FQDN (%)
1	98.94
2	1.04
3	0.02
4	0.01

TABLE 5.3: *CSP penetration by FQDN*

However relaying on multiple CSP arguably is a cost-sensitive decision which justify its power law distribution in Tables 5.3 and 5.4.

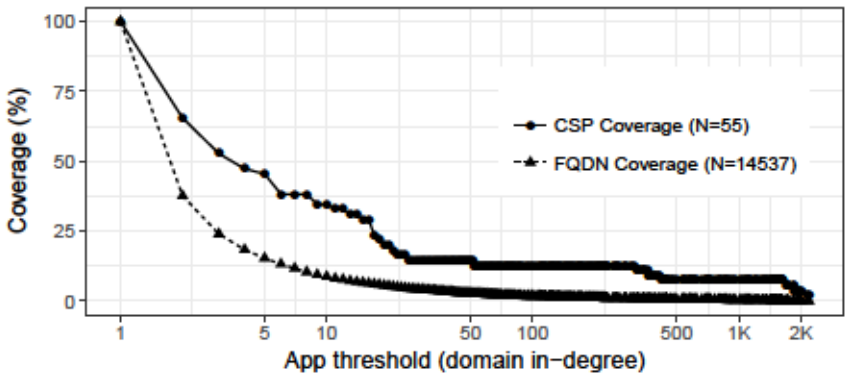

FIGURE 5.4: CSP and domain coverage as a function of domain popularity (*i.e.,* domain-in degree: number of connecting apps).

Furthermore I study the distribution of FQDN on CSP. There are 14,537 FQDN with 4,178 Second Level Domain (SLD) that communicate with 55 CSPs. For the study I focus only on Second Level of FQDN (SLD) which are associated with CSP communication. According to figure 5.4 27% of considered SLD are used to communicate with Amazonaws.

Analysing the most app penetrated domains reveals that ad related domains are at the top of the list. Table 5.2 shows that ad related domains such as " doubleclick.net"and "googlesyndication.com" have the highest app penetration rate.

num-CSP	SLD(%)
1	84.9
2	13.61
3	1.33
4	0.12
5	0.02
7	0.02

TABLE 5.4: *CSP penetration by Second Level Domain of FQDN*

The mentioned domains are used to store ad or related content for " Google AdSense" and " DoubleClick". The fact can explain their equal app penetration.

5.4 The Study Outcome

In this chapter, I performed the first holistic analysis of the complex ecosystem formed by mobile apps and cloud service providers (CSPs). I aimed to comprehensively characterise their relationships. I leveraged accurate traffic fingerprints from thousands of mobile apps that was collected through crowd-sourcing with Lumen [84]. This data allowed me to identify the most relevant CDNs and cloud providers for mobile traffic. The results show a significant reliance of apps on mobile CSPs with the major CSPs being used by 85% of the apps. Therefore using DNS is not reliable in NTC filed. fact [85]

The tight relationship between mobile apps and CSPs as well as the fact that a limited number of CSPs are dominated the market, impose a new challenge in to NTC. A single CSP can support many apps from different category and provide them similar resources. For *e.g.*, several apps with different functionality can use the same ad network to improve their revenue. In these case, the apps generate flows with the same network footprint to connect to the ad network. Therefore the false positive rate can increase and affect the NTC classifier performance.

P. Foremski and *et al.* proposed DNS-Class classifier based on DNS information at [85]. They demonstrate that DNS-Class can classify 1/3 of their dataset. The study of the mobile ecosystem indicates the coverage of DNS-Class can be reduced even more in the ecosystem. Because third party connection is dominated the mobile apps communications. In addition different mobile apps can hire services from the same popular CSP.

This chapter concludes that third party communication brings more challenges to NTC in mobile internet which consequently can alter the performance of classifiers rely on ML approach or DNS information.

Chapter 6

Influence of CSP on NTC Performance

The study of mobile Internet ecosystem illustrates that there is a strong reliance between mobile apps and CSPs like CDNs. In addition, the study proofs that a limited number of CSPs dominate the ecosystem. Therefore, it is possible that several apps with different functionalities are supported by the same CSP. Indeed, Table 5.2 reports that several apps relay on DoubleClick to deliver advertisement and monetise themselves. Such a communication generates a traffic from various apps with the same statistical pattern despite their apps functionality. This mixture of the network traffic increases the ambiguity to do NTC using ML approach. Therefore, CSP related traffic can be considered noise for the ML approach. I study the effect of CSP traffic on NTC in the following chapter by removing the CSP traffic from the ground truth and measure the precision of j48 as well as Chain with the new ground truth.

6.1 Filtering out CSP related Flows

I follow the same approach as described in Chapter 5 to detect CSP related traffic and then filter them out from the ground truth collected with GTEngin. The ground truth contains 742 unique IP addresses which I could not harvest PTR record for 25% of them. I relay on Autonomous System (AS) information for those IP addresses without PTR record. During filtering process, 69% of the flows are detected as CSP related traffic. 30% of the detection are based on the AS information.

TABLE 6.1: Detected CSPs and their distribution on the ground truth

CSP Name	Distribution (%)
Google	40.08
Facebook	21.95
Amazon AWS	20.96
Amazon CloudFront	8.39
Akamai	7.1
CDNGP	1.15
Highwinds	0.33
BunnyCDN	0.05

Table 6.1 reports distribution of detected CSP on the ground truth. The table confirms the domination of limited number of CSPs in the mobile Internet ecosystem which was already shown in Chapter 5, figure 5.3. The consistency between table 6.1 and figure 5.3 confirms that GTEngin succeed to provide a fairly representative ground truth for NTC.

Table 6.2 summaries the distribution of CSPs traffic generated by considered app in the ground truth.

TABLE 6.2: App reliance to CSP services considering the traffic generated by apps

App Name	CSP Reliance (%)	CSP Provider(s)
youtube (https)	100.0	Google, Amazon AWS, Akamai
spotify	61.35	Google, Akamai, Facebook, Amazon AWS Highwinds
android.gms	100.0	Google
instagram	89.73	Facebook
clashroyale	100.0	Amazon AWS, Facebook, Amazon CloudFront Google
snapchat	100.0	Google, Amazon CloudFront, Amazon AWS
candycrushsaga	7.76	Amazon CloudFront, Facebook, Akamai
messenger	100.0	Facebook, Amazon AWS, Amazon CloudFront
facebook (https)	97.51	Facebook
subwaysurf	82.72	Amazon AWS, Facebook, Amazon CloudFront Akamai, Google
DU Battery	55.08	Facebook, Google, Amazon AWS, CDNGP Amazon CloudFront, Highwinds
clashofclans	100.0	Amazon AWS, Google, Facebook Amazon CloudFront
ucmobile (http)	0.0	
eightballpool	49.61	Amazon AWS, Facebook, Amazon CloudFront
mytalkingtomfree	69.75	Akamai, Google, Amazon AWS, CDNGP, Facebook Amazon CloudFront, Highwinds, BunnyCDN
ucmobile (https)	0.0	

According to table 6.2, all the considered apps except ucmobile relay on CSPs services. In addition, most of the studied apps follow multiCSP strategy to stay close to their users' expended all around the world. In addition, most of the app under the study prefer to outsource their services and relay on multiple CSPs infrastructure to serve their users.

Furthermore, it is expected that apps like `mytalkingtomfree` and `DU Battery` follow multi-CDN strategy as it communicate with multiple CDN providers. However, it is not possible to make a strong conclusion wether the app itself follow the strategy or a service that it hires from a CSP causes the communication to different CDN providers. To clarify the issue, it is required upgrading `GTEngin` to record FQDNs.

Relaying on destination IP address information (such as PTR record and AS information) to identify a flow as a CSP traffic, can not reveal the service that a CSP provides for an app. Therefore, significant amount of traffic generated from apps developed by a given CSP (e.g. `instagram`, `messenger`) is identified as CSP traffic and filtered from the ground truth. Identifying CSPs services that a flow carries, is beyond the scope of my thesis and can be considered as a future work.

Following, table 6.3 reports the remaining traffic after filtering out the CSP related traffic.

TABLE 6.3: Structure of the ground truth without CSP related traffic in term of number of flows, number of packet and size of flows which are generated by different apps.

App	# Flow	# Packet	Size (byte)
mytalkingtomfree	382	7822	2,7M
candycrushsaga	330	6629	1,2M
spotify	221	4875	1,7M
DU Battery	134	2795	0.9M
subwaysurf	84	1648	0.6M
eightballpool	79	1707	0.5M
ucmobile (https)	50	1079	0.2M
ucmobile (http)	43	1181	0.08M
instagram	30	644	0.2M
Total	1353	28380	8.08M

Excluding CSP related traffic leads to shrinking the ground truth size significantly. Consequently, considering parameters such as recall is not fair to evaluate the influence of CSP's traffic on classifiers using the ML approach. Therefore, I focus on precision alternation to study the effect of CSP related traffic on NTC.

I use the same configuration setup as described in section 3.4.1 to do the study with the ground truth summarised by table 6.3.

Figure 6.1 clearly states that filtering CSP's related traffic enhances precision in most of the classes. However, in couple of app classes the performance does not follow the trend. A learning model from imbalanced training dataset is always biased toward the classes with the majority instances. Filtering CSP related traffic significantly reduces the instagram instances from 236 to 30. Although j48 is more robust than other conventional ML algorithms in dealing with imbalanced training dataset [86], it affects the classifier performance in extreme cases like instagram.

The high recall and low precision of ucmobile using CoralReef classifier in figures 4.3 and 4.2 indicated that other apps also generate HTTP and HTTPS traffics with the same statistics as ucmobile. Therefore, this reduced the contrast of traffic generated from ucmobile and the other apps. Consequently, the reduction in the contrast causes reduction on the precision of ucmobile classifying by j48. Several apps and many flows are removed from the ground truth by filtering CSP related traffic. This improves the contrast of traffic generated from ucmobile's and other apps. Figure 6.1 shows that the contrast improvement enhances the precision of ucmobile classification significantly.

6.2 Performance Measurment

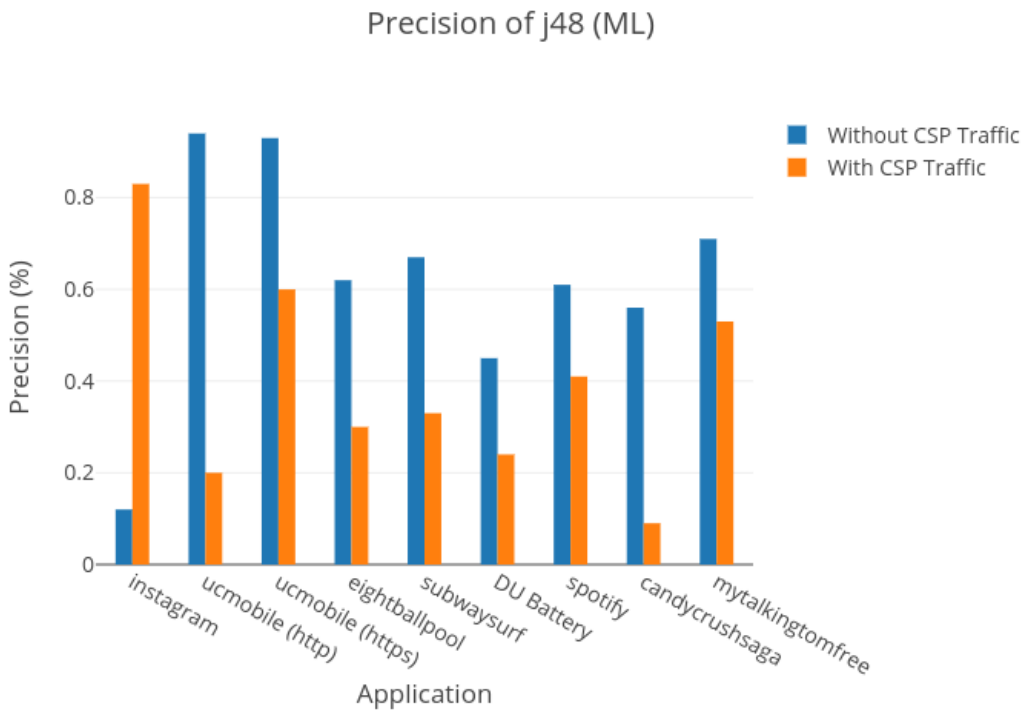

FIGURE 6.1: Precision of j48 with and without CSP traffics

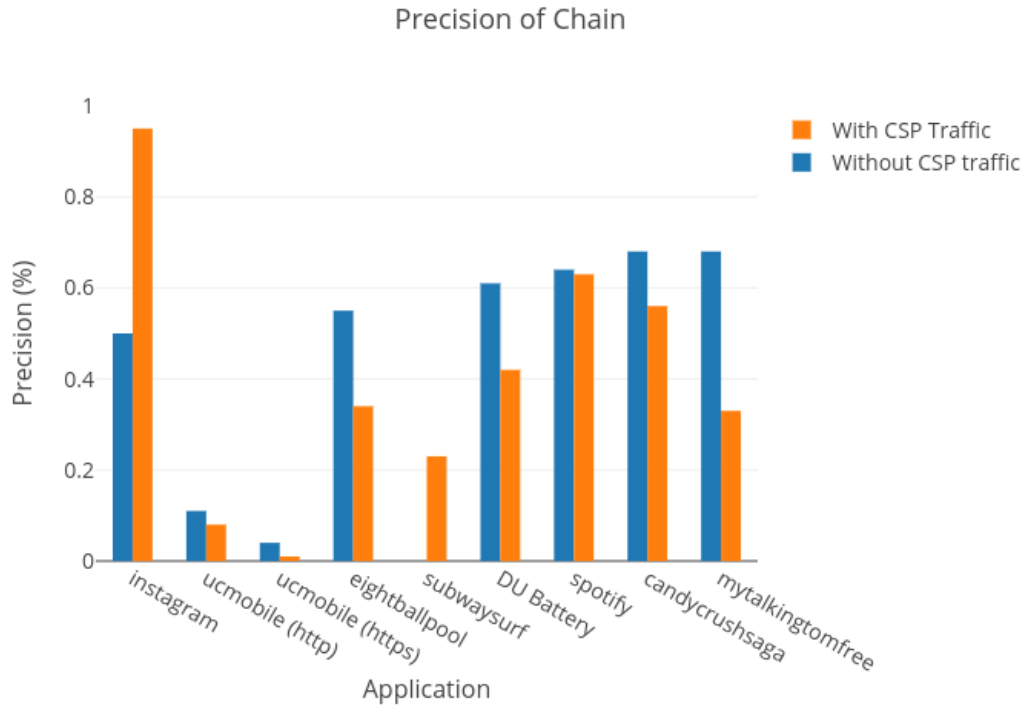

FIGURE 6.2: Precision of Chain with and without CSP traffics

It is expected that the precision of Chain gets increased due to its sequential design. Indeed figure 6.2 shows that j48 precision improvement leads to precision enhancement of Chain in several classes consequently. However, nDPIng does not inherit the same precision improvement from j48 and a fraction of the improvement is dumped by nDPIng module. The outdated signature database of nDPIng limits the second module of Chain's results to ucmobile(http), ucmobile(https) or unkown classes. Therefore, nDPIng increases the false positive rate of ucmobile classes and reduces their precision.

j48 can label just those flows which nDPIng considers them as unknown. nDPIng classifies 42% and 52% of subwaysurf traffic as ucmobile(http) and ucmobile(https) respectively. Following, j48 classifies 6% of the remaining traffic generated by subwaysurf as spotify. Therefore, the precision of subwaysurf is zero in the figure 6.2.

j48 contributes false positive to instagram class by misclassifying some flows from candycrushsaga, spotify, DU Battery, subwaysurf, ucmobile (https) classes. nDPIng as the second module of Chain labels all of them except flows from DU Battery and ucmobile (https) incorrectly as ucmobile (https). Consequently, the precision of instagram classification is increased with the trade of precision reduction of ucmobile (https) which is shown in figure 6.2.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

This thesis proposed the Chain classifier as a practical approach for improving the efficiency of DPI traffic classification techniques by chaining it to a *low-recall*, but *fast* and *high-precision* classification stage (based on *port* and *ML*). I have demonstrated within this thesis that Chain classifier improves the classification efficiency with respect to nDPIng, a state-of-the-art DPI classifier: it precisely classifies network traffic 45% faster than nDPIng, with comparable classification performance. Since DPI is widely used in application firewalls, Chain can be used as a solution to operate online classification quicker and with less computational resources.

DPI suffers from low flexibility to adopt with a new ecosystem. Indeed, extracting signatures from a vast number of new apps which merge to internet daily ecosystem is an intensive human work. Furthermore, updating a DPI database with the new signatures contributes additional overhead to adopt a DPI classifier to a new ecosystem. On the other hand, Chain can be adopted with the same level of complexity as a ML classifier with a new ecosystem. In this case, Chain performs close to the ML classifier which is much higher than a DPI classifier with outdated signature database. However, it is required to update Chain's DPI module to achieve its maximum performance. Therefore, Chain outperforms a DPI classifier when the mixture of traffic is in transition phase by emerging a new concept (*e.g.*, Emerging mobile traffic to the internet ecosystem).

The multi-stage architecture is specifically suited for the recently standardised approach for network services defined in [87] (Service Function Chaining Architecture). Both stages of Chain can be implemented as Virtual Network Functions (VNFs) and can perform the role of Classifiers for the network policies depending on the *app* that generated the traffic [88]. Their implementations as VNFs allows to separately instantiate them, maximally benefiting from the scalability and elasticity of Network Function Virtualisation to dynamically adjust resources and traffic forwarding depending on the traffic mix that traverses the Chain stages. Service Chaining also provides a standard mechanism for the encapsulation of flow metadata that affects traffic forwarding and service chaining: the Network Service Header (NSH) [89]. In the case of Chain, the first stage encapsulates in the NSH either the classification response of the first stage (and exits Chain services), or the verdict of ML and the subsequent classification service (next stage of Chain).

Chain allows for a relatively more privacy-friendly approach compared with full-DPI, by limiting the extraction of features from the payload of the packets only when classification has not been possible with other privacy-preserving means.

Within this thesis I have studied the relationship between mobile app and `CSP` as well as its impact on NTC *precision*. The outcome shows that the traffic generated based on `CSP` and app communication reduces the performance of classifiers which

developed following of ML approach. It proves that although evaluating different aspects of a flow is an efficient approach in many ways to detect the original app, it is not sufficient. In fact, having a deep knowledge about the ecosystem is essential to deal appropriately with the network traffic mixture and interpret correctly the information which is included in a flow.

Although a ground truth is a fundamental ingredient to do any measurement in NTC filed, it is challenging task to obtain a representative ground truth. I have proposed and implemented GTEngin as a reliable and scalable platform to build a ground truth consisting of network traffic from any desired android apps. Multiple devices usually share a connection link in the real life scenario. This divides the available capacity of the link among the devices and consequently affects the statistic of their traffic. The ability of involving and controlling multiple devices, gives the unique ability to GTEngin to build the ground truth with statistics close to real-world traffic which is vital for ML classifiers.

7.2 Future Works

Online operation is an important capability of a NTC classifier which is vital for many real life applications like security and resource management. Chain is designed to perform online and implement on a platform which also supports online operation. However, Chain online classification functionality has not been investigated yet and it is an open filed of research for future.

Furthermore, implementing Chain in a hardware like NetFPGA can be considered as a next step in direction of investigating its online performance characteristic.

The multi-stage architecture of Chain is also specifically suited for implementation in the form of VNF chaining, where stages of different memory/computational cost can be independently scaled and deployed according to dynamic network traffic mix and rate.

Detection of any illegitimate access to a network resource is in the scope of Intrusion Detection (ID) [90]. Many IDs[91][92][93] use DPI to detect malware and malicious activity looking for their signature in the flowing traffic. Although it is beyond the scope of this thesis to study the performance of Chain in different network application, it is interesting to investigate the performance gain which can be achieved by substituting conventional DPI classifiers by Chain in IDs.

CSP absorbs the attention of many network players and motivates apps developer to outsource their infrastructures by relaying on the cloud services more than ever. Consequently, the communication of an app is not limited to its servers any more. In this thesis I have proposed an algorithm to detect CSP communication based on PTR and ASN. However, my solution does not have the ability of distinguishing between the various services which a CSP can provide. The CSPs services can be categorised in two main categories: I) Standard services such as advertisement and CDN, II) General services like cloud computing which can be modified to support the main functionalities of an app. Although the generated traffic from the second communication is valuable for NTC, the first category traffic is considered noise from the point of view of NTC following ML approach. Therefore, it is essential to distinguish between traffic generated from these two communication categories to filter the noise and preserve the valuable data. Identifying CSPs's services is an interesting open area of research which can be considered as a future work.

There are vast number of apps which published world-wide and GTEngin provides a great opportunity to build a ground truth from their traffics. GTEngin is

scalable and can control multiple devices simultaneously. Besides, GTEngin runs different apps on different devices in a given time. Therefore, the captured traffic is a mixture of flows from different devices and apps which reflects the real life scenario. App's developers are interested to integrate a new functionality to their app daily to satisfy their customers. Although GTEngin labels the capture flows based on their process name, NTC needs a ground truth with the higher resolution to develop a classifier with the ability of detecting the various functionalities of a given app. To this end, GTEngin should be updated.

Besides, Internet of Thing (IoT) is supposed to bring sensors, computation and communication in all aspects of human life. Therefore, supporting IoT devices communication is one of the constrain of 5G design[94][95]. It is expected that traffic which generate by IoT devices will shape the future network traffic. Consequently, it is essential to update GTEngin to support IoT devices to build a ground truth required by any NTC study targeting IoT traffic.

Appendix A

CSP Footprint

A.1 list of CSP PTR records

TABLE A.1: List of PTR record assocsiated to CSPs.

CSP Name	PTR **Record**
Akamai	.akamaitechnologies.fr
Akamai	.akamaitechnologies.com
Akamai	.akamaized.net
Akamai	.edgekey.net
Akamai	.akadns.net
Akamai	.srip.net
Akamai	.edgesuite.net
Akamai	.akamai.net
Akamai	.akamaiedge.net
Akamai	.akamaihd.net
Google	1e100.net
Google	.l.doubleclick.net
Google	.youtube.
Google	.googlesyndication.
Google	.googleusercontent.com
Google	.gstatic.com
Google	.google.
Google	.googlehosted.com
Google	googlevideo.com
Edgecast	.transactcdn.net
Edgecast	.v4cdn.net
Edgecast	.v2cdn.net
Edgecast	.v5cdn.net
Edgecast	.v3cdn.net
Edgecast	.v1cdn.net
ChinaCache	.ccgslb.com
ChinaCache	.ccgslb.net
EdgeCast	.wac.
ChinaCache	.c3cache.net
EdgeCast	.edgecastcdn.net
EdgeCast	.systemcdn.net
EdgeCast	.adn.
ChinaCache	.chinacache.net

To be Continue

Table A.1 – *The continue of the PTR record list*

CSP Name	PTR **Record**
EdgeCast	.wpc.
ChinaCache	.c3cdn.net
ChinaNetCenter	.wscdns.com
CDNetworks	.panthercdn.com
Yahoo	.yahooapis.com
Yahoo	.ay1.b.yahoo.com
Facebook	.facebook.net
ChinaNetCenter	.lxdns.com
Yahoo	.yimg.
Facebook	.cdninstagram.com
Yahoo	yimg.com
Facebook	.facebook.com
Facebook	.fbcdn.net
ChinaNetCenter	.wscloudcdn.com
CDNetworks	.cdngc.net
ChinaNetCenter	.ourwebpic.com
CDNetworks	.gccdn.net
CDNetworks	.gccdn.cn
Azion	.azion.net
MaxCDN	.netdna.com
Azion	.azioncdn.com
Mirror Image	.cap-mii.net
Aryaka	.aads1.net
Mirror Image	.instacontent.net
Fastly	.fastly.net
Mirror Image	.mirror-image.net
MaxCDN	.netdna-cdn.com
Azion	.azioncdn.net
Medianova	.mncdn.net
Aryaka	.aads-cn.net
Aryaka	.aads-cng.net
Fastly	.fastlylb.net
MaxCDN	.netdna-ssl.com
Taobao	.tbcdn.cn
Taobao	.taobaocdn.com
Taobao	.gslb.taobao.com
Fastly	.nocookie.net
Medianova	.mncdn.org
Medianova	.mncdn.com
Limelight	.lldns.net
CDN77	.cdn77.org
NYI FTW	.nyiftw.com
Cloudflare	.cloudflare.net
Windows Azure	.vo.msecnd.net
Windows Azure	dc-msedge.net
OnApp	.r.worldcdn.net

To be Continue

Table A.1 – *The continue of the PTR record list*

CSP Name	PTR **Record**
LeaseWeb CDN	.lswcdn.eu
TurboBytes	.clients.turbobytes.net
KINX CDN	.kinxcdn.com
leaseweb	leaseweb.net
leaseweb	leaseweb.com
CDN77	.cdn77.net
TurboBytes	.turbobytes-cdn.com
KINX CDN	.kinxcdn.net
amazonaws	amazonaws.com
Cloudflare	.cloudflare.com
Instartlogic	.insnw.net
LeaseWeb CDN	.lswcdn.net
NYI FTW	.nyiftw.net
OnApp	.r.worldssl.net
amazonaws	amazonaws.com.ch
Limelight	.llnwd.net
Rev Software	.revdn.net
Rev Software	.revcn.net
Instartlogic	.inscname.net
ntt	cloudn-service.com
a2hosting	a2hosting.com
hiberniacdn	hiberniacdn
bluehost	cdnlive.net
SwiftCDN	.swiftcdn1.com
fasthosts	fasthosts.net.uk
cloudvps	cloudvps.com
Rackspace	.raxcdn.com
MediaCloud	.cdncloud.net.au
ReSRC.it	.resrc.it
Internap	.internapcdn.net
sgded	sgded.com
x2n	x2n.com.br
cdnetworks	cdnetworks.com
PageRain	.pagerain.net
dsl	dslnet.pk
theplanet	theplanet.com
jsDelivr	.cdn.jsdelivr.net
Incapsula	.incapdns.net
section.io	.squixa.net
ovh	nexxea.com
Blue Hat Network	.bluehatnetwork.com
rockynet	rockynet.com
hostnet	hostnet.nl
vrvm	vrvm.com
cdngp	cdngp.net
xlshosting	xlshosting.net

To be Continue

Table A.1 – *The continue of the PTR record list*

CSP Name	PTR **Record**
BitGravity	. doubleclick.net
inmotionhosting	inmotionhosting.com
clara	clara.net
GoCache	.cdn.gocache.net
dreamhost	dreamhost.com
Akamai China CDN	.tl88.net
afxcdn.net	.afxcdn.net
idealhosting	idealhosting.net.tr
BunnyCDN	.b-cdn.net
KeyCDN	.kxcdn.com
Cotendo CDN	.cotcdn.net
Open Telekom	open-telekom-cloud.com
hosting	hosting.com
aruba	aruba.it
hostforweb	hostforweb.com
Reflected Networks	.rncdn1.com
virtua	virtua.com.br
SwiftServe	.swiftserve.com
smartfocus	emv2.com
Tata communications	.cdn.bitgravity.com
UnicornCDN	.unicorncdn.net
basefarm	basefarm.net
hostgator	hostgator.com
CDNify	.cdnify.io
Zenedge	.zenedge.net
Highwinds	.hwcdn.net
quadhost	quadhost.net
Optimal CDN	.optimalcdn.com
Twitter	.twimg.com
BelugaCDN	.belugacdn.com
Telefonica	.cdn.telefonica.com
CDNsun	.cdnsun.net
Simple CDN	.simplecdn.net
SFR	.cdn.sfr.net
hostingxs	hostingxs.nl
NGENIX	.ngenix.net
hostpoint	hostpoint.ch
BO.LT	.bo.lt
Netlify	.netlify.com
aclst	aclst.com
hosteurope	hosteurope.de
QUANTIL/ChinaNetCenter	.speedcdns.com
Ananke	.anankecdn.com.br
HiberniaCDN	.hiberniacdn.com
Level3	.footprint.net
Level 3	.fpbns.net

To be Continue

Table A.1 – *The continue of the PTR record list*

CSP Name	PTR **Record**
Bison Grid	.bisongrid.net
Yottaa	.yottaa.net
nyinternet	nyinternet.net
Telenor	.cdntel.net
Caspowa	.caspowa.com
psychz	psychz.net
AT&T	.att-dsa.net
VoxCDN	.voxcdn.net
Microsoft Azure	.azureedge.net
WordPress	.wp.com
Advanced Hosters CDN	.pix-cdn.org
xlhost	xlhost.com
StackPath	.stackpathdns.com
cubeCDN	.cubecdn.net
Reapleaf	.rlcdn.com
cbi	cbici.net
purepeak	purepeak.com
Hosting4CDN	.hosting4cdn.com
Cachefly	.cachefly.net
Amazon CloudFront	.cloudfront.net
Alimama	.gslb.tbcache.com
bandwidthx	bandwidthx.net
myracloud	myracloud.com

A.2 list of CSP Autonomous System

TABLE A.2: List of Autonomous System (AS) number associated with CSPs

CSP Name	**AS Name**	**AS Number**
Akamai	akamai-lon	34164
Akamai	akamai-asn1	20940
Akamai	akamai-as	35994
Akamai	akamai-as	16625
Akamai	AKAMAI-ASN2	21342
Akamai	AKAMAI-TYO-AP Akamai Technologies Tokyo ASN	24319
Yahoo	yahoo-swb	393245
Yahoo	inktomi-lawson	14776
Yahoo	YAHOO-ULS	43428
Yahoo	yahoo-cha	14196
amazonaws	amazon-aes	14618
amazonaws	amazon-as-ap	38895
amazonaws	amazon-02	16509
basefarm	BASEFARM-SE-ASN Basefarm AB. Stockholm	8523

To be Continue

Table A.2 – *The continue of the AS list*

CSP Name	**AS Name**	**AS Number**
inmotionhosting	imh-west	22611
Highwinds	highwinds3	20446
Highwinds	HIGHWINDS5	29798
inmotionhosting	inmoti-1	54641
basefarm	basefarm-asn	25148
Google	google	15169
dc-msedge	microsoft-corp-msn-as-block	8068
Cachefly	cachenetworks	30081
hostnet	hostnet	197902
Instartlogic	instart	33047
Incapsula	incapsula	19551
smartfocus	emailvision	39905
Facebook	facebook	32934
hostpoint	hostpoint-as	29097
dreamhost	dreamhost-as	26347
leaseweb	LEASEWEB-NETWORK Amsterdam	16265
cdngp	cdnetworksus-02	36408
xlhost	enet-2	10297
HiberniaCDN	Hibernia-cdn	60922
NGENIX	CCT-AS NGENIX	34879

Bibliography

[1] • *number of daily android app releases worldwide 2018 | statista*, https://www.statista.com/statistics/276703/android-app-releases-worldwide/, (Accessed on 07/24/2019).

[2] *Service name and transport protocol port number registry*, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, (Accessed on 08/04/2019).

[3] *Home [traffic identification engine]*, http://tie.comics.unina.it/doku.php?id=topmenu:home, (Accessed on 07/24/2019).

[4] *Cba - broadband communications research group - traffic classification*, https://cba.upc.edu/monitoring/traffic-classification, (Accessed on 07/24/2019).

[5] V. Paxson, "Bro: A system for detecting network intruders in real-time", *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999. DOI: 10.1016/s1389-1286(99)00112-7. [Online]. Available: https://doi.org/10.1016%2Fs1389-1286%2899%2900112-7.

[6] L. Stewart, G. Armitage, P. Branch, and S. Zander, "An architecture for automated network control of QoS over consumer broadband links", in *TENCON 2005 - 2005 IEEE Region 10 Conference*, IEEE, 2005. DOI: 10.1109/tencon.2005.301139. [Online]. Available: https://doi.org/10.1109%2Ftencon.2005.301139.

[7] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning", *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008. DOI: 10.1109/surv.2008.080406. [Online]. Available: https://doi.org/10.1109%2Fsurv.2008.080406.

[8] F. Baker, B. Foster, and C. Sharp, "Cisco architecture for lawful intercept in IP networks", Tech. Rep., 2004. DOI: 10.17487/rfc3924. [Online]. Available: https://doi.org/10.17487%2Frfc3924.

[9] C. Wright, F. Monrose, and G. M. Masson, "HMM profiles for network traffic classification", in *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security - VizSEC/DMSEC 04*, ACM Press, 2004. DOI: 10.1145/1029208.1029211. [Online]. Available: https://doi.org/10.1145%2F1029208.1029211.

[10] *Port forwarding on your router for team fortress 2*, https://portforward.com/team-fortress-2/, (Accessed on 07/28/2019).

[11] *Firewall evasion with icmp (pingtunnel) | insecurity*, https://stephenperciballi.blogspot.com/2014/09/firewall-evasion-with-icmp-pingtunnel.html, (Accessed on 07/28/2019).

[12] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications", in *Passive and Active Network Measurement*, Springer, 2005, pp. 41–54.

[13] K. Fukuda, "Difficulties of identifying application type in backbone traffic", in *2010 International Conference on Network and Service Management*, IEEE, 2010. DOI: 10.1109/cnsm.2010.5691234. [Online]. Available: https://doi.org/10.1109%2Fcnsm.2010.5691234.

[14] S. Alcock and R. Nelson, "Libprotoident: Traffic classification using lightweight packet inspection", *WAND Network Research Group, Tech. Rep*, 2012.

[15] C. Shen and L. Huang, "On detection accuracy of l7-filter and OpenDPI", in *2012 Third International Conference on Networking and Distributed Computing*, IEEE, 2012. DOI: 10.1109/icndc.2012.36. [Online]. Available: https://doi.org/10.1109%2Ficndc.2012.36.

[16] S. Alcock and R. Nelson, "Measuring the accuracy of open-source payload-based traffic classifiers using popular internet applications", in *38th Annual IEEE Conference on Local Computer Networks - Workshops*, IEEE, 2013. DOI: 10.1109/lcnw.2013.6758538. [Online]. Available: https://doi.org/10.1109%2Flcnw.2013.6758538.

[17] T. N. Thinh, T. T. Hieu, V. Q. Dung, and S. Kittitornkun, "A FPGA-based deep packet inspection engine for network intrusion detection system", in *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, IEEE, 2012. DOI: 10.1109/ecticon.2012.6254301. [Online]. Available: https://doi.org/10.1109%2Fecticon.2012.6254301.

[18] M. Canini, W. Li, M. Zadnik, and A. W. Moore, "Experience with high-speed automated application-identification for network-management", in *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems - ANCS 09*, ACM Press, 2009. DOI: 10.1145/1882486.1882539. [Online]. Available: https://doi.org/10.1145%2F1882486.1882539.

[19] W. Jiang and M. Gokhale, "Real-time classification of multimedia traffic using FPGA", in *2010 International Conference on Field Programmable Logic and Applications*, IEEE, 2010. DOI: 10.1109/fpl.2010.22. [Online]. Available: https://doi.org/10.1109%2Ffpl.2010.22.

[20] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "Netfpga sume: Toward 100 gbps as research commodity", *IEEE micro*, vol. 34, no. 5, pp. 32–41, 2014.

[21] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification", *Computer Networks*, vol. 76, pp. 75–89, 2015. DOI: 10.1016/j.comnet.2014.11.001. [Online]. Available: https://doi.org/10.1016%2Fj.comnet.2014.11.001.

[22] *Man-in-the-middle attack - wikipedia*, https://en.wikipedia.org/wiki/Man-in-the-middle_attack, (Accessed on 07/28/2019).

[23] *Transport layer security - wikipedia*, https://en.wikipedia.org/wiki/Transport_Layer_Security, (Accessed on 07/28/2019).

[24] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox", in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM 15*, ACM Press, 2015. DOI: 10.1145/2785956.2787502. [Online]. Available: https://doi.org/10.1145%2F2785956.2787502.

[25] V. Carela-Español, P. Barlet-Ros, M. Solé-Simó, A. Dainotti, W. de Donato, and A. Pescapé, "K-dimensional trees for continuous traffic classification", in *Traffic Monitoring and Analysis*, Springer Berlin Heidelberg, 2010, pp. 141–154. DOI: 10.1007/978-3-642-12365-8_11. [Online]. Available: https://doi.org/10.1007%2F978-3-642-12365-8_11.

[26] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "Ndpi: Open-source high-speed deep packet inspection", in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, 2014. DOI: 10.1109/iwcmc.2014.6906427. [Online]. Available: https://doi.org/10.1109%2Fiwcmc.2014.6906427.

[27] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape, "Portload: Taking the best of two worlds in traffic classification", in *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, IEEE, 2010. DOI: 10.1109/infcomw.2010.5466645. [Online]. Available: https://doi.org/10.1109%2Finfcomw.2010.5466645.

[28] *Ntop - revision 9350: /trunk/ndping*, https://svn.ntop.org/svn/ntop/trunk/nDPIng/, (Accessed on 07/24/2019).

[29] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification", *Computer Networks*, vol. 76, pp. 75–89, 2015. DOI: 10.1016/j.comnet.2014.11.001. [Online]. Available: https://doi.org/10.1016%2Fj.comnet.2014.11.001.

[30] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "Ndpi: Open-source high-speed deep packet inspection", in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, 2014. DOI: 10.1109/iwcmc.2014.6906427. [Online]. Available: https://doi.org/10.1109%2Fiwcmc.2014.6906427.

[31] T. Bujlow, *Classification and analysis of computer network traffic*. Networking & Security, Department of Electronic Systems, Aalborg University, 2014.

[32] S. L. Salzberg, "C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993", *Machine Learning*, vol. 16, no. 3, pp. 235–240, 1994. DOI: 10.1007/bf00993309. [Online]. Available: https://doi.org/10.1007%2Fbf00993309.

[33] P. Perera, Y.-C. Tian, C. Fidge, and W. Kelly, "A comparison of supervised machine learning algorithms for classification of communications network traffic", in *Neural Information Processing*, Springer International Publishing, 2017, pp. 445–454. DOI: 10.1007/978-3-319-70087-8_47. [Online]. Available: https://doi.org/10.1007%2F978-3-319-70087-8_47.

[34] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms", in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, IEEE, 2016. DOI: 10.1109/compcomm.2016.7925139. [Online]. Available: https://doi.org/10.1109%2Fcompcomm.2016.7925139.

[35] "Bayesian network classifiers", in *Chapman & Hall/CRC Computer Science & Data Analysis*, CRC Press, 2010, pp. 205–230. DOI: 10.1201/b10391-10. [Online]. Available: https://doi.org/10.1201%2Fb10391-10.

[36] J. Li, S. Zhang, Y. Lu, and J. Yan, "Real-time p2p traffic identification", in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, IEEE, 2008. DOI: 10.1109/glocom.2008.ecp.475. [Online]. Available: https://doi.org/10.1109%2Fglocom.2008.ecp.475.

[37] Y. Zhang, H. Wang, and S. Cheng, "A method for real-time peer-to-peer traffic classification based on c4.5", in *2010 IEEE 12th International Conference on Communication Technology*, IEEE, 2010. DOI: 10.1109/icct.2010.5689126. [Online]. Available: https://doi.org/10.1109%2Ficct.2010.5689126.

[38] *Caida: Center for applied internet data analysis*, http://www.caida.org/home/, (Accessed on 08/02/2019).

[39] R. Fontugne and K. Fukuda, "A hough-transform-based anomaly detector with an adaptive time interval", in *Proceedings of the 2011 ACM Symposium on Applied Computing - SAC 11*, ACM Press, 2011. DOI: 10.1145/1982185.1982290. [Online]. Available: https://doi.org/10.1145%2F1982185.1982290.

[40] S. Lee, H. Kim, D. Barman, S. Lee, C. kwon Kim, T. Kwon, and Y. Choi, "Netramark", *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, p. 22, 2011. DOI: 10.1145/1925861.1925865. [Online]. Available: https://doi.org/10.1145%2F1925861.1925865.

[41] W. D. Donato, A. Pescape, and A. Dainotti, "Traffic identification engine: An open platform for traffic classification", *IEEE Network*, vol. 28, no. 2, pp. 56–64, 2014. DOI: 10.1109/mnet.2014.6786614. [Online]. Available: https://doi.org/10.1109%2Fmnet.2014.6786614.

[42] *Portal university of the republic | cover page*, http://www.universidad.edu.uy/, (Accessed on 08/04/2019).

[43] *Brescia university*, https://www.brescia.edu/, (Accessed on 08/04/2019).

[44] *Tcpdump/libpcap public repository*, https://www.tcpdump.org/, (Accessed on 08/06/2019).

[45] *Bpf*, https://www.freebsd.org/cgi/man.cgi?query=bpf, (Accessed on 08/06/2019).

[46] L. I. Kuncheva, *Combining pattern classifiers: Methods and algorithms*. John Wiley & Sons, 2014.

[47] A. Dainotti, A. Pescapé, and C. Sansone, "Early classification of network traffic through multi-classification", in *Traffic Monitoring and Analysis*, Springer Berlin Heidelberg, 2011, pp. 122–135. DOI: 10.1007/978-3-642-20305-3_11. [Online]. Available: https://doi.org/10.1007%2F978-3-642-20305-3_11.

[48] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Multi-classification approaches for classifying mobile app traffic", *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018. DOI: 10.1016/j.jnca.2017.11.007. [Online]. Available: https://doi.org/10.1016%2Fj.jnca.2017.11.007.

[49] P. Foremski, C. Callegari, and M. Pagano, "Waterfall: Rapid identification of IP flows using cascade classification", in *Computer Networks*, Springer International Publishing, 2014, pp. 14–23. DOI: 10.1007/978-3-319-07941-7_2. [Online]. Available: https://doi.org/10.1007%2F978-3-319-07941-7_2.

[50] W. de Donato, A. Pescapè, and A. Dainotti, "Traffic identification engine: An open platform for traffic classification", *IEEE Network*, vol. 28, no. 2, pp. 56–64, 2014. DOI: 10.1109/MNET.2014.6786614. [Online]. Available: https://doi.org/10.1109/MNET.2014.6786614.

[51] *Coralreef software suite*, http://www.caida.org/tools/measurement/coralreef/, (Accessed on 07/24/2019).

[52] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software", *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, p. 10, 2009. DOI: 10.1145/1656274.1656278. [Online]. Available: https://doi.org/10.1145%2F1656274.1656278.

[53] V. Carela-Español, T. Bujlow, and P. Barlet-Ros, "Is our ground-truth for traffic classification reliable?", in *Passive and Active Measurement*, Springer International Publishing, 2014, pp. 98–108. DOI: 10.1007/978-3-319-04918-2_10. [Online]. Available: https://doi.org/10.1007%2F978-3-319-04918-2_10.

[54] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, "Reviewing traffic classification", in *Data Traffic Monitoring and Analysis*, Springer Berlin Heidelberg, 2013, pp. 123–147. DOI: 10.1007/978-3-642-36784-7_6. [Online]. Available: https://doi.org/10.1007%2F978-3-642-36784-7_6.

[55] A. Dainotti, F. Gargiulo, L. I. Kuncheva, A. Pescapè, and C. Sansone, "Identification of traffic flows hiding behind TCP port 80", in *Proceedings of IEEE International Conference on Communications, ICC 2010, Cape Town, South Africa, 23-27 May 2010*, 2010, pp. 1–6. DOI: 10.1109/ICC.2010.5502266. [Online]. Available: https://doi.org/10.1109/ICC.2010.5502266.

[56] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification", *Computer Networks*, vol. 55, no. 5, pp. 1083–1099, 2011. DOI: 10.1016/j.comnet.2010.11.002. [Online]. Available: https://doi.org/10.1016%2Fj.comnet.2010.11.002.

[57] Y. Wang and S.-Z. Yu, "Machine learned real-time traffic classifiers", in *2008 Second International Symposium on Intelligent Information Technology Application*, IEEE, 2008. DOI: 10.1109/iita.2008.536. [Online]. Available: https://doi.org/10.1109%2Fiita.2008.536.

[58] T. Fawcett, "An introduction to ROC analysis", *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. DOI: 10.1016/j.patrec.2005.10.010. [Online]. Available: https://doi.org/10.1016%2Fj.patrec.2005.10.010.

[59] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks", *ArXiv preprint arXiv:1611.00791*, 2016.

[60] S. E. Gómez, B. C. Martínez, A. J. Sánchez-Esguevillas, and L. H. Callejo, "Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal", *Computer Networks*, vol. 127, pp. 68–80, 2017. DOI: 10.1016/j.comnet.2017.07.018. [Online]. Available: https://doi.org/10.1016%2Fj.comnet.2017.07.018.

[61] H. Jung, "Cisco visual networking index: Global mobile data traffic forecast update 2010–2015", Technical report, Cisco Systems Inc, Tech. Rep., 2011.

[62] *Mobile app usage - statistics & facts | statista*, https://www.statista.com/topics/1002/mobile-app-usage/, (Accessed on 08/07/2019).

[63] D. Ciuonzo, W. D. Donato, A. Pescapè, G. Aceto, and A. Montieri, "A system for reliable and scalable ground truth generation and traffic classification of mobile apps encrypted traffic", 2017. DOI: 10.13140/RG.2.2.16118.29765. [Online]. Available: http://rgdoi.net/10.13140/RG.2.2.16118.29765.

[64] *Ui/application exerciser monkey | android developers*, https://developer.android.com/studio/test/monkey, (Accessed on 07/24/2019).

[65] *Dd-wrt*, https://dd-wrt.com/, (Accessed on 07/24/2019).

[66] *Tcpdump/libpcap public repository*, https://www.tcpdump.org/, (Accessed on 07/24/2019).

[67] *The gnu netcat – official homepage*, http://netcat.sourceforge.net/, (Accessed on 07/24/2019).

[68] *Free android market data, history, ranking | 2011 - 2019*, https://www.androidrank.org/, (Accessed on 07/24/2019).

[69] *Rfc 5246 - the transport layer security (tls) protocol version 1.2*, https://tools.ietf.org/html/rfc5246, (Accessed on 08/23/2019).

[70] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying TLS usage in android apps", in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies - CoNEXT 17*, ACM Press, 2017. DOI: 10.1145/3143361.3143400. [Online]. Available: https://doi.org/10.1145/3143361.3143400.

[71] *Rfc 6101 - the secure sockets layer (ssl) protocol version 3.0*, https://tools.ietf.org/html/rfc6101, (Accessed on 08/23/2019).

[72] *Qualys ssl labs - projects / user agent capabilities: Android 6.0*, https://www.ssllabs.com/ssltest/viewClient.html?name=Android&version=6.0&key=129, (Accessed on 08/23/2019).

[73] N. Vallina-Rodriguez Et al., "Tracking the trackers: towards understanding the mobile advertising and tracking ecosystem", in *Proc. DAT Workshop*, 2016.

[74] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem", in *Proceedings 2018 Network and Distributed System Security Symposium*, Internet Society, 2018. DOI: 10.14722/ndss.2018.23353. [Online]. Available: https://doi.org/10.14722%2Fndss.2018.23353.

[75] ICSI, *Lumen privacy monitor*, https://play.google.com/store/apps/details?id=edu.berkeley.icsi.haystack, 2016.

[76] P. Richter, F. Wohlfart, N. Vallina-Rodriguez, M. Allman, R. Bush, A. Feldmann, C. Kreibich, N. Weaver, and V. Paxson, "A multi-perspective analysis of carrier-grade NAT deployment", in *Proceedings of the 2016 ACM on Internet Measurement Conference - IMC 16*, ACM Press, 2016. DOI: 10.1145/2987443.2987474. [Online]. Available: https://doi.org/10.1145%2F2987443.2987474.

[77] A. Lutu, M. Bagnulo, A. Dhamdhere, and K. C. Claffy, "NAT revelio: Detecting NAT444 in the ISP", in *Passive and Active Measurement*, Springer International Publishing, 2016, pp. 149–161. DOI: 10.1007/978-3-319-30505-9_12. [Online]. Available: https://doi.org/10.1007%2F978-3-319-30505-9_12.

[78] *Cdnfinder by cdnplanet*, https://www.cdnplanet.com/tools/cdnfinder/.

[79] *Cross-platform webpagetest agent*, https://github.com/WPO-Foundation/wptagent, 2017.

[80] McAfee, *Trusted source*, http://www.trustedsource.org/.

[81] OpenDNS, *domain tagging*, https://domain.opendns.com.

[82] *Censys.io*, https://www.censys.io/, 2017.

[83] *Cdn mapping*, https://github.com/Hossein-Doroud/cdn-detector/blob/master/cdnDetector.py, 2017.

[84] A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson, "Haystack: A multi-purpose mobile vantage point in user space", *ArXiv preprint arXiv:1510.01419v3*, 2016.

[85] P. Foremski, C. Callegari, and M. Pagano, "DNS-class: Immediate classification of IP flows using DNS", *International Journal of Network Management*, vol. 24, no. 4, pp. 272–288, 2014. DOI: 10.1002/nem.1864. [Online]. Available: https://doi.org/10.1002%2Fnem.1864.

[86] Z. Liu, R. Wang, M. Tao, and X. Cai, "A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion", *Neurocomputing*, vol. 168, pp. 365–381, Nov. 2015. DOI: 10.1016/j.neucom.2015.05.089. [Online]. Available: https://doi.org/10.1016/j.neucom.2015.05.089.

[87] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture", RFC Editor, RFC 7665, 2015. DOI: 10.17487/rfc7665. [Online]. Available: https://doi.org/10.17487%2Frfc7665.

[88] E. Datsika, A. Antonopoulos, N. Zorba, and C. Verikoukis, "Software defined network service chaining for OTT service providers in 5g networks", *IEEE Communications Magazine*, vol. 55, no. 11, pp. 124–131, 2017. DOI: 10.1109/mcom.2017.1700108. [Online]. Available: https://doi.org/10.1109%2Fmcom.2017.1700108.

[89] P. Quinn, U. Elzur, and C. Pignataro, "Network service header (nsh)", RFC Editor, RFC 8300, 2018. DOI: 10.17487/rfc8300. [Online]. Available: https://doi.org/10.17487%2Frfc8300.

[90] *Intrusion detection system - wikipedia*, https://en.wikipedia.org/wiki/Intrusion_detection_system, (Accessed on 08/09/2019).

[91] A. S. Desai and D. P. Gaikwad, "Real time hybrid intrusion detection system using signature matching algorithm and fuzzy-GA", in *2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT)*, IEEE, 2016. DOI: 10.1109/icaecct.2016.7942601. [Online]. Available: https://doi.org/10.1109%2Ficaecct.2016.7942601.

[92] *Snort - network intrusion detection & prevention system*, https://www.snort.org/, (Accessed on 08/09/2019).

[93] Y. Wang, W. Meng, W. Li, J. Li, W.-X. Liu, and Y. Xiang, "A fog-based privacy-preserving approach for distributed signature-based intrusion detection", *Journal of Parallel and Distributed Computing*, vol. 122, pp. 26–35, 2018. DOI: 10.1016/j.jpdc.2018.07.013. [Online]. Available: https://doi.org/10.1016%2Fj.jpdc.2018.07.013.

[94] A. Ijaz, L. Zhang, M. Grau, A. Mohamed, S. Vural, A. U. Quddus, M. A. Imran, C. H. Foh, and R. Tafazolli, "Enabling massive IoT in 5g and beyond systems: PHY radio frame design considerations", *IEEE Access*, vol. 4, pp. 3322–3339, 2016. DOI: 10.1109/access.2016.2584178. [Online]. Available: https://doi.org/10.1109%2Faccess.2016.2584178.

[95] D. Mishra and S. De, "Energy harvesting and sustainable m2m communication in 5g mobile technologies", in *Internet of Things (IoT) in 5G Mobile Technologies*, Springer International Publishing, 2016, pp. 99–125. DOI: 10.1007/978-3-319-30913-2_6. [Online]. Available: https://doi.org/10.1007%2F978-3-319-30913-2_6.