UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Traffic Classification Based on Statistical Tests for Matching Empirical Distributions of Lengths of IP packets

Miguel Ângelo Silva Neto

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Pedro Ricardo Morais Inácio

Covilhã, Junho de 2013

Dedicatory

"... to my parents and my brothers, for the tenderness and strength that they gave me during the elaboration of this document. Specially dedicated to you, Anita, for the love, patience and tolerance."

Acknowledgements

First of all, I would like to thank my supervisor, Professor Doutor Pedro Inácio, for the opportunity to integrate his dynamic investigation team and for guiding me during my academic path, while sharing his knowledge and sapience with me.

My second thought goes to João Gomes, for all the great discussions, for the shared knowledge and all the patience to deal with several research problems along the master's degree programme. No less important, I would like to mention my colleagues on the lab, specially Rui Brás, Pedro Correia, and Fábio Teixeira, for their support and friendship.

A no less important but different acknowledge goes to my parents and my brothers, mostly for having faith in me through all these years, and not only regarding my academic or professional course, but also in every single personal project in which I was involved in.

Last, but not least, I would like to thank my girlfriend for her motivation, support and understanding during this last year in which, unfortunately, I could not be as present as I would like to. For many months, most of my free time was dedicated to this work, abdicating on some opportunities of spending time with her. For her, my true gratitude and love.

Thank you all.

Resumo

Hoje em dia, a classificação de tráfego de rede assume um papel importante na gestão de redes de computadores. As ferramentas e técnicas que permitem a segregação do tráfego em classes são cruciais para que os administradores de redes possam assegurar a qualidade de serviço, o normal funcionamento e ainda a segurança das redes que administram. No entanto, a constante evolução da infrastrutura e principalmente dos nós terminais, bem como o consequente aumento da complexidade das redes tem dificultado esta tarefa, quer em termos de eficácia como de recursos computacionais. Entre os fatores que mais prejudicam a classificação de tráfego estão a adopção de técnicas de cifragem e de evasão, aplicadas por aplicações de rede. Por estes motivos, muitos investigadores têm concentrado esforços na descoberta de novos meios para classificar tráfego de rede ou na melhoria dos existentes.

Nesta dissertação é relatado um trabalho de investigação no tema da classificação de tráfego de rede, focado na segregação dos fluxos de acordo com a aplicação que os gera, independentemente do facto dessas aplicações usarem diferentes paradigmas de comunicação. Para isso, foi montado um cenário de rede semelhante a um cenário real, onde foi capturado tráfego gerado por várias aplicações. Este tráfego foi objeto de uma análise inicial que permitiu identificar padrões no seu comportamento sem recorrer á informação do conteúdo dos pacotes, usando apenas a distribuição empírica dos tamanhos dos pacotes. Realizada a análise, foram idealizadas e guardadas assinaturas compostas pela distribuição empírica do tamanho dos pacotes e pelo nome da aplicação que lhes deu origem, para todas as aplicações e tipos de tráfego analisado. Estudou-se, de seguida, a melhor forma de obter correspondência entre as assinaturas e o tráfego em análise em tempo-real e pacote-a-pacote, tendo sido adaptados dois testes estatísticos conhecidos por Chi-Squared e Kolmogorov-Smirnov, posteriormente implementados em protótipos de classificadores de tráfego. Para permitir a análise pacote-a-pacote, as duas estatísticas dos testes mencionados antes são calculadas para uma janela de valores deslizante, que itera sempre que chega um novo pacote do fluxo. O número de operações envolvidas na atualização das estatísticas é constante e baixo, permitindo obter uma classificação a qualquer altura de um fluxo.

Cada um dos métodos de classifcação foi implementado num protótipo diferente, tendo ainda sido desenvolvido um classificador que combina as estatísticas de ambos, através de uma heurística. Os classificadores foram testados e avaliados separadamente recorrendo a novos registos de tráfego, gerados pelas diferentes aplicações consideradas, capturados num ponto de agregação. Embora os resultados obtidos para cada um dos classificadores sejam bastante bons, apresentando uma eficácia de classificação acima dos 70%, a combinação dos dois produz uma melhoria significativa dos resultados, classificando corretamente mais de 90% dos fluxos analisados. Os protótipos desenvolvidos foram ainda comparados com outras ferramentas

descritas na literatura da especialidade e disponíveis *online*, tendo-se verificado que, em muitos casos, os classificadores propostos produzem melhores resultados para o tráfego analisado.

Palavras-chave

Análise Estatística, Classificação de Tráfego de Rede Baseado no seu Comportamento, Classificação de Tráfego no Escuro, Monitorização de Tráfego de Rede, Teste Chi-Quadrado, Teste Kolmogorov-Smirnov.

Resumo alargado

Introdução

Este capítulo sintetiza, mas de forma alargada e em língua portuguesa, o trabalho descrito nesta dissertação. Este capítulo começa pela descrição do problema abordado e quais os objetivos a alcançar. De seguida, são apresentados alguns tópicos ligados às técnicas existentes para classificação de tráfego. A penúltima secção é dedicada à descrição dos métodos usados para a resolução do problema identificado. O capítulo termina com a breve discussão das principais conclusões obtidas e com a apresentação de algumas direções para trabalho futuro.

Enquadramento, Descrição do Problema e Objetivos

As redes de computadores compreendem, nos dias de hoje, uma tecnologia sem a qual o Mundo já não sabe viver. À medida que a sua importância foi aumentando, também os requisitos de qualidade de serviço e segurança se tornaram indispensáveis. O preenchimento destes requisitos está, contudo, dependente da capacidade de segregar, classificar e controlar o tráfego que flui na infrastrutura que lhes dá suporte. A classificação de fluxos de rede é, por isso, um recurso crítico para administradores de redes, organizações e provedores de Internet. Tradicionalmente, a classificação do tráfego era feita recorrendo aos números das portas do protocolo *Transmission Control Protocol* (TCP) e *User Datagram Protocol* (UDP). Contudo, assim que várias aplicações começaram a usar números aleatórios para as portas ou números associados a outros protocolos (por exemplo, a porta 80), este método tornou-se obsoleto. Como alternativa, desenvolveram-se métodos de inspeção profunda de pacotes, que usam o conteúdo dos pacotes *Internet Protocol* (IP) para classificar o tráfego. Estes mecanismos são normalmente os mais precisos e detalhados no que diz respeito à identificação da aplicação que gerou determinado pacote IP ou fluxo. Ainda assim, apresentam duas grandes desvantagens: (i) não podem ser aplicados quando é aplicada cifragem dos conteúdos e (ii), são normalmente mais pesados em termos computacionais, por serem baseados na correspondência de cadeias de caracteres com o conteúdo dos pacotes. As limitações apontadas antes têm vindo a motivar a investigação de técnicas para classificação que não use o conteúdo dos pacotes IP, por vezes designadas por técnicas no escuro. Os classificadores resultantes não costumam ser tão precisos como os anteriores, mas têm normalmente uma complexidade computacional menor e podem ser aplicados a tráfego cifrado. Estas técnicas são normalmente baseadas na descrição e captura do comportamento esperado para várias classes em análise usando, por exemplo, estatísticas para várias características do tráfego. O trabalho descrito nesta dissertação focou-se no estudo das propriedades do tráfego de rede e no desenvolvimento de um classificador de tráfego no escuro. O seu tema enquadra-se na área de investigação de *monitorização e análise de tráfego*. O problema abordado nesta dissertação é o de construir um classificador agnóstico aos conteúdos dos pacotes IP e simultaneamente preciso e computacionalmente

eficiente. Como objetivos secundários incluem-se o estudo e geração de conhecimento acerca do comportamento do tráfego de rede e o teste exaustivo do classificador proposto. De forma a atingir os objetivos apresentados, o trabalho de investigação baseou-se na seguinte abordagem:

1. Contextualização com o problema e revisão da literatura da especialidade;

2. Delineação das experiências e configuração do laboratório para obter registos de tráfego e ainda realizar uma análise preliminar a esse tráfego de modo a obter um conjunto de assinaturas que representem as várias classes de tráfego.

3. Estudar e propor formas de fazer a correspondência entre as assinaturas definidas e tráfego capturado da placa de rede.

4. Desenvolvimento de uma arquitetura para o classificador e implementação de protótipos para provar o conceito aqui descrito.

5. Testar os protótipos desenvolvidos e comparar os resultados com outras aplicações existentes na literatura.

Principais Contribuições

A principal contribuição desta dissertação consiste na proposta de um novo método para classificar tráfego de rede baseado no comportamento das diferentes aplicações consideradas, usando apenas o tamanho dos pacotes IP e informações relativas ao fluxo. Outras contribuições importantes incluem: (i) o classificador proposto recorre a estatísticas dos testes estatísticos de Kolmogorov-Smirnov e Chi-Quadrado para fazer a correspondência entre distribuições empíricas, que representam classes e o tráfego em análise; (ii) os protótipos desenvolvidos que implementam as estatísticas em cima mencionadas são capazes de produzir resultados pacote-a-pacote com recurso a uma janela deslizante com tamanho fixo; (iii) os testes ao classificador recorrem a um conjunto de tráfego gerado e capturado em ambiente de laboratório, e os resultados obtidos foram comparados com outras ferramentas existentes na literatura.

Parte deste trabalho de investigação é matéria do artigo científico intitulado *Real-Time Traffic Classification Based on Statistical Tests for Matching Signatures with Packet Length Distributions*, publicado nas atas da *19th IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN)*, que decorreu em Bruxelas, Bélgica, entre 10 e 12 de Abril de 2013 [NGFI13].

Estado da Arte

O trabalho descrito nesta dissertação pressupôs um estudo prévio de abordagens para classificação tráfego já existentes e ainda uma avaliação das suas limitações e vantagens, tendo

em conta o seu contexto de utilização. Esse estudo é descrito no capítulo 2. De uma forma resumida, é dito que os métodos baseados apenas nas portas TCP ou UDP são considerados obsoletos e que os métodos baseados na inspeção profunda de pacotes são dos que têm mais desenvolvimento na indústria. A discussão evolui para os métodos baseados em comportamento e que não recorrem ao conteúdo dos pacotes IP, dado serem os que mais se relacionam com este trabalho.

Alguns trabalhos que também usaram a distribuição empírica do tamanho dos pacotes para classificação de tráfego estão discutidos em [JXXLH10, LLL+09, BMM+07, PBL+03, LLHL12, WP10], mas nenhum deles usou as mesmas estatísticas usadas neste trabalho para fazerem a correspondência das assinaturas, nem implementaram o mecanismo da janela deslizante para permitir a classificação pacote-a-pacote. Por exemplo, em [PBL+03], a distância euclidiana é usada para fazer a correspondência, enquanto que em [WP10], as distribuições são calculadas apenas para os primeiros pacotes de cada fluxo.

Análise do Tamanho dos Pacotes do Tráfego de Rede

Um dos passos iniciais do trabalho de investigação descrito nesta dissertação consistiu na análise do comportamento do tráfego gerado por várias aplicações de *software*, recorrendo para isso à construção da distribuição empírica do tamanho dos pacotes IP. As conclusões deste estudo inicial, descritas no capítulo 3, mostram que existem variações no comportamento passíveis de serem capturadas pela abordagem tomada, e que serão depois utilizadas na construção de um classificador de tráfego de rede. O tráfego de rede usado nesta análise foi capturado junto à fonte onde foi gerado e ainda num ponto de agregação de um laboratório de redes. Desta forma, foi possível observar as propriedades nos dois cenários e estabelecer a verdade dos registos de tráfego. De modo a proceder-se à análise da distribuição empírica do tamanho dos pacotes do tráfego, foi construída uma pequena aplicação que, ao receber o tráfego recolhido, cria a sua distribuição. Tendo em conta os resultados obtidos, esta aplicação foi modificada de forma a poderem ser criadas assinaturas para cada classe de tráfego considerada neste estudo. Estas assinaturas, que contêm apenas a distribuição empírica e o respetivo nome da aplicação que a gerou, foram usadas pelo classificador descrito no capítulo 4. Exemplos de duas destas assinaturas podem também ser consultadas no anexo A.1. Visto que a complexidade computacional destes mecanismos é de extrema importância, foi adotado um mecanismo, denominado de janela deslizante, que permite a criação da distribuição empírica, e a sua alteração, usando um conjunto fixo e relativamente pequeno de operações. O mesmo mecanismo permite obter uma classificação por cada pacote IP capturado e, portanto, durante todo o tempo de vida de um fluxo. Para cada fluxo é criada uma janela com um tamanho de 500 pacotes, cujo funcionamento é descrito na figura 3.2. Os pacotes que pertençam a um dado fluxo são adicionados na janela até que esta atinja o número máximo de pacotes. Depois disto, à chegada de um novo pacote, este é adicionado à janela e o valor inserido há mais tempo é

removido, mantendo sempre o tamanho da janela. Em concordância com o trabalho discutido em [PBL+03], os resultados deste estudo permitem afirmar que a distribuição empírica dos pacotes pode ser uma característica a ter conta no área da classificação do tráfego de rede.

Classificação de Tráfego de Rede Baseada em Testes Estatísticos

O desafio que se colocava na etapa a que esta secção diz respeito era o de conseguir fazer a correspondência entre o tráfego de rede capturado da placa de rede e as assinaturas antes desenvolvidas de uma forma eficiente. Dado que os objetos matemáticos envolvidos nesta correspondência eram distribuições empíricas, procurou-se uma resposta na teoria das probabilidades e, mais concretamente, na área dos testes estatísticos. Depois de estudar o funcionamento deste tipo de testes, chegou-se à conclusão de que o artefacto mais interessante para este trabalho seriam as estatísticas usadas nos referidos testes. As estatísticas usadas no âmbito desta dissertação foram a estatística D do teste de Kolmogorov-Smirnov e a sobejamente conhecida χ^2, definidas no capítulo 4. Dado que estas estatísticas são normalmente calculadas durante análises indeferidas, foram descritos nesta dissertação os algoritmos que permitem a sua estimação para uma janela deslizante de valores. A discussão inclui referência ao facto dos procedimentos acarretarem apenas um conjunto fixo de operações matemáticas simples e de manipulação de memória, de maneira a argumentar que os classificadores construídos com base nestes procedimentos têm requisitos computacionais baixos. As duas estatísticas foram depois usadas para construir os classificadores, cuja arquitetura se ilustra na figura 4.3. Para efeitos de estudo, cada uma das estatísticas foi primeiro usada para implementar protótipos de classificadores diferentes e só depois combinadas num terceiro. Cada protótipo é capaz de capturar e segregar os pacotes em fluxos através dos campos endereço fonte e destino do cabeçalho IP e dos campos porta fonte e destino do cabeçalho TCP ou UDP. De seguida, o tamanho do pacote é inserido na janela deslizante, correspondente ao seu fluxo, e a respetiva distribuição empírica é atualizada de forma eficiente. Por último, a distribuição empírica é comparada com todas as assinaturas de tráfego, sendo devolvida a classe daquela que apresenta o menor valor da estatística D ou χ^2 (dependendo do classificador). O protótipo que combina as duas estatísticas funciona de forma semelhante, exceto no instante de estabelecer a classe: caso as duas estatísticas favoreçam a mesma assinatura, a classe respetiva é devolvida; caso contrário, a classe da assinatura com menor valor em qualquer das estatísticas é devolvida. Todos os protótipos são capazes de produzir classificações por cada pacote analisado.

Testes e Discussão dos Resultados

Tendo em conta o último ponto dos objetivos desta dissertação na secção que inicia este capítulo, os classificadores propostos no âmbito deste trabalho foram sujeitos a conjunto de testes para avaliar a sua precisão e requisitos computacionais. Os testes efetuados recorreram a registos de tráfego capturados em ambiente laboratorial, para efeitos de reprodutibilidade, embora tentando simular o mais possível cenários de utilização reais. Foi tido o cuidado de

construir um conjunto de registos representativos da utilização das redes de computadores de hoje em dia, com as aplicações consideradas no âmbito deste trabalho e para as quais já haviam sido construídas assinaturas. Após terem sido capturados os registos de tráfego, estes foram processados por cada classificador proposto e os respetivos resultados foram usados para avaliar o número de falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos. Para a respetiva avaliação dos métodos foram as métricas *precisão* e *sensibilidade*, ambas descritas na secção 5.2. Os resultados obtidos mostram que os classificadores são precisos e capazes de classificar corretamente a maioria do tráfego analisado. Na tabela 5.1 é possível verificar que, em alguns casos, os classificadores chegam a apesentar uma *precisão* de 100%. Também é possível ver que o pior caso diz respeito a tráfego Voice over Internet Protocol (VoIP). Na parte final deste estudo, o classificador que apresentou melhores resultados na avaliação anterior foi comparado com um conjunto de classificadores do estado da arte e que são parte integrante da ferramenta *NeTraMark*. Os resultados obtidos nesta comparação são bastante satisfatórios, uma vez que classificador proposto demonstrou ser tão bom como os restantes na maioria dos casos e até melhor para algumas classes de tráfego. Por fim, e de modo a obter uma evidência empírica dos baixos requisitos computacionais do classificador proposto, foram ainda feitos alguns testes de *stress*. Nestes testes, o classificador foi executado para vários registos de tráfego e os tempos de processamento foram medidos, bem como a quantidade máxima de memória utilizada na análise. Os resultados desta experiência demonstram claramente que carga de processamento depende linearmente do número de pacotes analisado e que a quantidade de memória depende linearmente do número de fluxos simultaneamente em análise.

Conclusões e Trabalho Futuro

Este trabalho partiu do pressuposto de que seria possível classificar tráfego de rede, de uma forma eficiente, usando distribuições empíricas dos tamanhos dos pacotes. O classificador proposto, que combina as duas estatísticas selecionadas no âmbito do trabalho, comprova esse pressuposto. Alguns dos resultados obtidos com este classificador revelaram-se surpreendentes, já que estavam ao nível de outros classificadores do estado da arte, e até um pouco melhor nalguns casos. Uma razão apontada para este sucesso foi que o uso de distribuições empíricas, apesar de parecer simples, explora uma grande quantidade de informação da característica analisada (neste caso, o tamanho dos pacotes). Depois de colocar em perspetiva os modestos requisitos de memória do classificador proposto, este é apresentado como uma solução atrativa para o problema proposto, embora tenha ainda arestas a limar, nomeadamente no que diz respeito a alguns dos seus parâmetros. As direções de trabalho futuro, descritas na secção 6.2, apontam no sentido de melhorar os próprios classificadores propostos, nomeadamente através do estudo do melhor tamanho para a janela deslizante, que neste trabalho foi ajustado para 500 com base nalgumas experiências e no discernimento humano. Outras direções de trabalho futuro compreendem a tentativa de implementar uma *interface* gráfica que permita uma melhor

interação com a aplicação, o estudo de métodos para derivação automática ou semi-automática de novas assinaturas para tráfego desconhecido, e o aumento do número assinaturas para novas classes de trafego, nomeadamente jogos *online*.

Abstract

Nowadays, traffic classification constitutes one of the most important resources in the task of managing computer networks. The tools and techniques that enable network traffic to be segregated into classes are critical for administrators to maintain their networks operating at the required Quality of Service (QoS) and security levels. Nonetheless, the steady evolution of the infrastructure and mainly of the terminal devices, as well as the consequent increase of the complexity of the networks, make this task a lot harder to achieve, both in terms of accuracy and computational requirements. Some of the factors that most prejudice traffic classification are the adoption of encryption and evasive techniques, employed by network applications. Several researchers have thus been focusing efforts in finding new means to classify traffic or improve the existing ones.

This dissertation discusses a research work on the network traffic classification subject, focused on the segregation of network flows according to the application that generated them, independently of the fact that such applications use different communication paradigms. For achieving that purpose, a network scenario similar to a real one was setup on a lab environment, and several traffic traces generated using different contemporary applications were collected. This traces were initially subject to human analysis, which enabled the identification of behavior patterns without resorting to information inside the contents of the packets, using only the empirical distribution of the size of the packets. After the initial analysis, a set of signatures composed by the aforementioned empirical distributions and the name of respective applications was build, for each one of the applications and type of traffic under analysis. Subsequently, the best means to obtain the correspondence between the signatures and the network traffic in real-time and in a packet-by-packet manner was investigated, from which resulted the modification of two statistical tests known as Chi-Squared and Kolmogorov-Smirnov, later implemented in prototypes for traffic classification. To enable the packet-by-packet analysis, the two statistics of the aforementioned tests are calculated for a *sliding window* of values, which iterates each time a new packet of the flow arrives. The number of operations involved in the actualization of the statistics is constant and low, which enables obtaining a classification at any given moment of the duration of a flow.

Each one of the two classification methods was implemented in a different prototype and then combined, using an heuristic, to obtain a third classifier. The classifiers were tested and evaluated separately resorting to new traffic traces, generated by the different applications considered in the study, captured in a network aggregation point. Even though the results obtained for each one of the two classifiers were good, presenting an accuracy above 70%, the combination of the two methods improves those results, correctly classifying more than 90% of the analysed flows. Additionally, the developed prototypes were compared with other similar

tools discussed on the related literature and available *online*, and it was verified that, in many cases, the proposed classifiers produce better results for the analysed traces.

Keywords Chi-Squared Test, Kolmogorov-Smirnov Test, Traffic Classification in the Dark, Traffic Monitoring, Real-Time, Statistical Analysis.

Contents

List of Figures

List of Tables

List of Listings

Acronyms

CPU Central Processing Unit

DPI Deep Packet Inspection

TCD Traffic Classification *in the dark*

FN false negative

SMTP Simple Mail Transfer Protocol

FP false positive

FTP File Transfer Protocol

HTTP Hypertext Transfer Protocol

IP Internet Protocol

MMS Microsoft Media Server

RAM Random Access Memory

MTU Maximum Transmission Unit

OS Operating System

P2P peer-to-peer

QoS Quality of Service

RTSP Real Time Streaming Protocol

SFTP Secure File Transfer Protocol

SIP Session Initiation Protocol

SSH Secure Shell

RTP Real Time Protocol

SVM Support Vector Machine

TCP Transmission Control Protocol

TN true negative

TP true positive

TV Television

UDP User Datagram Protocol

UTP Unshielded Twisted Pair

VoIP Voice over Internet Protocol

ISP Internet Service Provider

IDS Intrusion Detection System

VPN Virtual Private Network

DFA Deterministic Finite Automata

BN Bayesian Network

NN Neural Networks

NB Naive Bayes

NBKE Naive Bayes Kernel Estimation

DT Decision Tree

KNN k-Nearest Neighbor

ML Machine Learning

XML Extensible Markup Language

ISO International Organization for Standardization

VBR Variable Bit Rate

GNU GNU's Not Unix

ACM Association for Computing Machinery

CCS Computing Classification System

NAT Network Address Translation

Chapter 1

Introduction

This chapter defines the focus and the problem addressed by the research work described in this dissertation, as well as the steps taken to solve that problem. The next-to-last section presents the main contributions for the advance of scientific knowledge and the last section describes the contents of each chapter of the dissertation.

1.1 Motivation and Scope

Computing devices and the networks interconnecting them have been simultaneously the recepient of technological advances and driving innovation for almost half a century. Computer networks connect billions of users everyday both at a local and at a global scale. They are critical for businesses and for the world economy, and support a wide panoply of services through an equally impressive number of protocols and software applications. Even though these networks are hierarchically structured and can be divided into logically smaller networks, knowing or dealing with such heterogeneous and dynamic environments comprises a difficult task. As computer networks became more important, Quality of Service (QoS) and security requirements became mandatory. On the other hand, assuring QoS and security on a computer network is often dependent on the segregation, classification and control of the traffic flowing through the underlying infrastructure.

Network traffic classification refers typically to the capability to identify the service or application that generated a traffic flow. While basic forwarding and blocking policies can be applied using simple flow segregation techniques, enforcing more elaborated QoS or security policies in modern networks requires more granular classification mechanisms. This is specially true if the network is connected to the Internet or if the users of terminal devices have permissions to install software. The problem is that the amount and diversity of traffic may drastically increase in such cases and, more importantly, several applications implementing evasive techniques may be downloaded and installed by the users, potentially affecting the availability and security of the overall network. This area has also been partially dominated by a cat-and-mouse game in the last few years, since developers have been trying to create resilient applications that run behind firewalls and Network Address Translation (NAT) systems, while preventing eavesdropping of the contents of the data units by monitoring systems.

In the Transmission Control Protocol (TCP)/Internet Protocol (IP) protocol suite, evasion is

often achieved resorting to two basic techniques: (i) usage of a transport layer port number that is normally assigned to other well-known protocol, preferably a protocol accepted by the policies implemented in the network; and (ii), encryption of the network or transport layer payload. The success of these techniques is mostly due to the two following reasons: (i) simple segregation mechanisms decide the class of an IP packet or flow using the numbers of the source and destination transport layer ports, which can be changed programatically; (ii) some classification mechanisms decide the class based on the contents of the payload, which is not available if encryption is applied. Actually, while the former segregation technique is now considered obsolete for accurate traffic classification, the latter refers to Deep Packet Inspection (DPI) techniques that were more recently developed. Both approaches are discussed with more detail in chapter 2. The association of the HTTP to traffic flowing through port 80 is an example of the traditional use of port numbers to identify the application protocols.

The advent of DPI techniques is a clear sign that network administrators and ISPs needed a more granular and accurate classification. Nonetheless, a DPI classifier elaborates on computationally expensive bit pattern matching procedures, which do not scale well and cannot handle encrypted traffic. The effectiveness of these techniques depends also of the signatures database, which has to be updated regularly. Additionally, DPI monitoring systems raise privacy concerns [SOG07]. These limitations have been motivating the research and development of techniques that do not rely on the IP packets contents, which are sometimes referred to as Traffic Classification *in the dark* (TCD) techniques [KPF05]. In most cases, *in the dark* mechanisms are based on statistical measures or heuristics applied to different traffic features. Their objective is to capture the behavior that applications or protocols imprint on the network traffic and, even though they are typically not as accurate as DPI techniques, they can be applied when the packets payload is encrypted.

The work described herein is focused on the study of the statistical properties of network traffic and on the research and development of an *in the dark* classifier. Its scope falls within the specific research area of *network monitoring and analysis* of the *computer science* discipline. In the Association for Computing Machinery (ACM) Computing Classification System (CCS), this work would fall in the *C.2.3 [COMPUTER-COMMUNICATION NETWORKS]: Network Operations--Network monitoring* category and descriptor.

1.2 Problem Statement and Objectives

The problem addressed in this master's degree programme is the one of correctly and efficiently classify network traffic. As hinted in the previous section and further discussed in chapter 2, this problem is dominated by several challenges. First of all, there is typically a trade-off between the accuracy and the efficiency of the classifier, since the accuracy may depend on the amount of available data and the ability to process it. The idea is that the classification is

more granular if more data is available, possibly requiring more processing. Secondly, aiming for the construction of a general solution to the aforementioned problem requires assuming that the traffic may either be encrypted or not, since that fact changes the amount of available data. Additionally, evasive techniques to avoid classification (and blocking) are constantly being developed. Thirdly, these kind of procedures have to deal with the fast evolution of the networks and with the increasing number of network protocols, services and applications. In other words, a classifier that behaves efficiently today may not perform as good in the future. Due to their nature, addressing all these challenges simultaneously is difficult.

Constructing a general classification means that handles both encrypted and non encrypted payloads is only possible if its operation does not depend on the payload at all, or if different procedures are applied depending on the case. The initial assumption behind this research work is that it is possible to classify traffic based only on the lengths of the IP packets, which is one of the most general features of network traffic. It is thus assumed that different classes of traffic impact those lengths differently -- an assumption that some authors did in the past too. A classifier using this approach (based on behaviour) is also agnostic to changes in the payload of non encrypted traffic, though the categorization may not be as granular as if a DPI technique was used. The problem is also to perform the classification with a minimum amount of computational resources, so as to keep the method compliant with real-time operation.

Given the above definition of the problem under analysis, the objectives of this research work can be further detailed as follows:

1. Study and gather additional knowledge about the statistical behaviour of the network traffic and about the impact that contemporary applications or protocols have on that behaviour.

2. Devise a classifier based on the IP packet lengths, namely on the empirical distribution of such feature, modest in terms of computational resources and capable of producing results in a packet-by-packet manner. The motivation behind the requirement of producing values on a per packet basis is that such mode of operation favours real-time compliance, since the resulting procedure is able to produce results at any given instance of a network flow.

3. Evaluate the performance of the proposed classifier using a concise and contemporaneous dataset of traffic and compare it with similar tools of the state-of-the-art.

1.3 Adopted Approach for Solving the Problem

The achieve the objectives described in the previous section, the research work was divided into the following phases:

1. Contextualization with the problem addressed in this dissertation and with related research work on this area of knowledge. This part included surveying the state-of-the-art on traffic classification, the study of the functioning of modern networks with detail, namely the TCP/IP protocol stack and structure of IP packets, and getting acquainted with the technologies required to solve the problem.

2. Delineation of the experiments and laboratory setup to obtain traffic traces, and conducting a preliminary analysis to those traces. The three main objectives of this preliminary analysis are to (i) assess if the main assumption, on which the solution relies on, still applies, (ii) get familiar with the task of capturing and processing network traffic and (iii), construct a dataset of signatures representing traffic classes.

3. Study of the best means to match the aforementioned signatures with live traffic.

4. Development of an architecture for the classifier and implementation of prototypes for proof-of-concept.

5. Testing the developed prototypes and comparing them with other state-of-the-art classifiers.

The overall organization of the dissertation (refer to section 1.5) reflects the way this research work was divided.

1.4 Main Contributions

The main contribution of this dissertation is the proposal of a new means to classify network traffic based on the behaviour of different applications, using only the IP packet lengths and flow-level information. Other important contributions can be summarized as follows:

1. The proposed classifier uses a combination of statistics based on the well-known Chi-Squared and Kolmogorov-Smirnov tests to perform the matching between empirical distributions, representing classes, with the traffic under analysis;

2. The algorithms to estimate the aforementioned statistics and to construct the empirical distributions were adapted so as to produce results in a packet-by-packet manner for an iterative *sliding window* with a fixed size. The modifications are described in this dissertation. The usage of the *sliding window* makes the classifier capable of producing classifications during the entire duration of a network traffic flow, and not only during its initial part or for some instants of time. This mode of operation also increases the overall accuracy, since some flows are easier to classify after an initial, and potentially unstable, connection establishment phase.

3. The classifier was extensively tested using traffic captured in a laboratory setup and its performance was compared with other state-of-the-art classifiers. The signatures and the testing datasets were constructed using traffic from a large number of applications, so as to provide evidence that the proposed classifier is accurate and suitable for scenarios where real-time operation is required.

Part of the research work described in this dissertation is the main matter of a scientific paper entitled *Real-Time Traffic Classification Based on Statistical Tests for Matching Signatures with Packet Length Distributions*, published in the proceedings of the 19th IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN), held in Brussels, Belgium, between the 10th and the 12th of April, 2013 [NGFI13].

1.5 Dissertation Overview

The body of the dissertation is composed by six chapters. The contents of each one of the chapters can be summarized as follows:

- Chapter 1 - **Introduction** - presents the scope and the motivation behind the work described in this dissertation, as well as the addressed problem and objectives. It includes a subsection describing its main contribution for the advance of knowledge and the approach taken to fulfil the objectives. The dissertation structure is also briefly discussed at the end of this chapter.

- Chapter 2 - **State-of-the-Art in Traffic Classification** - contains a discussion on the techniques used for traffic classification along the years, along with their advantages and disadvantages. It also contains a revision of the literature on this area of knowledge, with focus on some related works.

- Chapter 3 - **Traffic Capturing and Preliminary Analysis** - describes the laboratory setup used to capture the set of traces analysed during the initial part of the research. The method to construct the empirical distributions in a packet-by-packet manner, as well as the experimental data, are also detailed in this chapter. Additionally, the most interesting findings of this initial phase are discussed, along with the means used to construct the traffic signatures.

- Chapter 4 - **Matching Based on Statistical Tests and Architecture of the Classifiers** - describes the two statistical tests that were on the basis of the methods used for matching the signatures with traffic under analysis. It elaborates on the modifications required for calculating the two statistics in an efficient manner and for a *sliding window* of values. It also presents the architecture of the implemented prototypes for classification, as well as the heuristic used to combine the results of the two matching procedures into a single

one.

- Chapter 5 - **Testing and Discussion of Results** - starts by presenting the means used to test the developed prototypes and evolves to the discussion of the results of the evaluation. The chapter includes the comparison of the accuracy of the prototypes with other classifiers described in the literature, as well as some comments on their computational requirements.

- Chapter 6 - **Conclusions and Future Work** - is devoted to presenting the main conclusions of this research work, with focus on the results, and pointing out future research directions.

Appendix A.1 - **Signatures** - contains two examples of signatures devised along this research work.

Chapter 2

State-of-the-Art in Traffic Classification

2.1 Introduction

The field of traffic classification has been the recepient of several contributions since the beginning of the Internet. The approaches for solving the problem of segregating the flows according to a given criteria have been changing, sometimes motivated by the specific purpose or granularity of the classification, other times as a reaction to tentatives to evade such classification techniques. Nowadays, the literature on this field isolates three main types of approaches: port-based, DPI and TCD techniques. Table 2.1 provides a simple comparison between the three types of approaches discussed below, placing some emphasis on their main advantages and disadvantages. This chapter is divided into three sections, each of them dedicated to the discussion of one of the aforementioned techniques. The subjects are chronologically organized, discussing the oldest approach first.

2.2 Traffic Classification Based on Port Numbers

Traffic classification based on User Datagram Protocol (UDP) or TCP port numbers is a straight-forward method that assumes that each application protocol uses always the same specific transport layer port. Traditionally, this method is mostly useful in the identification of well-known protocols like, for example, *Telnet* or File Transfer Protocol (FTP), which use port numbers 23 and 21, respectively. However, it is rather easy for applications to deceive this type of classification. Many applications began using random port numbers or ports normally assigned to well-known application protocols like e.g., Hypertext Transfer Protocol (HTTP), that normally use port 80. Due to these reasons, this type of traffic classification is considered obsolete nowadays [KFc04, MP05, KBB+04].

Regardless of all of its drawbacks, a few studies have used this kind of approach, or at least the port numbers, to identify applications and protocols in network traffic in the past. In [KW02], the authors analyzed traffic from different peer-to-peer (P2P) applications as *Gnutella*, and *FastTrack* collected in an aggregation point. For distinguish the flows from different applications, they used the TCP port numbers. Gerber et al. [GHN+03] use port numbers to identify traffic from several P2P applications.They used traffic collected from an Internet Service Provider (ISP) and an university campus network, having analysed the properties of the traffic. The authors of [LBBSS02] analyzed traffic from several P2P systems and use port

Table 2.1: Comparison of the approaches for traffic classification (based on [GIF+10]).

Type of Approach	Features Used	Advantages	Drawbacks
Port Number matching	- Associate Port Numbers to Specific Applications.	- Low Computational Requirements; - Easy to implement.	- Less Precision due to random port numbers.
Deep Packet Inspect	- Use the content of payload	- Hight level of accuracy classification.	- may not work for encrypted traffic; - Requires high computational resources; - Can only be used for known applications;
Classification *in the dark*	- Use only packet header and flow level information;	- Can be used in Encrypted Traffic; - Usually have better performance than DPI; Can identify unknown applications from target classes.	Usually has lower classification performance when compared with DPI.

numbers to identify the different applications under analysis. Several tools use also this kind of approach to provide information about the application protocol, like Wireshark [wir13] and *Corel Reef* presented by Moore et al. [MKK+01].

2.3 Traffic Classification Based on DPI

DPI methods are the most obvious alternative to the port based mechanisms. The system using this kind of mechanisms is capable of inspecting the payload of the IP packets, normally up to the application layer, where it searches for bit patterns of a set of signatures stored in a database. Each signature is associated with a given application protocol, virus or, more generally, a given class. This approach can thus be also applied to the identification of threats, malicious data and other anomalies. Because of their accuracy, classification based on DPI is especially useful for accounting solutions, or for situations or systems in which accuracy is crucial as, for example, Intrusion Detection System (IDS). Figure 2.1 presents a rule specification for the *Snort* IDS to detect *BitTorrent* traffic, derived by Carvalho et al. [Dav]. *Wireshark* also uses this kind of approach to identify several network protocols. For example, figure 2.2 shows the signature used by *wireshark* to identify HTTP traffic.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P BitTorrent Outgoing announce
request"; flow:to_server,established; content:"GET"; offset:0; depth:4;
content:"/announce"; distance:1; content:"info_hash="; offset:4;
content:"event=started"; offset:4; classtype:policy-violation; sid:1000301; rev:1;)$
```

Figure 2.1: Rule for detection of traffic generated through *BitTorrent*, as presented in [Dav].

DPI approaches are typically the most accurate ones, but they present some drawbacks. Inspecting each packet is a demanding task, in terms of computational resources, which can

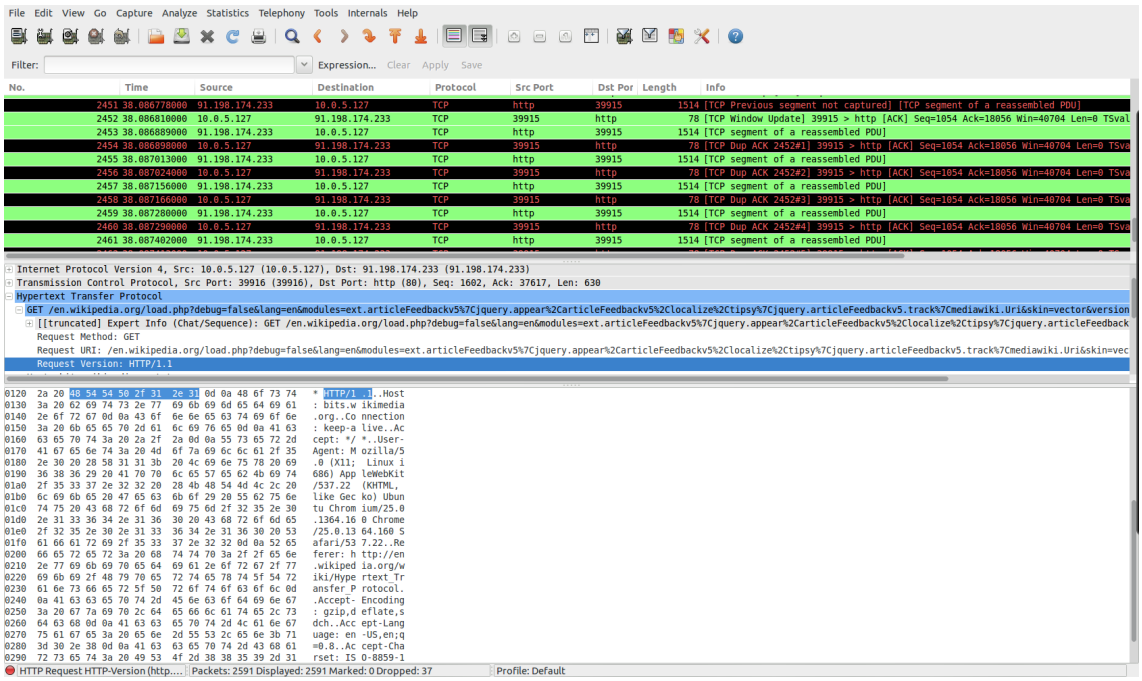

Figure 2.2: Example of a signature used by *Wireshark* to identify an HTTP packet.

lead to the malfunctioning of these techniques in high-speed networks since, in such situations, the system may not be able to handle all traffic, become slow or stop responding. To avoid or, at least, minimize this specific problem, some authors or manufacturers of DPI systems make them search only in some parts of the packets or in some subset of the packets constituting a flow, effectively establishing a compromise between efficiency and accuracy. In such case, some flows may remain unclassified because the packets enabling the classification are never processed by the DPI engine. On the other hand, DPI mechanisms can also raise legal issues related with user privacy protection, since the user data travelling in the application payload may be inspected, even if automatically, by a DPI engine.

One of the major drawbacks of DPI techniques concerns their inability to be used when traffic is encrypted. If the packet payload is mostly inaccessible due to encryption, DPI methods are restricted to a few unencrypted packets of a flow or to the headers of the IP packets, often to the transport layer header. Some protocols use encryption for data exchange, but they leave parts of the session unencrypted, namely the parts for establishing such session, which may leave some space for DPI techniques to correctly classify that traffic. Moreover, some applications use both UDP and TCP connections concurrently, but only the TCP sessions is encrypted, again enabling their classification. This approach is also particularly sensitive to modifications in the protocol or to the evolution of the respective application, since alterations in the protocol or in the application may imply modifications in the signature in the DPI database. Additionally, these mechanisms can only identify protocols or applications for which signatures have been devised and installed in the DPI systems.

Several studies focused their classification procedure on the contents of the packet payload by resorting to byte-level signatures associated with specific types of traffic. Dewes et al. [DWF03] used DPI to identify the traffic from different chat protocols. Karagiannis et al. [KBB+04] made an analysis of data from several P2P systems using DPI to identify the traffic. Moore and Papagiannaki [MP05] described a mechanism which aggregates a set of DPI classifiers and searches for signatures within the entire packet. Sen et al. [SSW04] proposed payload signatures for several P2P applications, as for example, *Gnutella*, *eDonkey* and *BitTorrent*. They tested the solution using traffic collected on an access network to a major backbone and on link with 45 Mbps connecting a Virtual Private Network (VPN) to the Internet. *BitTorrent* was the worst case of their experiment, for which they obtained approximately 9.90% of false negatives. For the remaining protocols, the number of false negatives variates between 0% and 4.97%. Nonetheless, they considered that the flows that use well known port numbers of P2P applications are, in fact, P2P traffic. Based on that assumption, each flow that used one of those ports, and was not classified as P2P traffic, was identified as a false negative case. In [BZTH07] the authors used the payload signatures to identify flows generated by some P2P applications. In this solution, when the packet is identified successfully, several information concerning the flow is combined and added to an hash table. This way, the proposed procedure only inspect the packets that belong the flows that were not yet classified. Traffic generated by chat applications is analyzed in [DWF03]. Through this analysis, the authors of this study identified some signatures contained in the payload of the packets. The precision and the recall obtained in subsequent tests were of 93.13% and 91.7% respectively.

Particularly preoccupied with the computational complexity associated with deeply inspecting every IP packet, a few authors have proposed methods based on DPI that can perform more efficiently [GQ08, SEJK08, CCR10]. In [CRE+10], the authors presented two optimizations of a DPI classifier that reduces the data that needs to be checked by the pattern matching engine. Cascarano et al. [CEG+09] evaluated the computational cost of a DPI mechanism by comparing it with a statistical one. This comparison show that depending on the composition of the traces, the DPI mechanism can be as much computationally heavy as the statistical classifier, which is an interesting fact to keep in mind. Smith et al. [SEJK08] used auxiliary variables and optimizations to implement a mechanism for deflating explosive Deterministic Finite Automata (DFA). Using their solution, the authors were able to optimize the process of signature matching, achieving promising results for FTP, Simple Mail Transfer Protocol (SMTP) and HTTP traffic.

Even though payload encryption usually invalidates the usage of DPI methods, some authors have focused their efforts in finding if it was possible to classify encrypted traffic using DPI methods. The authors of [CPF09b], for example, manually devised several payload signatures for *BitTorrent* encrypted traffic. They explored the fact that some fields of the data units were

fixed, regardless of being encrypted. In their papers, the authors provide a set of rules for *Snort*. The same authors present a similar approach to identify encrypted P2P Television (TV) traffic in [CPF09a, EIPF11], this time resorting to the fact that part of the packets follow in plaintext, namely the ones belonging to the session establishment phase or to a secondary control channel. More recently, some authors combined the inspection of the packet payload with statistical analyses or heuristics [SHR+09, LLM10, ZZZY10] to try to improve the accuracy and reduce the computational requirements. A few authors [BMM+07, DPTP10] have also explored the fact that the bytes in packet payloads are more random in encrypted traffic than in non-encrypted traffic. Based on that, they used statistical measures, like entropy or the chi-squared test, to create methods that segregate encrypted from non-encrypted traffic.

Due to their attractive characteristics, namely their classification accuracy and granularity, several DPI applications and systems have been developed or constructed. *l7-netpdlclassifier* [net13], *L7-filter* [l7f09] and *nDPI* [ope13] are good examples of such applications.

2.4 Traffic Classification in the Dark

As discussed in previous section, the deep inspection of the payload of each IP packet is not always a valid option for the identification of applications and traffic in the networks. Therefore, several methods that do not used information contained in the packet payload have been developed along the years. The main strategy of this kind of methods, usually known by *in the dark* methods, is to classify traffic using heuristics or statistical patterns of generic properties of the traffic (e.g., number of connections, flow-level information like packet length, ports and IPs).

Using this type of approach, classification *in the dark* methods are able to identify a protocol or application without needing to examine the payloads. The drawback is that, as they do not use the payload information, this kind of approach cannot aspire to the same accuracy level of DPI methods. Their results should be understood as a strong suspicion about the traffic protocol. Nevertheless, many recent studies have achieved high success rates in the classification of traffic generated by Internet applications. In addiction, this kind of approach can more easily be applied to unknown applications, since many TCD techniques segregate the traffic flows into classes of applications (e.g., web traffic, Voice over Internet Protocol (VoIP), Streaming, P2P, etc) instead of identifying specific protocols. Additionally, classification *in the dark* mechanisms require, normally, less computational resources than DPI mechanisms.

Recently, many authors proposed methods based on heuristics, using several flow-level or packet-level features [SFKT06, JT08, MP11]. In other studies, the behavior of the traffic is modelled using statistical measures. Some of the methods defined statistical fingerprints for

different classes of Internet traffic and used them to classify network data [CDGS07, DCGS09]. Bartlett et al. [BHP07] identified behavioural signatures for P2P *file-sharing*, like the number of failed connections, the ratio of incoming and outgoing connections, and the use of unprivileged ports. They evaluated the mechanism by classifying *BitTorrent* and *Gnutella* traffic, captured from a commercial ISP and from academic institutions. Freire et al. [FZS08] analyzed several properties of traffic and used the Kolmogorov-Smirnov and the Chi-squared tests to implement a mechanism to distinguish VoIP and legitimate Web traffic. They analyzed sevral proprieties of the network traffic to distinguish VoIP and normal Web Traffic and selected some parameters like Web response size, inter-arrival time between requests, number of requests per page and page retrieval time, etc. The evaluation was made using *Skype* and *Google Talk VoIP* data, previously collected in both ISP and university links on a controlled manner.

The authors of [GIF+08] analyzed the heterogeneity of the lengths of the packets to distinguish different types of traffic. They analyzed several traces from P2P and non-P2P applications and tried to identify a behavioral pattern of the P2P traffic. Their conclusions show that the P2P applications generate packet lengths with an high degree of heterogeneity when compared to the packets lengths of non-P2P applications. They used entropy as a measure for heterogeneity and calculated its value for a *sliding window* containing a fixed number of packets to enable the segregation of traffic into traffic classes. Lin et al. [LLL+09] resorted to the distributions of the lengths of the packets and to port association to identify the application protocol. The authors used traffic collected in a real environment to evaluate the method. In [BMM+09] the authors present a method to identify user-sessions using the so-called cluster algorithm. However, the mechanism proposed by them is limited to offline usage, as they need to process the entire flows before producing any classification. Palmieri and Fiore [PF09] proposed a mechanism to classify Internet traffic that is based on recurrence quantification analysis. The authors studied non-linear properties of specific IP flows in which they could determine the recurrence phenomena and hidden non-stationary transition patterns related to each type of traffic. To test the method, the authors used three distinct traces, captured in a university network, and compared their approach with a DPI mechanism. Karagiannis et al. presented *BLINC* in [KPF05], a mechanism for flow classification that does not use information contained in the packets payloads or transport port numbers to identify the application protocol. *BLINC* analyzes properties of each node, like the relation with the remaining hosts, the role in the connection (server or client), the transport layer information, or the average packet size. The authors used traffic captured in different sites of their university to evaluate the mechanism and compare it with the DPI mechanism. Iliofotou et al. [IcKF+11] explored the interactions between the hosts and used such iterations to classify the traffic. The mechanism was tested using two traces from different ISPs and a DPI-based method as baseline. The results showed that the solution was suitable for traffic classification.

More recently, many authors have focused their efforts on the application of machine learning algorithms to the development of mechanisms for traffic classification. In [BMM+07], the authors implemented a Naïve Bayes classification mechanism that uses the payload length and the inter-arrival times. They also implemented a classifier based on the packet payload that uses *Chi-Square* test to identify *Skype* traffic exploring the randomness due to the encryption of the payload. The authors tested their approach using traffic from an ISP and from a campus and compared its accuracy against a signatures based method. Sun et al. [SYP+10] used a distributed application to traffic samples collected in host users and used a probabilistic neural network to classify Web and P2P traffic. In [GCRHMA+06], the authors resorted to Support Vector Machines (SVMs) to identify P2P traffic, while Jiang et al. [JG10] proposed a method based in the $K - NN$ algorithm to classify Internet traffic. In [EMA06], the authors proposed using an unsupervised machine learning solution for traffic classification. They analyzed the performance of the solution using traces of traffic captured in the university campus and compared the results obtained with a supervised machine learning algorithm. Auld et al. [AMG07] presented a classifier based on a Bayesian trained neural network. This neural network was trained with data derivated the inspection of the packets contents. A set of traffic properties and statistics was used as input for the classification process. The authors test the performance of the mechanism with traffic collected eight months after the data used to train the classification mechanism. The results of the accuracy of their classifier are between 95% and 99%. Alshammari et al. [AZH09] used several machine learning algorithms (*AdaBoost*, SVM, Naïve Bayesian, *RIPPER* and *C4.5*) to identify encrypted traffic, using Secure Shell (SSH) and *Skype* as case studies, and compared the results obtained with all of them. Finamore et al. [FMMR10] calculated the probability distribution of the first bytes of the payload of UDP packets and used a test similar to the *Pearson's Chi-Squared* test to match the patterns.

A few previous articles have also used the empirical distributions of the lengths of the packets to classify traffic. In [JXXLH10], the authors calculated the distributions for P2P TV traffic and resorted to a SVM to match the data with the known patterns. Bonfiglio et al. [BMM+07] used the distributions of the lengths of the packets and of the inter-arrival times in cooperation to classify Internet traffic. Lin et al. [LLL+09] use the distributions of the lengths of the packets to classify traffic and used the Euclidean distance to match the patterns.

The most similar works to the classifiers described herein were proposed in [PBL+03] and [WP10]. Parish et al. [PBL+03] used the distributions to classify the traffic and resorted to the correlation coefficient and to the Euclidean distance to match the distributions. Moreover, they calculated the whole distributions for each flow at every 30 seconds, so as to make the method applicable to real-time operation. In [WP10], the authors followed a similar approach but used the Kolmogorov-Smirnov test to measure the distances between the distributions. They calculated the distributions only for the first packets of each flow. Lu et al. [LLHL12], present

a method to match the distribution based in the Euclidean distance and port association to identify traffic.

2.5 Conclusion

Classification of network traffic constitutes one of the most defying research topics nowadays. As can be concluded from the small survey on this chapter, many researchers have contributed to this area of knowledge using very different approaches along the years, corroborating the idea that this is a very dynamic area. Along the years, the solutions for classifying traffic have become more complex, as a proportional reaction to the evolution of the networks and terminal devices. The techniques evolved from simple TCP or UDP port based classifiers to the computationally demanding DPI engines.

For some years, it was believed that DPI would solve the traffic classification problem, since it is a more accurate and granular approach. Using DPI, a system could identify the type of traffic, the application producing the packets and other specific information that other methods could not. The major drawback of DPI is that its accuracy depends on the packets payload and as soon as encryption at the session or higher layers started to be applied, new approaches were needed. More recently, there is an evident effort to devise means for traffic classification that may not depend on the payloads of the packets, sometimes referred to as *traffic classification in the dark* methods, many of them elaborating on machine learning techniques.

This chapter presents not only the most important works on the area, as identifies the features used by each means to classify network traffic, along with some of their advantages and disadvantages. It is possible to deduce from the discussion in this chapter that many of the related works are limited to a given number of traffic classes (e.g., Skype vs. Web traffic) and are may be susceptible to small variations in the protocols of the applications. The four characteristics that define a classifier are its computational requirements, accuracy, ability to deal with encrypted traffic and ability to handle modifications to protocols or applications. There is often a trade-off between the computational requirements and accuracy. While DPI techniques are the most accurate ones, TCD techniques can often handle encrypted data and small modifications to the protocols or applications.

The following chapters describe a TCD method based on the lengths of the IP packets. Because of that, it can handle encrypted traffic and it is highly agnostic to the protocol or application level details. It is mostly dependent on the overall behaviour of the applications and protocols that is printed in the packet length distributions. It is shown that the method is accurate and computationally modest.

Chapter 3

Traffic Capturing and Preliminary Analysis

3.1 Introduction

This chapter describes with detail the phase that succeeded the revision of the state-of-the-art. As discussed in chapter 1, the work plan for tackling the problem at hands includes a preliminary analysis of network traffic, whose objectives were to gain proficiency in capturing and handling traces, study the behavior of different classes of traffic and, later on, to devise behavioral signatures. Most of the effort was placed on the analysis of the packet lengths, and on understanding the relation of their probability distribution with the different applications or protocols. The choice to focus on this particular feature of the traffic is justified by the two following reasons:

1. the packet lengths are always available to traffic collectors, independently of the traffic being encrypted or not;

2. for a given host and excluding the IP packet payload, this traffic feature is the one that potentially reflects better the functioning of an application.

Focusing on this feature of the traffic aids in building agnostic classifiers, since they are based on a very general property of the packets, thought they may be susceptible to intentional modification of the packets lengths to avoid correct classification. Nonetheless, such modification may impact the performance of the respective network applications and protocols, which should discourage such actions. It should be noted that other particular values of the IP packets are also indirectly used by the proposed classifiers. For example, IP addresses, port numbers and the protocol field of the IP headers are used to separate traffic into flows prior to further processing. Last, but not least, it should be also mentioned that the behavior of other features of the traffic was not excluded from the preliminary analysis *a priori*, though the work converged to the usage of that feature only.

The following two sections describe the data sets used in the preliminary analysis and the experimental test-bed in which they were obtained. An explanation on how the empirical distributions of the traffic are calculated and updated in a packet-by-packet manner is included afterwards, before discussing some findings resulting from this part of the work and concluding the chapter.

Table 3.1: Classes of traffic analyzed within the scope of this work.

Traffic Class	Applications / Protocols
Web browsing	Browsing of general web pages, excluding media streaming
HTTP download	Download of a large file, e.g., an executable or an ISO file
Live Streaming	MMS, RTSP, and *Flash*-based streaming
Streaming On-demand	HTTP, and *Flash*-based streaming
P2P video	*PPStream*, *TVUPlayer*, and *SopCast*
P2P *file-sharing*	*BitTorrent*, *Gnutella*, and *eDonkey*
VoIP	*Skype*, and SIP
File transfer	FTP, SFTP
Remote session	SSH

3.2 Experimental Data

The first task that needed to be performed was to devise a series of experiments for capturing network traffic generated by different Internet applications. Since the analysis required the previous knowledge of the applications generating the data, the network traffic was captured in a controlled environment, described in the next section, where each computer was running a single application only. Thus, it was possible to know the ground-truth of the traces without relying on the accuracy of a third-party classifier. A set of applications representative of nowadays usage of the Internet was identified prior to data collection.

An effort to produce a data-set as heterogeneous as possible was made during this part of the work. During the capturing process, traffic samples for different applications of several traffic classes was collected. Table 3.1 summarizes the classes considered in the analysis and the applications used for each traffic class. For generating streaming traffic, several protocols or technologies were used, namely Real Time Protocol (RTP), Microsoft Media Server (MMS), HTTP, or *Flash*. This was done so as to consider the possible influence of the different properties of each protocol. Apart from the different streaming protocols, traffic with distinct properties was analysed, as for example traffic generated by P2P applications or by traditional client-server applications, such as HTTP, FTP and SFTP, or SSH. Additionally, the contexts where the applications were running were varied, by capturing traces with different durations and in computers with distinct Operating Systems (OSs), namely Linux and Windows. A total number of 92.317 traffic flows were counted, corresponding to 35.317.091 packets and approximately 15.7 GB of traffic.

The list of applications were chosen so as to be representative of the normal usage of the Internet by the time this master's degree programme was performed. Whenever possible, the most recent versions of each application and OS was used. The traffic class that is missing from this analysis is online gaming, which is left as future work.

3.3 Experimental Test-bed

The experimental test-bed was implemented in the network lab of the Department of Computer Science, at the University of Beira Interior. As schematized in figure 3.1, the lab is formed by 24 computers, connected, in groups of four, to six Enterasys switches. The six switches are connected to a main switch that links all the computers to a gateway. The test-bed is connected to the Internet through the internal network of the university campus, 3 hops away of the gateway.

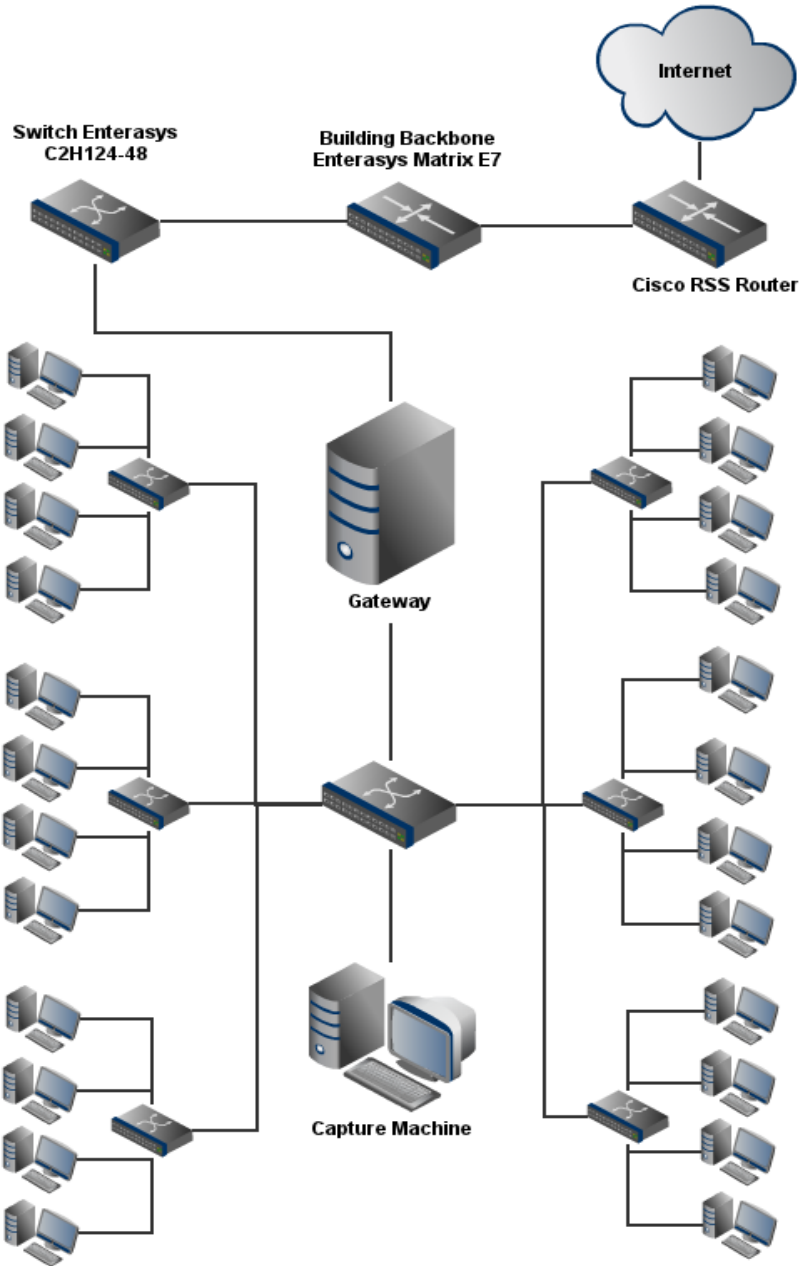

Figure 3.1: Schematics of the network lab of the Department of Computer Science of the University of Beira Interior.

The computers were running different applications from the classes described in table 3.1 and,

when possible, different client applications of the same protocol were used. For example, traces for the *BitTorrent* protocol where generated utilizing the well-known *BitTorrent* client and *Vuze*, another popular client for this protocol. During the experiments, and as previously mentioned, some of the computers were running *Microsoft Windows Server* OS, while others where started in the *Fedora Linux* OS. Table 3.2 summarizes the computational specifications of the hardware of the terminal computers of the lab, namely in terms of Central Processing Unit (CPU) and Random Access Memory (RAM). The table also shows the software that was running on the computers.

Table 3.2: Hardware and software characteristics of the equipment used in the research lab test-bed.

Type and Operating System	CPU	RAM	Software
Desktop Windows Server 2003	Pentium D 2.66Ghz	1GB	GoalBit, Skype, Youtube, TVUPlayer, UStream, etc.
Laptop Windows 7	Core 2 Duo 2.13Ghz	4GB	GoalBit, PPStream, SopCast, HTTP, Ondemand and Live-Streaming
Laptop Ubuntu 10.10	Core Duo 1.8Ghz	1GB	Skype, FTP and Secure File Transfer Protocol (SFTP) and SSH
Desktop Fedora 9	Pentium D 2.66Ghz	1GB	Skype
Gateway Fedora 9	Xeon 2.26Ghz	6GB	TCPDUMP

As depicted in the figure, the aggregated traffic generated by all the computers of the test-bed was captured in the main switch. This approach allowed capturing traces of aggregated traffic and keep the ground-truth information.

3.4 Real-Time Calculation of the Empirical Distributions of the Packet Lengths

As discussed earlier, the main assumption behind this work is that it is possible to classify network traffic using the packet lengths empirical distributions using a constant and small number of operations. In order to do so, an efficient method for calculating such distributions, as the packets are processed, was required. Calculating the distributions only after all the packets of given flow were captured would not suit the requirements, mostly because the classification would only be produced at the end of the flow.

In this part of work, it would suffice to construct the several empirical distributions using normal means, since the focus was still on the study of the behavior and not on the packet-by-packet matching procedure. Nonetheless, the efficiency of the involved procedures was a concern since the beginning, motivating the anticipation of the development of the method described below.

To cope with the objective of producing results in a per packet manner, the concept of *Sliding Window with a fixed size of* N *packets* was used, whose operation is depicted in figure 3.2 and in listing 3.1. Each time a new flow is detected by the implemented traffic analysis tool or by

the developed prototypes (described in section 4.4), a new *sliding window* is instantiated and the structures for storing the state of a new distribution are initialized. The developed tools keep the states and structures linked to the representation of the flows, as further elaborated in chapter 4. The lengths of the packets belonging to a given flow are added to the corresponding *sliding window* until it fills up (`else` clause on listing 3.1). Once that happens, the distribution is calculated. After that, each time packet length is added to the respective *sliding window*, the oldest packet is removed, to keep the size of this structure fixed. The two corresponding empirical probabilities are then updated, one related with the packet length that was removed and the other related with the one inserted. This way, the update procedure takes only 4 simple composite operations per each packet. The updating procedure for when the window is full is represented by the excerpt of code inside the `if` clause on listing 3.1.

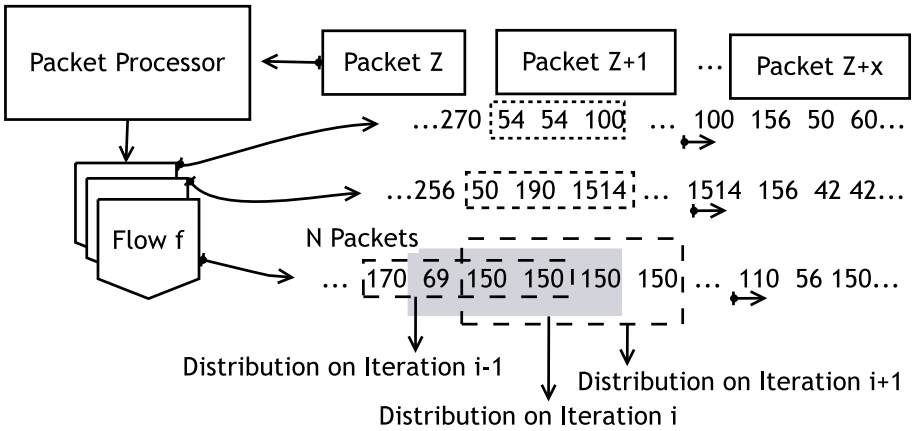

Figure 3.2: *Sliding window* with size of N packets that contain packet lengths for each flow, and the distribution is calculate in each iteration.

It was decided to include listing 3.1 in this section because it clearly shows the computational complexity associated with one of the main procedures behind the proposed classifiers. The operations required for updating the empirical distributions are mostly constituted by multiplications, divisions and sums, and their number is fixed and small (11 lines of code), i.e., the number of required operations does not depend on the number of packets of a flow. The class attribute `number_buckets`, included in the snippet of code, will be discussed with more detail in the next section.

```
1 if(this->window_size <= this->number_packets){
2     int in, out;
3     double & aux2 = window.front();
4     aux=(aux2/this->max);
5     out=(int)(aux*this->number_buckets);
6     aux=length/this->max;
7     in=(int)(aux*this->number_buckets);
8     this->window.pop_front();
```

```
9       this->window.push_back(v);
10      this->distribution[out]-=1;
11      this->distribution[in]+=1;
12      this->number_packets++;
13  }else{
14      int final;
15      aux=(length/this->max);
16      final=(int)(aux*this->number_buckets);
17      this->window.push_back(v);
18      this->distribution[final]++;
19      this->number_packets++;
20  }
```

Listing 3.1: Packet-by-packet calculation of the empirical distributions of the packet lengths.

3.5 Packet Length Empirical Distributions

Packet length empirical distributions for each one of the traffic classes mentioned in section 3.2 were computed and humanly analyzed. It came as no surprise that the analysis gave origin to different packet lengths distributions, depending on the class of traffic that generated the several flows. Previous works, as for example [PBL+03], also corroborate this outcome. Figure 3.3 contains a representation of empirical cumulative distributions for some of the applications tested in this set of experiments. From the observation of the figure, it is possible to conclude that the cumulative empirical distribution of applications using the client-server paradigm exhibit the shape of bi-modal or tri-modal distributions and that, in the case of P2P related applications, the lengths are more variable.

The previous conclusions can, most of times, be explained by the nature and the purpose of the applications. Applications that use a client-server paradigm establish stable connections, for example for downloading files, which originate flows with large packets (containing the chunks of the files) and TCP ACKs for advertising their reception. On the other hand, *file-sharing* P2P applications use small, but with variable length, packets for queries, and may establish several connections for content sharing. Each connection may have a different Maximum Transmission Unit (MTU), resulting in several flows with different properties in terms of packet lengths. Traffic related with VoIP applications generate packets with small and variable lengths, mostly due to the way humans talk and to the encoding technique.

Several traffic traces for the same application installed in different OSs and running in different environments were collected and analyzed. For example, capturing was performed in the test-bed described in 3.3 and, sometimes, in a personal computer with Internet connection.

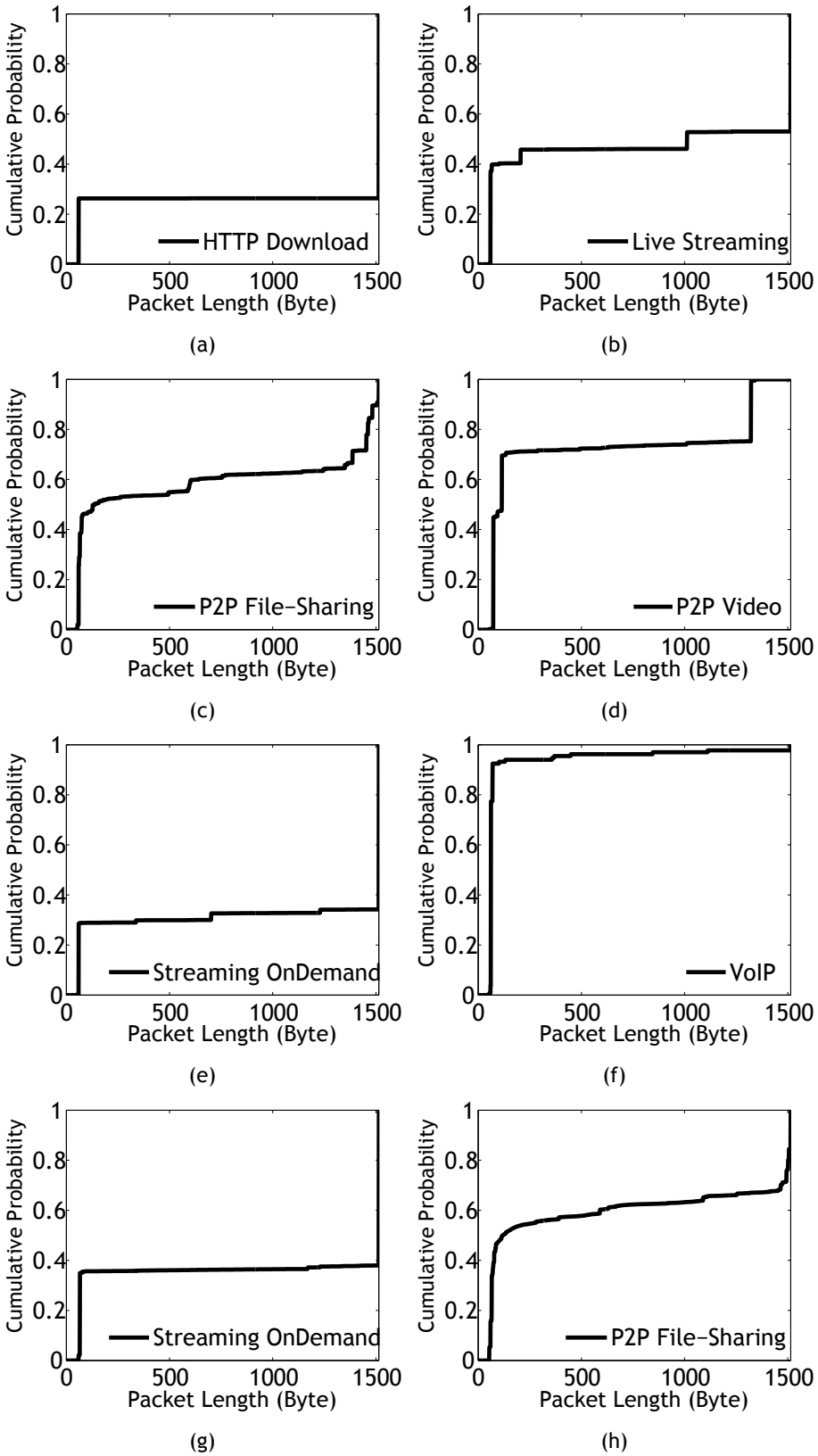

Figure 3.3: Empirical cumulative distributions for traffic of different applications: (a) HTTP download; (b) instance of live streaming; (c) instance #1 of *file-sharing* using a P2P protocol; (d) P2P Video; (e) instance #1 of streaming on-demand; (f) Skype VoIP; (g) instance #2 of streaming on-demand; and (h) instance #2 of *file-sharing* using a P2P protocol.

This experience showed that the distribution of packets depends more on the application than on the environment or OSs, and that the approach is suitable for the heterogeneous network environment. For example, charts (e) and (g) of figure 3.3 concern the representation of two packet length distributions for the same application, but for traffic captured in different hosts with different OSs. The same applies for charts (c) and (h). Their empirical distributions are very similar.

In the first iteration of this part of the work, the empirical (discrete) distributions were constructed for all packet lengths available. In the case of IP over Ethernet, that usually means packet lengths smaller than 1514 bytes. It was noticed, however, that the small variations in the distributions could negatively impact the performance of a classifier. To address that problem, it was decided to create the empirical distributions for buckets with a size of 16 bytes. Using this bucket size, approximately 100 buckets are created, meaning that the *support* of the empirical distributions is rescaled to $0, 1, 2, ..., 99$. The procedure to update the empirical distributions is the same, independently of the bucket size. The only difference is that the packet lengths have to be rescaled prior to updating the probability of the respective bucket. This operation is already shown in listing 3.1.

Figure 3.4 depicts the empirical distributions created with the rescaled *support* for three different applications. The bucket size effectively decreases the granularity of the distributions which, in this case, improves the matching procedure. This is mostly due to the fact that occurrences of similar packet lengths potentially fall within the same bucket, while previously the same case would give rise to different absolute frequencies for the two values. Nonetheless, the bucket size was not subject to analysis and was adjusted following a rule of thumb. The study of its relation with the accuracy of the classifiers is left as future work.

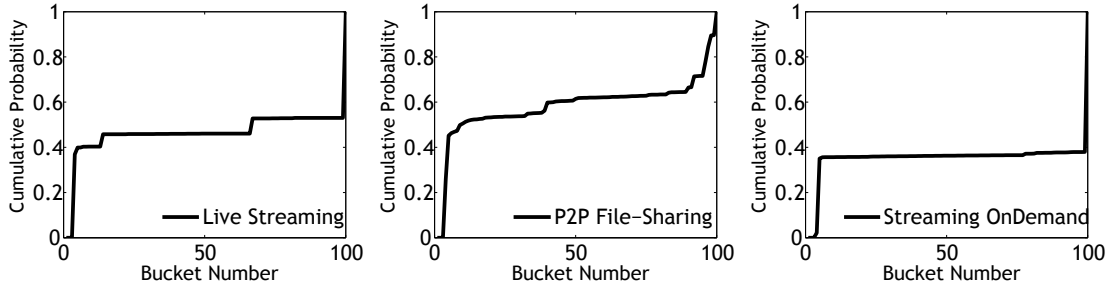

Figure 3.4: Empirical cumulative distributions for three classes of traffic created using bucket with size of 16 bytes.

The remaining part of this work used the empirical distributions as signatures for traffic classification. At the end of the preliminary analysis, a set of signatures was thus built, which were subsequently fed to the developed prototypes. The signatures are files containing only the empirical cumulative distributions and the name of the originating application. For the

sake of consistency, the bucket size was fixed to 16 bytes during the entire process. Notice that, even though the *sliding window* procedure for updating the distributions was described earlier in this chapter, it was not used to produce the signatures or during the preliminary analysis. Additionally, for obtaining an accurate and free of correlations description of a given traffic class, a sampling technique, in which at least 500 packet lengths are randomly selected from a representative flow, was used. Since some applications exhibit different behaviour for different configurations, multiple signatures for those specific cases were built. *Skype* is an example of an application whose traffic behavior changes depending on the configured speech coded, which led to the creation a signature for each available codec. Adjusting the codecs of this application requires handling hidden configuration options on an Extensible Markup Language (XML) file. Examples of signatures can be found on appendix A.1 (listings A.1 and A.2).

3.6 Conclusion

This chapter presented the experimental data and described the experimental test-bed in which most of the traffic traces, used in the scope of this work, were collected. The explanation evolved to the discussion of the findings of a preliminary analysis to some of those traces, after a description of the procedure used to update the empirical distribution of the packet lengths in an efficient and packet-by-packet manner. To keep the computational requirements of the classification methods low, the concept of a *sliding window* of values was adopted, which normally enables reducing the complexity of several statistical estimators to a fixed number of operations. The proposed procedure was described in this chapter because it was developed in this part of the work, thought it was only used in the prototypes presented in the following chapter. It should be noticed, however, that the complexity of this procedure is on the basis of the results for the computational requirements presented in chapter 5. Using a procedure that updates efficiently each time a packet arrives is also an argument in favour of the real-time compliance of the developed prototypes.

The different applications selected to produce the traffic traces exhibited different behaviours in terms of the packet lengths. In this work, this behaviour is captured using the empirical distributions of the referred values. It was argued that the behaviour of the packet lengths can be connected to the nature of the application per se. For example, when a Variable Bit Rate (VBR) codec is used, VoIP applications generate variable length packets because of the way humans talk and because traffic must be dispatched as soon as the voice gets encoded. This way of functioning originates small and variable length packets. If an application intentionally changes the behaviour of the traffic in terms of packet lengths (for example, via the injection of random packets with variable lengths), it may deceive the classification procedures described in the next chapter. Nonetheless, such intent will most probably degrade the performance of the application, both in terms of processing and consumed bandwidth.

At the end of this part of the work, several signatures composed by the name of the application and the values of the respective empirical cumulative distribution were created. The next challenge is on how to match such signatures with traffic captured in real-time.

Chapter 4

Matching Based on Statistical Tests and Architecture of the Classifiers

4.1 Introduction

The previous chapter ended up with the analysis of the behaviour of the packet lengths for different network applications, which was captured by storing the empirical distributions of that traffic feature in files. The next step concerns finding the means to compare live traffic (i.e., traffic currently flowing in the network) with the stored signatures. Since the (mathematical) *objects* that needed to be compared are probability distributions, a search for adequate tools was conducted on the probability theory, particularly on the area concerning statistical tests.

Normally, a statistical test is used to accept or reject a given null hypothesis. They are typically performed during offline analysis of datasets, without any requirements concerning real-time operation. These tests can often be applied to test a sample of a given variable follow a known statistical distribution (the null hypothesis), or if samples of two different variables have the same distribution. Within the context of this work, the functionality mentioned in last is the most useful, since the classification will be based on matching the empirical distribution of live traffic with the previously devised signatures, which are also empirical distributions. Nonetheless, for this work, it is not particularly interesting to reject or accept any hypothesis. It is more interesting to find the signature that best matches the live traffic. That is why the procedures described below uses the statistics of such tests, but not the critical values for acceptance of rejection of the null hypothesis.

This chapter describes with detail the two statistical tests that were considered in this part of the research, as well as how they were implemented to cope with the objectives. Section 4.2 is focused on the Kolmogorov-Smirnov test, while section 4.3 elaborates on the Chi-Squared test. After describing the means to update the required statistics in a packet-by-packet manner, the discussion evolves to the presentation of the architecture of the proposed classifiers, which also elaborates on some implementation details.

4.2 Matching Based on the Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test [CF09], named after its creators, Andrey Kolmogorov and Nikolai Smirnov, was the first statistical test to be taken into consideration in this work, mostly because

of its popularity. This non-parametric test defines a statistic that is typically denoted by D, where $0 \leq D \leq 1$, defined as

$$D = \max_x |F_{1,n}(x) - F_{2,n'}(x)|, \quad (4.1)$$

where $F_{1,n}$ and $F_{2,n}$ are the two empirical cumulative distributions under comparison, and n, n' are the number of independent observations for each one of the variables under analysis. Under the null hypothesis, the distribution of $\sqrt{n \times n'}D$ is known to converge to the Kolmogorov distribution, from which one can determine critical values for the acceptance or rejection of the null hypothesis with a given level of significance. As discussed above, in this work, the most interesting artefact of the statistical test is the value of D for the several signatures and not exactly the goodness of fit. The values obtained for D are used to to choose the signature that best matches the traffic, not to accept or reject one of them. In other words, the signature with the smallest distance is considered to be the one of the class of that flow.

In two of the developed prototypes described below, the calculation of the Kolmogorov-Smirnov statistic is implemented by a snippet of code similar to the one in listing 4.1. This procedure is used when the *sliding window* fills up (or the flow ends) and requires the procedure to iterate through all buckets for each one of the signatures.

```
1 double d_D=fabs(signature->distribution[0]-((double) this->distribution[0]);
2 for(int j=1; j< this->number_buckets; j++){
3     double d_sub;
4     d_sub=fabs(signature->distribution[j]-((double) this->distribution[j]);
5     if( d_D < d_sub )
6        d_D=d_subtraction;
7 }
8 signature->KolmogorovDistance=d_D;
```

Listing 4.1: Procedure for calculating the Kolmogorov-Smirnov statistic in the prototype when the *sliding window* fills up.

The snippet of code used for when the window is full is not included because it was integrated with the procedure for updating the distributions in a packet-by-packet manner. In this case, when a packet arrives to the classifier, the corresponding empirical distribution is updated and checked against the signatures. This operation is optimized, in the sense that the procedure only verifies if the previously calculated distances are affected by this update, thought it has to do that for all signatures in the database. If a given distance is affected, than it is updated and the classification is produced again. Otherwise, the previous class is returned.

Figure 4.1 illustrates the matching procedure for a trace containing video on-demand traffic.

Figure 4.1a depicts the comparison between the empirical distribution generated by the video on-demand traffic and *Skype G729* signature, while Figure 4.1b depicts an analogous comparison for the P2P *file-sharing* signature. The last one, figure 4.1c, shows the comparison between two empirical distributions from the same traffic class. This example clearly that the Kolmogorov-Smirnov statistics favor the signature on the right, since the other distances (Figure 4.1a and 4.1b) are notably larger that the last one.

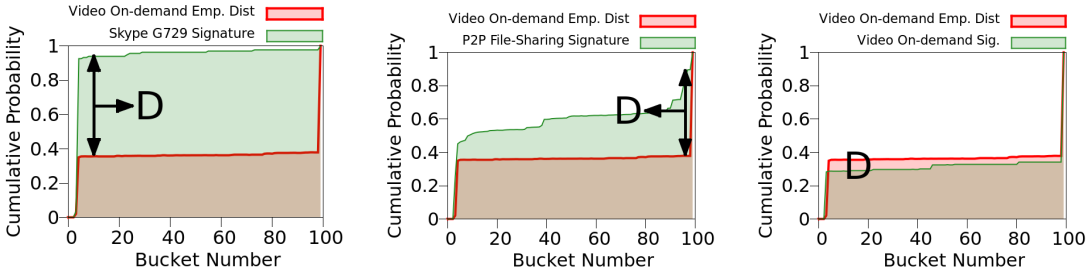

Figure 4.1: Representation of the Kolmogorov-Smirnov statistic for the comparison between the Video On-demand empirical distribution and several signatures: 4.1a maximum distance to the Skype G.729 signature; 4.1b maximum distance to the P2P *file-sharing* signature; 4.1c maximum distance to the Video On-demand signature.

4.3 Matching Based on the Chi-Squared Test

Traditionally, the popular Chi-Square test (known as *Pearson's chi-squared test*) estimates the goodness-of-fit between a sample of a random variable and a given theoretical distribution. In this work, the χ^2 statistic is, once again, used as a measure to compare two empirical distributions (one of them corresponding to the signature and another one corresponding to the live traffic). This statistic is normally defined by equation 4.2, where x_i and E_i $(0 \leq i \leq k)$ denote the observed and the expected frequency of a given event, and $k \in \mathbb{N}$ denotes the total number of different events. In the context of the developed prototypes k is the number of buckets. As in the previously discussed method, the smaller value of χ^2 is, the closer the two distributions are:

$$\chi^2 = \sum_{i=1}^{k} \frac{(x_i - E_i)^2}{E_i}. \quad (4.2)$$

The procedures for calculating the χ^2 statistic in the prototypes developed along this work are depicted in listings 4.2 and 4.3. The procedure in listing 4.2 is applied when the *sliding window* is filled up (or if the flow ends before that), while the one in listing 4.3 applies to the contrary case. Before the window is full, the procedure is slower because the number of events is still increasing, affecting only term E_i. Obtaining updated values for each arriving packet requires a `for` cicle to go through all buckets. Because of that, this procedure is only applied once.

```
int x2=0;
std::vector<double> sig_dist = signature->getDistribution();
for(int i=0; i<this->number_buckets;i++){
    if(e[i]==0)
        continue;
    powv=pow(((double)this->distribution[i]-(sig_dist[i]*(double)this->
        window_size)),2)/(sig_dist[i]*(double)this->window_size);
    x2=x2 + powv;
}
signature->setChiSquare(x2);
```

Listing 4.2: Procedure for calculating the χ^2 statistic in the prototype when the *sliding window* fills up.

Figure 4.2 was included to illustrate the idea behind the Chi-Squared test. The left side of the figure (figure 4.2a) contains the representation of the empirical distributions of the packet lengths for the same class of traffic. On the right side (figure 4.2b), one may find the representation of two empirical distributions for different traffic classes. If formula (4.2) was applied to these samples, the χ^2 value for the samples on the right side would be a large value, indicating a better match for the ones on the left.

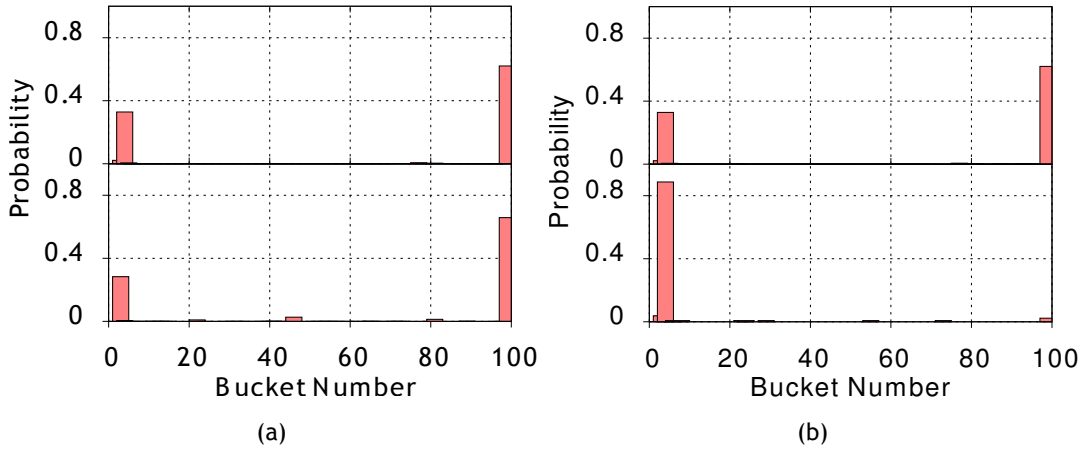

Figure 4.2: Empirical distribution for 4 traffic sample: 4.2a both samples are from video streaming on-demand traffic; 4.2b the top sample is from video streaming on-demand and the bottom sample is from VoIP traffic.

For each flow, the implemented prototype calculates the χ^2 values as soon as the *sliding window* fills up. It calculates this statistic for each signature in the database. After that first costly calculation, it only has to update the statistics resorting to the operations depicted by the equations (4.4), (4.3) and (4.5) where P_{out} and P_{in} denote the addends of (4.2) affected by

the packet lengths that left and entered the *sliding window*, respectively:

$$P_{out} = \frac{(x_o - (E_o \times W))^2}{(E_o \times W)} - \frac{((x_o - 1) - (E_o \times W))^2}{(E_o \times W)}; \quad (4.3)$$

$$P_{in} = \frac{((x_e + 1) - (E_e \times W))^2}{(E_i \times W)} - \frac{(x_e - (E_e \times W))^2}{(E_e \times W)}; \quad (4.4)$$

$$\chi^2_{new} = \chi^2_{ant} - P_{out} + P_{in}. \quad (4.5)$$

In the equations, x_o, x_e and E_o, E_e denote the observed and expected frequencies of the events affected by the change, while $\chi^2_{new} and \chi^2_{ant}$ represent the updated and the previous value of the χ^2 statistic, respectively.

```
1  double op1=0, op2=0, op3=0, op4=0;
2  if(e[out]!=0 ){
3      op1=pow(((double)i_arr_out_1 - (e[out]*(double)this->window_size)),2)/(e[
           out]*(double)this->window_size);
4      op2=pow(((double)i_arr_out_2 - (e[out]*(double)this->window_size )),2)/(e[
           out]*(double)this->window_size);
5   }
6  if(e[in]!=0){
7      op3=pow(((double)i_arr_in_1- (e[in]*(double)this->window_size)),2)/(e[in]*(
           double)this->window_size);
8      op4=pow(((double)i_arr_in_2 -(e[in]*(double)this->window_size)),2)/(e[in]*(
           double)this->window_size);
9  }
10 d_soma = signature->getChiSquare();
11 d_soma -= (op1);
12 d_soma += (op2);
13 d_soma -= (op3);
14 d_soma += (op4);
15 signature->setChiSquare(d_soma);
```

Listing 4.3: Procedure for calculating the χ^2 statistic in the prototype when the *sliding window* is full and a new packet arrives.

The implementation of equations (4.3) to (4.5) in the classifiers results in a procedure with lower and controlled computational complexity. Because of this detail, the processing requirements are only dependent of the number of flows and signatures, but not on the number of packets or buckets. The snippet included in listing 4.3 was included to emphasize precisely that. The

procedure for updating the χ^2 statistic for a given signature and flow requires a fixed number of operations.

4.4 Classifiers Architecture and Implementation

Three prototypes for traffic classifiers were implemented during this master's degree programme. The two matching approaches presented in sections 4.2 and 4.3 were integrated in two independent prototypes. The third prototype combines the two approaches, for comparison reasons. The architecture of all prototypes is similar and can be schematized as in figure 4.3.

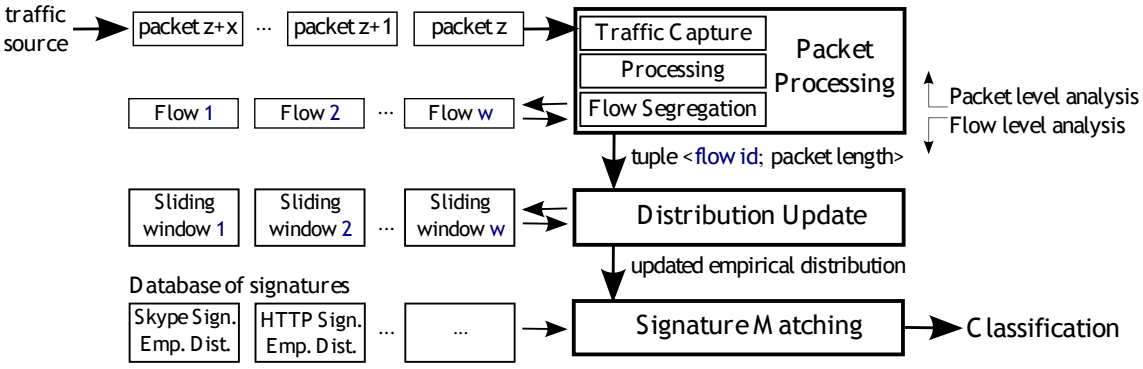

Figure 4.3: Architecture of the proposed classifier.

The architecture of the prototypes is composed by three main modules:

1. The *packet processing* module is responsible for extracting the data, required by the classifiers, from the network traffic under analysis. This module imports the libraries necessary for collecting traffic and parse it into IP packets. It obtains the transport and network layer information needed to identify the flow to which the packet belongs to, as well as the length of the packet. If a packet defines a new flow, the required structure is initialized. This module keeps and manages the states of the active flows so as to enable flow-level segregation of traffic. The concept of flow adopted in this work coincides with the notion of connection. In other words, each flow is identified by the tuple source / destination IP address and transport layer port numbers. Additionally, the flows are divided against the transport layer protocol (UDP or TCP). A time-out of 90 seconds is used to detect the end of a flow for UDP traffic. The output of the *packet processing* module is a tuple containing the identification of the flow (`flow_id`) and the packet length.

2. The *distribution update* module updates the empirical distribution of the flow identified by `flow_id` using the packet length received from the previous module and resorting to the procedure described in section 3.4. In the case of the classifier based on the Kolmogorov-Smirnov test, the D statistic is updated along with the empirical distribution but, conceptually, this is performed in the next module. This module stores and manages

the states (empirical distributions, packet lengths, etc.) of the *sliding windows* of the active flows. If the `flow_id` is unknown to this module, a new *sliding window* is initialized.

3. The *signature matching* module includes the procedures for matching the signatures and producing the final classification. The procedures of this part of the classifiers implement the modified algorithms explained earlier in this chapter.

The prototypes were implemented in the C++ programming language and they are prepared to perform live and offline classification using the `libpcap` library [Lui10]. This library provides a portable framework for low-level network monitoring and allow packet capturing. This language is suitable for this type of application, since it typically originates optimized byte code.

As previously mentioned, a third classifier combining the two approaches was also developed and tested. The combination of the two approaches requires the definition of rules for producing a consistent classification, since both are able to operate independently. The devised rule is simple: if both statistics favour the same signature, the respective class is returned; otherwise, the classifier returns the class corresponding to the statistic with the smaller value. The caveat of this rule is that the two statistics cannot be directly compared because the support of D and χ^2 are different. The former is defined in the interval $[0, 1]$, while the latter is defined for $[0, +\inf]$. It was decided to normalize the values of χ^2 by dividing them by twice the median, which is approximately twice the number of buckets, in this case. This division maps *most* (statistically) of the χ^2 values to the interval $[0, 1]$.

4.5 Conclusion

This chapter presents the inspiration behind the procedures used for matching the signatures, described in the previous chapter, and the traffic under analysis. The architecture of the implemented prototypes for classification was also described in detail herein, along with some comments regarding their implementation.

Due to the nature of the mathematical objects involved, and to obtain meaningful results, the matching procedures were based on two well-known statistical tests: the Kolmogorov-Smirnov and Chi-Squared tests. The two statistics defined in these tests (denoted herein by D and χ^2) were adopted as the means to measure the distance between the empirical distribution of the packet lengths of live traffic and the signatures. Essentially, for each one of the two approaches, the classification methods choose the traffic class of the signature with the smallest value for D or χ^2. In order to comply with the usage of the *sliding window* and with the packet-by-packet operation mode, the estimation procedure for the two statistics was adapted and the modifications were discussed in this chapter. The modifications enable updating the values of these statistics with a fixed and small number of operations, which depends on the

number of flows and signatures, but not on the number of packets or buckets. This conclusion, along with the first conclusion of chapter 3, reinforces the idea that the presented procedures are low on computational requirements, which is important for real-time operation.

The two matching approaches were implemented in two different classification prototypes. Additionally, a third prototype combining the two approaches was built also. Their architecture is divided into 3 different modules: a set of procedures for collecting and processing the traffic; a set of procedures for updating the empirical distributions and the *sliding window*; and the procedures to update the statistics, access the signatures database, and output a classification. The prototypes were implemented in the C++ programming language, which normally originates optimized byte code. At the end of this part of the research work, the prototypes were ready to be tested.

Chapter 5

Tests and Discussion of Results

5.1 Introduction

The final part of this work consisted on testing the matching procedures and the prototypes discussed in chapter 4 along with the signatures discussed in chapter 3. With that purpose in mind, several different experiments were performed, namely: (i) the three matching approaches were first compared to obtain an idea of which of them was the best; (ii) the classifier based on the combination of the two proposed matching procedures was compared with similar tools of the state-of-the-art; and (iii) the computational resources of the implemented prototypes were measured and analysed, so as to obtain an empirical confirmation of their linear dependence of number of flows. The three inner sections of this chapter discuss each one of these experiments with more detail.

5.2 Performance Evaluation and Comparison of the Three Prototypes

The first main experiment was performed to evaluate the accuracy of the three implemented prototypes. They were evaluated resorting to offline traces containing traffic generated by the different Internet applications identified in table 3.1. The use of offline traces enables the experiment to be reproduced and the verification of the results. To decide the correctness of the results of a classifier, some studies compare them against the ones obtained with a third-party classifier that is used as a baseline. In this work, the experimental tested discussed in section 3.3 was used to built different datasets with an established ground truth. The traces composing those datasets were captured in three distinct periods, summing up to a total amount of 24.09 GB of traffic. These traces are different from the ones used to devised the set of signatures.

In the experiment, each one of the prototypes was fed with the traces, obtaining the classifications in return. Their outputs were then compared with the ground-truth information. These results were used to estimate the number of true negatives (TNs), true positives (TPs), false positives (FPs) and false negatives (FNs). Each flow was considered to be a single case, as the corresponding packets are generated by a single application. In other words, a single flow may only contribute to one of those metrics (TNs, TPs, FPs or FNs). Since each prototype

returns a classification on a per packet basis, it may happen that not all of the packets of a given flow are classified with the same class. In order to uniformize the results for performance evaluation purposes, it was considered that the class of the flow was the one of at least 60% of its packets. In other words, if the number of packets with the same classification does not account for at least 60% of total number of packets of their respective flow, such flow is marked as *misclassified*.

The metrics *precision* and *recall*, respectively defined by equations (5.1) and (5.2) [OD08], were used to evaluate the performance of the classifier:

$$Precision = \frac{TP}{TP + FP}; \quad (5.1)$$

$$Recall = \frac{TP}{TP + FN}. \quad (5.2)$$

The *precision* evaluates how many of the total number of cases classified as positive are, in fact, positives, while the *recall* measures the amount of positive cases that are correctly classified. These metrics are commonly used together, as they are generally unable to express the performance of a classifier separately. Their value varies between 0 and 1, arguing in favour of the classifier for values closer to 1.

Table 5.1 and figure 5.1 summarize the results of this experiment, presenting the values for the *precision* and *recall*, for each one of the traffic classes considered in this study and each one of the implemented prototypes. One important remark that should to be made at this point concerns the size of the *sliding window* used during the experiments described in this chapter. The results presented herein were obtained using a *sliding window* with 500 packets, which allows obtaining statistically significant results while keeping the statistics sensitive to isolated changes in the packet lengths. The size of the *sliding window* is a parameter of the classifiers. Its impact on their performance was not subject to a detailed analysis. in this dissertation, and is left as a topic of future work, though good results were obtained with this setup.

The values of the *precision* were larger than 90% for most of the cases, being very close to 100% for streaming on-demand related traffic. The results concerning the *precision* show that the classifiers produce a small number of FPs, when compared with the number of TPs. The values of the *recall* are also generally larger than 80% for all of the implemented matching approaches, though they were better for the matching approach based on the Chi-Squared test and for the classifier combining the two approaches. Actually, when using the former approach, the *recall* is 100% for three traffic classes and larger than 85% in the remaining cases. Nevertheless, the results demonstrate that the classifier combining the two matching approaches is generally better than the other two, both in terms of the *precision* and *recall*.

Table 5.1: Results of the performance evaluation in terms of *precision* and *recall* for the three matching approaches.

	Kolmogorov-Smirnov		Chi-Squared		Two Methods	
Traffic Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Web Browsing	92.35%	86.63%	93.75%	88.24%	98.15%	95.54%
HTTP Download	84.67%	98.97%	87.50%	100.00%	97.53%	100.00%
VoIP	78.35%	85.89%	77.78%	87.50%	89.36%	91.42%
Streaming On-Demand	94.87%	92.19%	100.00%	90.91%	99.74%	94.39%
Live Streaming	91.76%	86.67%	92.86%	85.35%	97.88%	94.14%
P2P Video	96.28%	95.63%	96.15%	96.69%	97.65%	97.51%
P2P *File-Sharing*	92.21%	90.91%	93.21%	91.54%	94.15%	95.74%
FTP	77.87%	76.65%	75.38%	100.00%	90.83%	100.00%
SSH	83.65%	81.84%	80.21%	100.00%	91.19%	99.89%
SFTP	100.00%	89.86%	100.00%	85.71%	100.00%	91.24%

Interestingly, the worst result for this classifier was obtained for VoIP traffic, the only case with a *precision* smaller than 90%. In the datasets used for testing and for building the signatures, VoIP traffic was mostly generated by *Skype*, which is already known for representing a hard challenge in terms of traffic classification. For the majority of the classification techniques in the state-of-the-art, the classification of P2P related traffic comprises also a difficult task. Regardless of that, the implemented prototypes were able to identify traffic generated by this type of application with high *precision* and *recall*.

5.3 Comparison with Other Classifiers

To obtain a more concrete and impartial perspective of the quality of the proposed classifiers, the prototype implementing the combination of the two matching approaches was compared with similar tools of the state-of-the-art. This experiment was performed resorting to a tool known as *NeTraMark* [LKB+11]. *NeTraMark* is an open-source Internet traffic classification benchmarking tool that integrate eleven state-of-the-art traffic classifiers: the payload-based classifier called CRL_PAY [KFc04]; the TCD approach called BLINC [KPF05]; seven approaches based on Machine Learning (ML) techniques; and the CoralReef Software Suite [cor11]. *NeTra-Mark* implements the approaches based on ML techniques by importing the libraries provided by *Weka* [wek12]. *Weka* is tool developed by researchers of the University of Waikato, which features a collection of machine learning algorithms for data mining.

The experiments of this evaluation had to be refined several times before obtaining the results presented in this section. Initially, the datasets used for testing were simply fed to *NeTraMark* and to the proposed classifier. Nonetheless, during a subsequent analysis to the source code of *NeTraMark*, it was verified that the classifiers based on ML techniques were using an unknown part of the traces as training sets. It was also noticed that the tool was automatically selecting small training sets, which could negatively impact the results. To improve these two aspects, several additional traces were purposely compiled into a training set, latter fed to Weka.

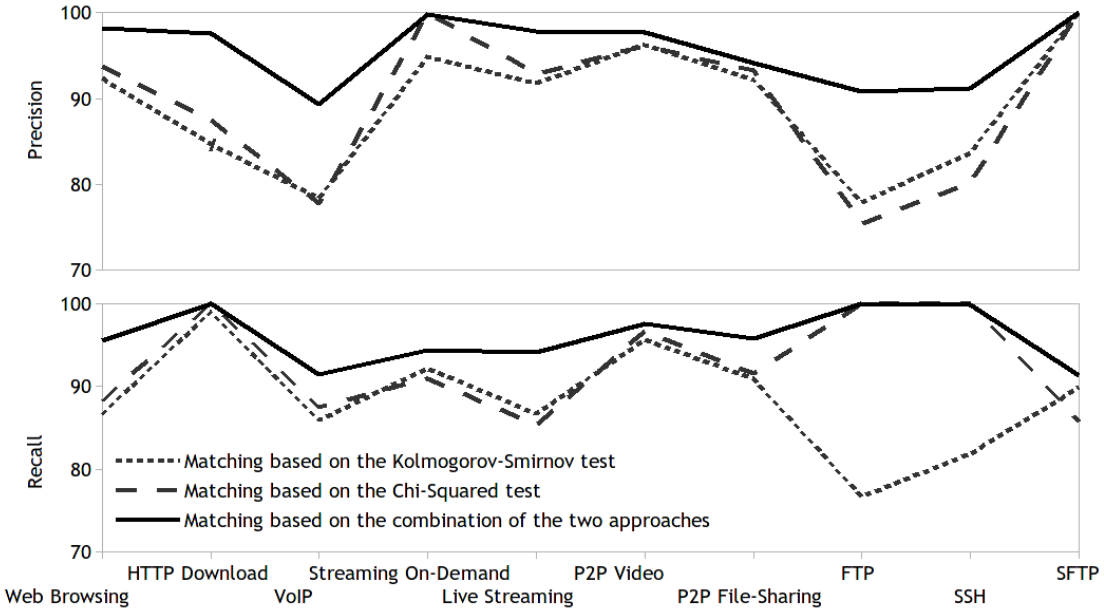

Figure 5.1: Graphical representation of the results concerning the *precision* and *recall* of the three matching approaches for each one of the traffic classes considered in the scope of this work.

The resulting `.arff` file representing the training of the ML techniques was then used in all subsequent experiments. This procedure allowed ensuring the ground-truth information of all classifiers.

After the previously described improvements, several traffic traces, generated by different applications, were processed by the classifiers integrated in the *NeTraMark* tool. The output of each classifier was used to estimate the number of TNs, TPs, FPs and FNs for each them. The metrics used to evaluate the performance of each classifier were the same used in the prototypes evaluation, defined in section 5.2. Nonetheless, the traffic traces used in this experiment were different than the ones used before. This new dataset contains traces that sum up to approximatively 19.0 GB of traffic.

The results obtained during this part of the research work were compiled and presented in table 5.3. In order to be fair and since some of the classifiers used in this evaluation were not able to produce results for some of the considered traffic classes, it was decided to fill up the respective cells of the table with the *Not applicable* expression, instead of including null values for the *precision* or *recall*. For example, the results for the HTTP Download, VoIP, Live Streaming, P2P Video and SFTP classes were not considered for the CorelReff, CRL_PAY and BLINC classifiers, since they are not able to classify this type of traffic.

The results of this experiment favour the proposed classifier. It generally performs as good as the other tools and, in some cases, even better. For example, in the case of VoIP traffic, the values of the *precision* and *recall* are better than for the majority of the classifiers based on ML

techniques, with the exception of the ones based on the SVM and Decision Tree (DT) algorithms. The cases concerning FTP and SSH traffic are also interesting, since the proposed classifier exhibits values of 95% and 92%, respectively, for the *precision* metric. These results are only surpassed by the Bayesian Network (BN) and DT classifiers. On the other hand, all classifiers seem to aggree that the VoIP and P2P *file-sharing* traffic classes are the most difficult to identify. Moreover, the results also show that the *Web-Browsing* traffic is the only class where the classifier using the packet payload presents a better *precision* than the classifier proposed herein. However, this fact may be the consequence of out-of-date signatures.

5.4 Computational Resources Analysis

The last experiment described in this dissertation addresses the computational requirements of the proposed classification means. Devising procedures that are low on those requirements was a preoccupation since the beginning of this research work. This experiment was performed to provide empirical evidence of the several claims made along the discussion.

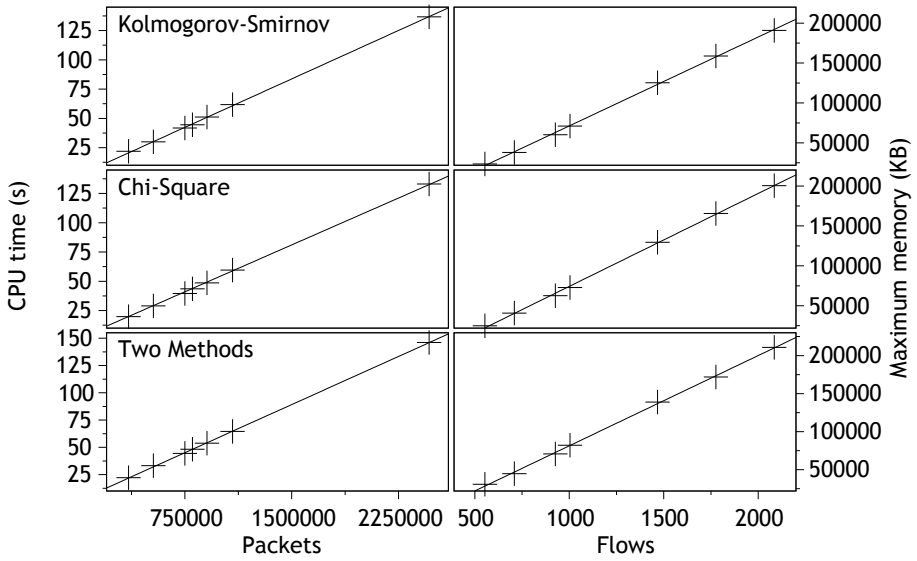

Figure 5.2: Representation of the CPU time and memory consumption as functions of the number of packets and flows, respectively.

The computational resources used by the implemented prototypes can be estimated by discussing the following aspects. First of all, the prototypes separate the traffic into flows and analyze them separately. For each one of the identified flows, a *sliding window* is instantiated to be used in the statistical processing. Only the packet lengths included in the *sliding window* are kept in memory. Since this structure has a constant size, the number of packet lengths kept in memory, for each flow, is constant and does not increase as new packets for the existing flows are processed. Hence, the memory used by the classifier depends linearly on the number of simultaneous flows only. On the other hand, for each processed packet, regardless of the flow to which it belongs to, the classifier has to execute a fixed number of operations. These operations are only executed after the *sliding window* is filled up. Hence, the CPU time should

exhibit a linear dependence to the number of analysed packets, which is critical for real-time operation.

In order to obtain an empirical proof of the previous claims, and even though the implemented prototype could be subject to further optimization, the CPU times and the maximum memory usage for classifying different amounts of data were measured. The GNU time (`/usr/bin/time`) utility was used for this purpose and the results were summarized in the form of a chart in Figure 5.2. Table 5.2 includes the data used to produce that chart in a different format. The linear dependencies are noticeable in all charts, where a line was fitted to the values represented in each of them, for reading convenience.

Table 5.2: CPU time and maximum memory usage of the classifier for different amounts of data (seven distinct trace files) and for each one of the three matching techniques.

Trace	Flows	Packets	Kolmogorov-Smirnov		Chi-Squared		Two Methods	
			CPU time(s)	Memory (KB)	CPU time(s)	Memory (KB)	CPU time(s)	Memory (KB)
File 1	354721	553	21.92	23648	19.84	24960	22.15	30850
File 2	528511	709	30.02	38058	29.02	40724	33.21	44849
File 3	749856	926	41.78	60253	39.61	62584	44.43	70845
File 4	803007	1004	44.51	71034	43.58	72877	48.26	82132
File 5	904537	1467	51.18	125285	48.61	129556	53.84	138874
File 6	1084323	1776	61.72	158781	59.58	165567	64.61	171871
File 7	2464836	2085	136.68	190845	133.06	200416	146.06	210716

5.5 Conclusion

This chapter discussed the results of several experiments, coducted with the objective of testing the prototypes developed in the scope of this work. The two matching approaches and the one combining the other two were first tested with a large dataset of traces to assess their accuracy and get an idea of which one was the best. The results obtained in the first performance evaluation were satisfactory, since the *precision* and *recall* of the classifiers were higher than 75% for all traffic classes considered in the study. The experiment also showed the combination of the two matching approaches, using a simple rule, results in a better classifier.

The best prototype was then compared with other similar tools of the state-of-the-art, namely with the ones integrated in the *NeTraMark* tool. The dataset used in this performance evaluation was different from the one used in the previous experiment. The proposed classifier behaved generally as good as the other classifiers and even better in some cases. The traffic classes used in the experiment are representative of nowadays usage of the Internet, which explains why some classifiers were lacking support to some of those classes.

The next-to-last section describes how the computational requirements were tested in practice, thought it was previously claimed that the involved procedures were modified so as to require a fixed number of operations per packet. The results corroborate those claims, showing that the

CPU depends linearly of the amount of processed packets, while the amount of RAM depends linearly of the number of simultaneous flows under analysis. As such, the proposed classifiers comprise scalable solution, suitable for devices requiring real-time operation.

Table 5.3: Results of the performance evaluation, in terms of *precision* (P) and *recall* (R), for the eleven classification approaches considered within the scope of this work.

	CorelReef		CRL_PAY		BLINC		BN		DT		KNN		NB		NBKE		NN		SVM		Two Methods	
Traffic Class	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**	**P (%)**	**R (%)**
Web Browsing	90.35%	89.06%	98.02%	96.56%	82.81%	80.10%	98.43%	94.12%	98.56%	97.01%	96.24%	94.14%	96.99%	97.01%	98.15%	94.13%	97.89%	95.01%	98.29%	94.48%	97.96%	95.64%
HTTP Download	*Not applicable*		*Not applicable*		*Not applicable*		96.87%	92.18%	97.41%	98.95%	95.34%	93.35%	97.61%	96.41%	96.95%	96.00%	95.93%	97.16%	97.02%	97.60%	96.05%	98.05%
VoIP	*Not applicable*		*Not applicable*		*Not applicable*		80.06%	79.23%	83.84%	80.95%	78.89%	74.98%	79.01%	76.21%	81.56%	79.65%	81.35%	80.48%	83.14%	82.29%	82.61 %	80.12%
Streaming On-Deamand	94.10%	93.92%	84.23%	79.04%	81.32%	80.02%	98.37%	95.23%	95.53%	90.23%	97.70%	95.04%	98.09%	96.05%	98.10%	97.49%	97.89%	97.70%	97.01%	98.16%	97.65%	98.02%
Live Streaming	*Not applicable*		*Not applicable*		*Not applicable*		95.87%	99.75%	98.09%	99.95%	97.04%	94.15%	98.01%	95.04%	97.21%	96.42%	97.02%	93.19%	98.90%	95.93%	98.32%	94.45%
P2P Video	*Not applicable*		*Not applicable*		*Not applicable*		88.27%	82.11%	90.11%	86.15%	83.14%	79.15%	86.46%	85.10%	88.45%	84.95%	89.42%	86.84%	86.73%	84.25%	87.25%	85.07%
P2P *File-Sharing*	47.43%	43.34%	85.10%	86.03%	52.13%	48.22%	93.03%	85.51%	89.41%	84.57%	78.20%	76.90%	83.14%	79.15%	83.06%	80.98%	84.84%	81.05%	86.45%	85.31%	85.32%	81.33%
FTP	92.23%	90.29%	95.13%	89.21%	83.09%	80.63%	96.67%	96.75%	97.17%	98.95%	92.47%	90.02%	93.43%	92.15%	94.68%	91.98%	96.52%	97.26%	94.32%	98.31%	95.87%	99.75%
SSH	88.53%	85.19%	89.29%	83.01%	85.13%	82.21%	93.27%	94.59%	94.41%	95.91%	88.44%	86.35%	90.43%	89.54%	90.78%	87.90%	89.25%	90.91%	92.72%	95.01%	92.87%	94.09%
SFTP	*Not applicable*		*Not applicable*		*Not applicable*		97.73%	98.25%	99.91%	98.53%	98.11%	96.98%	96.75%	95.84%	97.16%	96.16%	97.89%	93.19%	98.45%	94.03%	98.05%	95.35%

Chapter 6

Conclusions and Future Work

This chapter is divided into two sections. The first section presents the main conclusions of the research work developed during this master's degree programme, while the second section contains suggestions for research topics related to this work, which can be addressed in the future.

6.1 Main Conclusions

The research work described in this dissertation tackled the problem of accurately and efficiently classify network traffic. As shown in chapter 2, this is an area of knowledge that has been evolving quickly in the last few years, almost on par with the evolution that computer networks have been subject to. The first classification approach was solely based on the transport layer port numbers, but it is now obsolete. Nowadays, DPI techniques are believed to comprise the most accurate means to classify traffic, unless encryption is being used to hide the contents of the IP packets. Some of the most recent contributions tackle precisely that problem and elaborate on techniques that do not rely on the payload of packets to classify traffic, sometimes sacrificing accuracy and granularity. The classifiers described in this dissertation divide the network traffic into flows using the information in the headers of the IP packets, to then segregate them with basis only on the lengths of the packets.

All the objectives defined in the beginning of this work were fulfilled, though it was not possible to pay attention to some small details that appeared along the research path. Resorting to several techniques and implementation choices, it was possible to address the majority of challenges associated with building a traffic classifier. For example, by using the concept of a fixed sized *sliding window*, it was possible to implement matching procedures with linear computational requirements and capable of producing results on a packet-by-packet manner. Alone, these two characteristics make of this solution scalable and suitable to be integrated in systems demanding real-time operation. The classifiers were not tested in very high speed network aggregation nodes but, given that their requirements are lower than, for example, DPI techniques, it is safe to conclude that they may be integrated in any system capable of running the latter.

The functioning of the proposed classifiers depends of previously devised signatures, which are composed of values representing the empirical distributions of the packet lengths for the

classes one wants to identify. Constructing these signatures requires building a training set with an established ground truth. It can then be argued that, in order to skip or deceive the classifier, one just has to change the empirical distributions of the packet lengths. While that might be true, one has to also take into account that, in order to achieve that effect, random garbage may have to be added to the packets or empty packets may have to be injected into the respective flows, possibly decreasing the performance of the respective application.

The procedure for matching the signatures with the empirical distribution of traffic flows was inspired in the functioning of statistical tests. These tests use statistics that measure the distance between the distributions under analysis. Though one of such statistics would be enough to construct the classifier, it was decided to use the two most popular ones: the χ^2 and the (Kolmogorov-Sminov test) D statistic. This decision resulted in the implementation of three different prototypes, one for each one of the statistics and a third one combining those two. The three prototypes were then tested and compared. They all produced very good results for the several applications considered in this study, with values for the *precision* and *recall* generally above 80%. Nonetheless, when compared with each other, the classifier using the D statistic is the worst, while the one combining the two statistics is the best, representing thus the main contribution of this work.

The distribution of a random variable aggregates a lot of information regarding the behaviour of that variable. It is assumed that this fact is behind the excellent results obtained during the performance evaluation of the third prototype. When compared with similar tools of the state-of-the-art, available via the *NeTraMark* benchmarking tool, the classifier proved to be as good as the others, or even better in some cases, exhibiting values larger than 82% for the *precision* and *recall* for all traffic classes. For example, one of the most interesting results concerns VoIP traffic, where the *precision* of the proposed classifier is only overcome by the SVM and DT classifiers. All the tests were conducted using large datasets, captured in a laboratory environment and composed of traffic traces representative of nowadays usage of the Internet. For the sake of consistency, the dataset that was used to construct the signatures was different from the ones used in the experiments regarding the performance of the classifiers. Additionally, the dataset that was used to train the classifiers based on ML was also different from all the others. If the results regarding the accuracy of the proposed classifier are added to the ones concerning its computational requirements, it is possible to conclude that it comprises a very attractive solution to the problem at hands, though not yet perfect.

6.2 Future Work

When devising traffic monitoring and analysis solutions, one of the major problems that researchers and manufacturers face is scalability with respect to the amount of traffic. It is difficult to, for example, devise a classifier capable of operating in real-time at any data rate,

unless the computational resources can be scaled up accordingly. Thought this may not comprise a problem in the future, given that the prices of many components are decreasing and cloud computing solutions may enable resources to be provided as needed, the power consumption of the methods is still a problem. In this work, compliance with real-time operation was addressed by devising methods able to produce results in a packet-by-packet manner (i.e., at any given moment of the duration of the flow) and very low on resources. Nonetheless, the computational requirements of the classifiers is still linearly dependent of the number of flows. For high speed networks, a sampling technique may be required to lower the computational burden and, as such, a possible future line of work concerns researching the best way to discard or accept packets for processing. The sampling rate may be, for example, be expressed in terms of the computational load of the system or of the effective data rate, or adjusted using artificial intelligence techniques.

In order to this work to progress, several assumptions regarding input parameters of the proposed classifiers had to be made, postponing the study of the impact of changing such input parameters into the future. The two parameters that may have impact on the results are the *bucket size* and the *window size*. The former defines the intervals of values for which the empirical distributions are built, while the latter defines the number of values taken into account when matching live traffic. During some of the experiments, it was noticed that a too small or too large *bucket size* impacts the accuracy negatively. It was also noticed that using a large *window size* would not necessarily result in added benefit. Nonetheless, the real relation between the parameters and performance was not studied, being left as a future research direction.

Another interesting aspect to explore in the future is the possibility to automate or semi-automate the creation of new signatures for unknown traffic, as well as adapting the existing signatures during normal operation. Such features would improve the classifier ability to react to behavioral changes along the time or to the appearance of new protocols and applications. The modifications to the existing signatures could be triggered after successful classifications or in a timely fashion.

Since the proposed classification methods act *in the dark*, they cannot be as accurate or granular as DPI techniques, but their accuracy may be improved after combining their output with other TCD or DPI techniques. The integration of several classifiers into a single system is dominated by several challenges. Two of the most interesting challenges concern the order by which the classifiers can be applied to reduce computational requirements and how to combine contradicting results returned by two different methods.

Thought a significant number of applications, representative of nowadays usage of the Internet, was used to perform and validate this work, no analysis were made to traffic generated by

online gaming applications, which are very popular nowadays. Further tests to the proposed classifiers including that class of traffic are planned.

Within the scope of this work, *Netramark*, a benchmarking tool, was used to compare the proposed prototypes with classifiers described in the literature. This tool constitutes an excellent resource for practitioners working on this area, and it reflects a significant research and engineering work, but it is currently lacking maintenance. Installing this tool on up-to-date OSs requires the user to search and install several dependencies and downgrade some existing packages. Producing a new version of *Netramark* using up-to-date dependencies comprises thus an interesting engineering project, specially if it includes the integration of more state-of-the-art classifiers, presented since the last time it was updated.

Last, but not least, it could be useful to other practitioners to disseminate the proposed classifiers and source code along with a graphical interface for adjusting some of the parameters (for example, the bucket and window size). For that, a more concise documentation is needed also.

Bibliography

[AMG07] T. Auld, A.W. Moore, and S.F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on Neural Networks*, 18(1):223 --239, jan. 2007. 13

[AZH09] R. Alshammari and A.N. Zincir-Heywood. Machine learning based encrypted traffic classification: Identifying ssh and skype. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009*, pages 1 --8, july 2009. 13

[BHP07] Genevieve Bartlett, John Heidemann, and Christos Papadopoulos. Inherent behaviors for on-line detection of peer-to-peer file sharing. In *Proceedings of the IEEE Global International Symposium*, pages 55--60, Anchorage, AK, USA, May 2007. 12

[BMM+07] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli. Revealing Skype traffic: When randomness plays with you. *ACM SIGCOMM Computer Communications Review*, 37(4):37--48, October 2007. xi, 11, 13

[BMM+09] A. Bianco, G. Mardente, M. Mellia, M. Munafo, and L. Muscariello. Web user-session inference by means of clustering techniques. *IEEE/ACM Transactions on Networking*, 17(2):405--416, 2009. 12

[BZTH07] Liu Bin, Li Zhi-Tang, and Tu Hao. A methodology for P2P traffic measurement using application signature work-in-progress. In *Proceedings of the Second International Conference on Scalable Information Systems (InfoScale '07)*, volume 304 of *ACM International Conference Proceeding Series*, pages 1--2, Suzhou, China, June 2007. ICST, Brussels, Belgium. 10

[CCR10] Niccolò Cascarano, Luigi Ciminiera, and Fulvio Risso. Improving cost and accuracy of DPI traffic classifiers. In *Proceedings of the 25th ACM Symposium Application Computing (SAC 2010)*, Sierre, Switzerland, March 2010. 10

[CDGS07] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Traffic classification through simple statistical fingerprinting. *ACM SIGCOMM Computer Communications Review*, 37(1):5--16, January 2007. 12

[CEG+09] Niccolò Cascarano, Alice Este, Francesco Gringoli, Fulvio Risso, and Luca Salgarelli. An experimental evaluation of the computational cost of a DPI traffic classi-

fier. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2009)*, Honolulu, HI, USA, November--December 2009. 10

[CF09] G. W. Corder and D. I. Foreman. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. New Jersey: Wiley, 2009. 25

[cor11] CoralReef Software Suite, July 2011. [Accessed 18 June 2013]. Available from: `http://www.caida.org/tools/measurement/coralreef/`. 35

[CPF09a] David A. Carvalho, Manuela Pereira, and Mário M. Freire. Detection of peer-to-peer TV traffic through deep packet inspection. In *Acta da 9a Conferência sobre Redes de Computadores*, page 6, Oeiras, Portugal, October 2009. INESC-ID and Instituto Superior Técnico. 11

[CPF09b] David A. Carvalho, Manuela Pereira, and Mário M. Freire. Towards the detection of encrypted BitTorrent traffic through deep packet inspection. In *Proceedings of the International Conference on Security Technology (SecTech 2009)*, volume 58 of *Communications in Computer and Information Science*, pages 265--272, Jeju Island, Korea, December 2009. Springer-Verlag, Berlin Heidelberg. 10

[CRE+10] Niccolò Cascarano, Fulvio Risso, Alice Este, Francesco Gringoli, Alessandro Finamore, and Marco Mellia. Comparing P2PTV traffic classifiers. In *Proceedings of the IEEE International Conference on Communications (ICC 2010)*, Cape Town, South Africa, May 2010. IEEE. 10

[Dav] David. *Towards the Detection of Encrypted Peer-to-Peer File Sharing Traffic and Peer-to-Peer TV Traffic Using Deep Packet Inspection Methods*. PhD thesis, University of Beira Interior. xix, 8

[DCGS09] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Tunnel Hunter: Detecting application-layer tunnels with statistical fingerprinting. *Elsevier Computer Networks*, 53(1):81--97, January 2009. 12

[DPTP10] Peter Dorfinger, Georg Panholzer, Brian Trammell, and Teresa Pepe. Entropy-based traffic filtering to support real-time Skype detection. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC '10)*, pages 747--751, Caen, France, June--July 2010. 11

[DWF03] Christian Dewes, Arne Wichmann, and Anja Feldmann. An analysis of Internet chat systems. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC 2003)*, pages 51--64, Miami Beach, FL, USA, October 2003. 10

[EIPF11] A.F. Esteves, P.R.M. Inácio, M. Pereira, and M.M. Freire. On-line detection of encrypted traffic generated by mesh-based peer-to-peer live streaming applications: The case of goalbit. In *Proceedings of the 10th IEEE International Symposium Network Computing and Applications (NCA), 2011*, pages 223--228, 2011. 11

[EMA06] Jeffrey Erman, Anirban Mahanti, and Martin Arlitt. Internet traffic identification using machine learning. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2006)*, pages 1--6, San Francisco, CA, USA, November-December 2006. IEEE. 13

[FMMR10] A. Finamore, M. Mellia, M. Meo, and D. Rossi. Kiss: Stochastic packet inspection classifier for udp traffic. *IEEE/ACM Transactions on Networking*, 18(5):1505 --1515, oct. 2010. 13

[FZS08] Emanuel P. Freire, Artur Ziviani, and Ronaldo M. Salles. Detecting VoIP calls hidden in web traffic. 5(4):204--214, December 2008. 12

[GCRHMA+06] F.J. Gonzalez-Castano, P.S. Rodriguez-Hernandez, R.P. Martinez-Alvarez, A. Gomez, I. Lopez-Cabido, and J. Villasuso-Barreiro. Support vector machine detection of peer-to-peer traffic. In *Proceedings of the IEEE International Conference on Computational Intelligence for Measurement Systems and Applications,2006*, pages 103 --108, july 2006. 13

[GHN+03] Alexandre Gerber, Joseph Houle, Han Nguyen, Matthew Roughan, and Subhabrata Sen. P2P, the gorilla in the cable. In *Proceedings of the National Cable & Telecommunications Association (NCTA)*, pages 8--11, Chicago, IL, USA, June 2003. 7

[GIF+08] J. V. P. Gomes, P. R. M. Inácio, M. M. Freire, M. Pereira, and P.P. Monteiro. Analysis of peer-to-peer traffic using a behavioural method based on entropy. In *Proceedings of the Performance Computing and Communications Conference., 2008. IPCCC 2008. IEEE International*, pages 201--208, December 2008. 12

[GIF+10] João V. P. Gomes, Pedro R. M. Inácio, Mário M. Freire, Manuela Pereira, and Paulo P. Monteiro. The nature of peer-to-peer traffic. In Xuemin Shen, Heather Yu, John Buford, and Mursalin Akon, editors, *Handbook of Peer-to-Peer Networking*, pages 1231--1252. Springer US, New York, NY, USA, 2010. xxi, 8

[GQ08] Zhenbin Guo and Zhengding Qiu. Identification peer-to-peer traffic for high speed networks using packet sampling and application signatures. In *Proceedings of the*

9th International Conference on Signal Processing (ICSP 2008), pages 2013--2019, Beijing, China, October 2008. 10

[IcKF+11] Marios Iliofotou, Hyun chul Kim, Michalis Faloutsos, Michael Mitzenmacher, Prashanth Pappu, and George Varghese. Graption: A graph-based P2P traffic classification framework for the internet backbone. *Elsevier Comp. Networks*, 55(8):1909--1920, June 2011. 12

[JG10] Weirong Jiang and M. Gokhale. Real-time classification of multimedia traffic using fpga. In *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), 2010*, pages 56 --63, 31 2010-sept. 2 2010. 13

[JT08] Wolfgang John and Sven Tafvelin. Heuristics to classify Internet backbone traffic based on connection patterns. In *Proceedings of the International Conference on Information Networking (ICOIN 2008)*, pages 1--5, Busan, Korea, January 2008. 11

[JXXLH10] Li Jin, Zhang Xin, Zuo Xiao-Liang, and Wang Hui. Using packet size distribution to identify p2p-tv traffic. In *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2010*, pages 150 --155, oct. 2010. xi, 13

[KBB+04] Thomas Karagiannis, Andre Broido, Nevil Brownlee, kc claffy, and Michalis Faloutsos. Is P2P dying or just hiding? In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2004)*, volume 3, pages 1532--1538, Dallas, TX, USA, November--December 2004. 7, 10

[KFc04] Thomas Karagiannis, Andre Broido Michalis Faloutsos, and Kc claffy. Transport layer identification of P2P traffic. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC 2004)*, pages 121--134, Taormina, Sicily, Italy, October 2004. ACM, New York, NY, USA. 7, 35

[KPF05] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. BLINC: Multilevel traffic classification in the dark. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, volume 35, pages 229--240, Philadelphia, PA, USA, August 2005. 2, 12, 35

[KW02] Balachander Krishnamurthy and Jia Wang. Traffic classification for application specific peering. In *Proceedings ACM SIGCOMM Internet Measurement Workshop (IMW 2002)*, pages 179--180, Marseille, France, November 2002. 7

[l7f09] L7-filter, Application Layer Packet Classifier for Linux, January 2009. [Accessed 10 April 2013]. Available from: http://l7-filter.sourceforge.net. 11

[LBBSS02] Nathaniel Leibowitz, Aviv Bergman, Roy Ben-Shaul, and Aviv Shavit. Are file swapping networks cacheable? Characterizing P2P traffic. In *Proceedings of the 7th International Workshop on Web Content Caching and Distribution (WCW)*, Boulder, CO, USA, August 2002. 7

[LKB+11] Suchul Lee, Hyunchul Kim, Dhiman Barman, Sungryoul Lee, Chong-kwon Kim, Ted Kwon, and Yanghee Choi. Netramark: a network traffic classification benchmark. *SIGCOMM Computer Communication Review*, 41(1):22--30, January 2011. Available from: http://doi.acm.org/10.1145/1925861.1925865. 35

[LLHL12] Chun-Nan Lu, Ying-Dar Lin, Chun-Ying Huang, and Yuan-Cheng Lai. Session level flow classification by packet size distribution and session grouping. *International Conference on Advanced Information Networking and Applications Workshops*, 0:221--226, 2012. xi, 13

[LLL+09] Ying-Dar Lin, Chun-Nan Lu, Yuan-Cheng Lai, Wei-Hao Peng, and Po-Ching Lin. Application classification using packet size distribution and port association. *Journal of Network and Computer Applications*, 32(5):1023--1030, September 2009. Next Generation Content Networks. xi, 12, 13

[LLM10] Feng Lu, Xiao-Lei Liu, and Zhi-Nan Ma. Research on the characteristics and blocking realization of Skype protocol. In *Proceedings of the International Conference on Electrical and Control Engineering (ICECE 2010)*, pages 2964--2967, Wuhan, China, June 2010. 11

[Lui10] Luis Martin Garcia. TCPDUMP/LIBPCAP public repository, 2010. [Accessed 18 June 2013]. Available from: http://www.tcpdump.org/. 31

[MKK+01] David Moore, Ken Keys, Ryan Koga, Edouard Lagache, and kc claffy. The CoralReef software suite as a tool for system and network administrators. In *Proceedings of the 15th USENIX System Administration Conference (LISA '01)*, pages 133--144, San Diego, CA, USA, December 2001. USENIX Association, Berkeley, CA, USA. 8

[MP05] Andrew W. Moore and Konstantina Papagiannaki. Toward the accurate identification of network applications. In Constantinos Dovrolis, editor, *Proceedings of the Passive and Active Measurement Conference (PAM 2005)*, volume 3431 of *Lecture Notes in Computer Science*, pages 41--54, Boston, USA, 2005. 7, 10

[MP11] Sándor Molnár and Marcell Perényi. On the identification and analysis of Skype traffic. *International Journal of Communication Systems*, 24(1):94--117, January 2011. 11

[net13] L7-Netpdl Classifier, 2013. [Accessed 20 April 2013]. Available from: `http://netgroup.polito.it/research-projects/l7-traffic-classification`. 11

[NGFI13] Miguel Neto, João V. Gomes, Mário M. Freire, and Pedro R.M. Inácio. Real-time traffic classification based on statistical tests for matching signatures with packet length distributions. In *Proceedings of the 19th IEEE Workshop on Local Metropolitan Area Networks (LANMAN), 2013*, pages 1--6, 2013. x, 5

[OD08] David L. Olson and Dursun Delen. *Advanced Data Mining Techniques*. Springer, 1st edition, March 2008. 34

[ope13] nDPI - Open and Extensible GPLv3 Deep Packet Inspection Library, 2013. [Accessed 19 June 2013]. Available from: `http://www.ntop.org/products/ndpi/`. 11

[PBL+03] D.J. Parish, K. Bharadia, A. Larkum, I.W. Phillips, and M.A. Oliver. Using packet size distributions to identify real-time networked applications. *IEEE Proceedings-Communications*, 150(4):221 -- 227, August 2003. xi, xii, 13, 20

[PF09] Francesco Palmieri and Ugo Fiore. A nonlinear, recurrence-based approach to traffic classification. *Elsevier Computer Networks*, 53(6):761--773, April 2009. 12

[SEJK08] Randy Smith, Cristian Estan, Somesh Jha, and Shijin Kong. Deflating the big bang: Fast and scalable deep packet inspection with extended finite automata. *ACM SIGCOMM Computer Communication Review*, 38(4):207--218, October 2008. 10

[SFKT06] Kyoungwon Suh, Daniel R. Figueiredo, Jim Kurose, and Don Towsley. Characterizing and detecting Skype-relayed traffic. In *Procedings of the 25th IEEE Conference on Computer Communications (INFOCOM 2006)*, pages 1--12, Barcelona, Spain, April 2006. 11

[SHR+09] Philipp Svoboda, Esa Hyytiä, Fabio Ricciato, Markus Rupp, and Martin Karner. Detection and tracking of Skype by exploiting cross layer information in a live 3G network. In Maria Papadopouli, Philippe Owezarski, and Aiko Pras, editors, *Proceedings of the First International Workshop on Traffic Monitoring and*

Analysis (TMA '09), volume 5537 of *Lecture Notes in Computer Science*, pages 93--100, Aachen, Germany, May 2009. 11

[SOG07] Douglas C. Sicker, Paul Ohm, and Dirk Grunwald. Legal issues surrounding monitoring during network research. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 141--148, New York, NY, USA, 2007. ACM. Available from: http://doi.acm.org/10.1145/1298306.1298307. 2

[SSW04] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. Accurate, scalable in-network identification of P2P traffic using application signatures. In *Proceedings 13th International Conference on World Wide Web (WWW '04)*, pages 512--521, New York, NY, USA, May 2004. 10

[SYP+10] Runyuan Sun, Bo Yang, Lizhi Peng, Zhenxiang Chen, Lei Zhang, and Shan Jing. Traffic classification using probabilistic neural networks. In *Proceedings of the Sixth International Conference on Natural Computation (ICNC), 2010*, volume 4, pages 1914 --1919, aug. 2010. 13

[wek12] WEKA - Machine Learning Project, 2012. [Accessed 30 April 2013]. Available from: http://www.cs.waikato.ac.nz/ml/index.html. 35

[wir13] Wireshark, Go deep, June 2013. [Accessed 18 June 2013]. Available from: http://www.wireshark.org. 8

[WP10] Xiaoming Wang and David J. Parish. Optimised multi-stage tcp traffic classifier based on packet size distributions. In *Proceedings of the 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service*, CTRQ '10, pages 98--103, Washington, DC, USA, 2010. IEEE Computer Society. Available from: http://dx.doi.org/10.1109/CTRQ.2010.24. xi, 13

[ZZZY10] Dongyan Zhang, Chao Zheng, Hongli Zhang, and Hongliang Yu. Identification and analysis of Skype peer-to-peer traffic. In *Proceedings 5th International Conference on Internet and Web Applications and Services (ICIW 2010)*, pages 200--206, Barcelona, Spain, May 2010. 11

Appendix A

Signatures

A.1 Example of Signatures

The listings included in this appendix (listings A.1 and A.2) present the contents of the files for two signatures devised along this work. As discussed in chapter 3, signatures are files containing the name of the originating application and the empirical cumulative distribution of the IP packet lenghts.

```
VoIP
0.0000
0.0000
0.0000
0.0376
0.9248
0.9248
0.9323
0.9323
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9398
0.9474
0.9549
0.9549
0.9549
0.9549
0.9549
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9624
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9699
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
0.9774
1.0000
```

Listing A.1: Example of signature defined to VoIP traffic.

```
P2P File-Sharing
0.0000
0.0000
0.0000
0.2595
0.4517
0.4631
0.4683
0.4728
0.4988
0.5053
0.5129
0.5180
0.5214
0.5230
0.5237
0.5254
0.5263
0.5309
0.5318
0.5325
0.5338
0.5345
0.5346
0.5349
0.5359
0.5360
0.5361
0.5368
0.5372
0.5374
0.5374
0.5385
0.5492
0.5495
0.5508
0.5515
0.5520
0.5529
0.5621
0.5982
0.5985
0.5995
0.6026
0.6036
0.6046
0.6049
0.6051
0.6061
0.6065
0.6142
0.6179
0.6187
0.6190
0.6191
0.6198
0.6200
0.6201
0.6204
0.6211
0.6212
0.6213
0.6220
0.6232
0.6233
0.6234
0.6241
0.6243
0.6248
0.6256
0.6265
0.6266
0.6270
0.6278
0.6280
0.6283
0.6321
0.6328
0.6330
0.6331
0.6339
0.6340
0.6340
0.6418
0.6435
0.6435
0.6442
0.6444
0.6445
0.6449
0.6651
0.6658
0.7139
0.7151
0.7158
0.7159
0.7787
0.8459
0.8953
0.8962
1.0000
```

Listing A.2: Example of signature defined for P2P *file-sharing* traffic.