UNIVERSIDAD DE GRANADA
Departamento de Teoría de la Señal,
Telemática y Comunicaciones

Programa de Doctorado en
Tecnologías de la Información y la Comunicación

NOVEL APPROACHES IN TRAFFIC CLASSIFICATION

Tesis Doctoral

Jawad Khalife

Directores:

Jesús E. Díaz Verdejo
Amjad Hajjar

Diciembre, 2015

Editor: Universidad de Granada. Tesis Doctorales
Autor: Jawad Khalife
ISBN: 978-84-9125-852-0
URI: http://hdl.handle.net/10481/43637

El doctorando Jawad Khalife y los directores de la tesis Jesús E. Díaz-Verdejo y Amjad Hajjar:

Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, 23 de noviembre de 2015

Director/es de la Tesis | Doctorando

Fdo.: Jesús E. Díaz-Verdejo | Fdo.: Jawad Khalife

Amjad Hajjar

Acknowledgements

After 4 years of collaborative work, this thesis has been fulfilled with success.

First, I dedicate this thesis to the soul of my dad, and to the future of my children. My great thanks are to my wife, mom, and family members who supported me all the way as a PhD student by providing the appropriate working environment.

Second, I would like to thank my supervisors, prof. Jesús Díaz-Verdejo and Dr. Amjad Hajjar for leading my efforts to success. The time they dedicated for adjusting experimental work, their comments and criticism throughout the research and writing process are undoubtedly key factors for my success, opening the door for my post-doctoral research. My deepest thanks are to prof. Jesús who dedicated his time and provided me with invaluable remarks and professional guidance. His assistance covered paper editing, experimental results verification and analysis, following-up with journal submission and answering reviewers, not to mention his remarks on the thesis dissertation, structure, content and format.

Under prof. Jesús guidance, I was able to write my first congress papers and to get my first journals accepted. His constructive and positive criticism together with his great academic experience, helped me in developing and cumulating great experience in editing skills and research methodologies, which are highly important both on the academic and research levels.

Third, my special thanks are to Dr. Amjad, who dedicated a full sabbatical year for handling my research questions and conducting many of the experiments. His knowledge in artificial intelligence and programming background were behind my success in implementing critical parts of the experimental testbed used throughout my research work. His innovative and inspiring ideas were at the core of my thesis work.

Last, but certainly not least, I must thank the staff in the UGR doctoral school, and CITIC research center, especially those who helped me in ensuring an affordable student residence and an appropriate working environment in terms of office facilities, library access, computer and printing resources. Finally, I would like to thank the thesis committee members for the time they dedicated for reviewing the dissertation and during the thesis defense.

Jawad Khalife
Granada, December 2015

To the soul of my dad,
to my beloved mom and wife...
for their devotion and dedication.

Abstract

Identifying Internet traffic applications is centric to many network security and management tasks. With the steady emergence of Internet applications, encryption and obfuscation techniques, researchers are facing various challenges in accurately identifying different applications. An optimal traffic-classification model has yet to be defined despite of the effort devoted by the research community in the last decade.

This thesis aims to provide an analytical review of existing traffic identification methods, while suggesting novel enhancements and approaches.

First, with the remarkably increasing number of papers in the literature covering traffic classification, we survey most recent works and propose a systematic multilevel taxonomy. Expressed in a consistent terminology, our proposed taxonomy can promote and unify the research efforts in designing the best future traffic identification model, which we describe and characterize while underlining main research requirements. Moreover, the different categories (e.g. payload-based, statistical, machine learning, graphical, etc.) are compared, as found in the literature, in terms of performance, accuracy, ability to detect critical applications like peer-to-peer, and other relevant criteria. The current research trend is also analyzed in the light of the surveyed works.

Second, with the lack of reference datasets for evaluating and comparing traffic classification methods, we collected real traffic sets in a significant volume and extracted all the parameters relevant to the classification process using customized tools.

Third, we assess payload based traffic identification methods and propose an optimization that can best achieve the trade-off between the classification accuracy and the user's privacy protection. For this purpose, we assess the performance of Deep Packet Inspection, and present a customized sampling policy for the traffic payloads. According to our testbed, promising results, related to the classification time gain, were obtained at the cost of less inspected payload while maintaining the classification accuracy.

Finally, we assess blind identification models. First, we explore the discriminative power of traffic features at the application layer by proposing a flow-based classifier relying on application message lengths analysis. Our approach analyzes application layer messages without breaching the user privacy. For this model, we propose a novel flow-based classifier using multi-modal distributions. Evaluated on a real captured dataset, the results evidence the goodness of the proposal and the existence of discriminative information regarding traffic classification in the sizes of the exchanged messages.

Then, we discuss an extended model for multi-label host classification, as most of the current host-based classification models do not reflect real usage scenarios, where a host might be using more than one application. For this purpose, we choose a host classification method, based on graphical techniques, that we enhance and extend to handle multi-label classification. Our proposal

can regarded as an attempt to radically change the conventional view of mono-label host classification. Our results show an improvement in classification accuracy for most protocols including peer-to-peer.

Resumen

La identificación de la aplicación asociada a los elementos que componen el tráfico circulante en Internet constituye una tarea central para la gestión y seguridad de la red. Con la aparición constante de nuevas aplicaciones en Internet, el uso de técnicas de cifrado y la ofuscación del tráfico, son varios e importantes los retos a los que se enfrentan los investigadores para desarrollar métodos precisos de clasificación. En este sentido, a pesar el enorme esfuerzo realizado últimamente por la comunidad investigadora, aún no se ha definido un procedimiento adecuado para ello.

El presente trabajo tiene como objetivo proporcionar una revisión analítica de los métodos existentes para la identificación del tráfico, así como proponer y evaluar mejoras y aproximaciones novedosas.

En primer lugar, con el notable aumento del número de trabajos en la literatura que abordan la clasificación de tráfico, examinamos las contribuciones más recientes y proponemos una taxonomía sistemática multinivel. Expresada en una terminología consistente, la taxonomía propuesta pretende establecer un marco de referencia unificado en dicho campo que facilite la comparativa de los sistemas, tanto existentes como futuros. En este sentido, se describen y caracterizan las propiedades y características deseables de estos sistemas, al tiempo que se subrayan los requisitos principales para la investigación en este ámbito. Por otra parte, esta taxonomía se utiliza para como vehículo para organizar y describir las aproximaciones existentes en la bibliografía en términos de técnica, rendimiento, precisión y capacidad de detectar aplicaciones críticas como peer-to-peer. También se analizan las tendencias de la investigación actual a la luz de las contribuciones estudiadas.

En segundo lugar, debido a la inexistencia de conjuntos de datos de referencia para la evaluación y comparación de métodos de clasificación de tráfico, se estableció un escenario de trabajo en el que se adquirieron conjuntos de tráfico real en un volumen relevante, que es varios órdenes de magnitud superior a los usados en la bibliografía. Este tráfico se caracterizó mediante la obtención de todos los parámetros relevantes de cara al proceso de clasificación utilizando para ello herramientas personalizadas al efecto.

En tercer lugar, evaluamos los métodos de identificación de tráfico basados en la inspección de la carga útil y proponemos un método de optimización que mejora el rendimiento computacional y reduce la invasión de la privacidad de los usuarios. La propuesta se basa en la utilización de varios métodos de muestreo de las cargas útiles, que son analizados y evaluados experimentalmente para determinar la mejor política de muestreo de entre las propuestas. Los resultados obtenidos evidencian que, mediante el muestreo seleccionado, se puede conseguir una reducción en el número de bytes analizados, con la consiguiente disminución del coste computacional y mejora en la privacidad, sin degradar la precisión de la clasificación.

Finalmente, proponemos y evaluamos un método de identificación ciego,

esto es, que no inspecciona las cargas útiles. Para ello, en primer lugar, se explora el poder discriminativo de las características de tráfico a nivel de la capa de aplicación mediante la propuesta de un clasificador de flujos basado en el análisis de la longitud de los mensajes iniciales. Nuestro enfoque analiza estos mensajes de la capa de aplicación sin violar la privacidad del usuario. El clasificador propuesto utiliza modelado de Markov y distribuciones multimodales para representar la secuencia de longitudes de los mensajes. Los resultados obtenidos ponen de manifiesto la bondad de la propuesta y la existencia de información discriminatoria respecto a la clasificación del tráfico en los tamaños de los mensajes intercambiados.

A continuación, se extiende un método basado en la clasificación de host para proporcionar una clasificación multi-etiqueta. Esta propuesta se realiza a partir de la constatación de que la mayoría de los sistemas de clasificación basados en host actuales no reflejan situaciones de uso real, en la que cada host podría estar generando tráfico de más de una aplicación. En consecuencia, se elige como punto de inicio un método de clasificación basado en técnicas gráficas, sobre el que se realizan con éxito algunas propuestas de mejora, y se realiza una extensión del mismo para manejar una clasificación multi-etiqueta. Nuestra propuesta puede considerarse como un intento de cambiar radicalmente la visión convencional de la clasificación de host en base a una única etiqueta. Los resultados muestran una mejora en la precisión de la clasificación para la mayoría de los protocolos, incluyendo los protocolos peer-to-peer que son, a priori, los que entrañan mayor dificultad de clasificación.

Contents

1 Introduction **1**
- 1.1 Introduction . . . 1
- 1.2 Network Traffic classification . . . 4
  - 1.2.1 Definition and Goals . . . 4
  - 1.2.2 Basic Approaches . . . 5
  - 1.2.3 Challenges . . . 6
- 1.3 Classification of P2P Applications . . . 8
  - 1.3.1 Relevance of P2P Traffic Identification . . . 9
  - 1.3.2 P2P Architectural Model and Discriminative Characteristics . . . 10
- 1.4 Thesis Scope and Contributions . . . 11
- 1.5 Thesis Structure . . . 13

2 State of the Art in Traffic Identification Methods **15**
- 2.1 A Novel Multilevel Taxonomy for Traffic Classification Methods 16
  - 2.1.1 Categorization by the Classification's Output . . . 19
    - 2.1.1.1 Traffic Objects . . . 19
    - 2.1.1.2 Traffic Classes . . . 20
  - 2.1.2 Categorization by the Input . . . 22
  - 2.1.3 Categorization by the Technique . . . 23
    - 2.1.3.1 Payload Inspection Techniques . . . 24
    - 2.1.3.2 Simple Statistical Techniques . . . 26
    - 2.1.3.3 Machine Learning Techniques . . . 30
    - 2.1.3.4 Graphical Techniques . . . 40
    - 2.1.3.5 Hybrid and Miscellaneous Techniques . . . 44
- 2.2 Comparison and Evaluation of Traffic Classification Methods . 45
  - 2.2.1 Comparison and Limitations of Existing Methods . . . 46
  - 2.2.2 Vendor Classification Engines . . . 48
  - 2.2.3 Requirements for Future Benchmarks . . . 49
- 2.3 Open Challenges in Traffic Classification . . . 50

3 The Experimental Setup **55**
3.1 Overview of the Data Acquisition . . . 55
3.2 Acquisition of Traffic . . . 57
3.2.1 Analysis of Traffic Requirements . . . 58
3.2.2 Scenario and Obtention of the Traffic . . . 60
3.2.3 Captured Datasets . . . 61
3.2.4 Preprocessing of Capture Files . . . 64
3.3 Labeling and Classification . . . 65
3.3.1 OpenDPI Classifier . . . 65
3.3.2 Results and Analysis . . . 67
3.4 Traffic Parametrization . . . 69
3.5 The Evaluation Method . . . 72

4 DPI Optimization Through Minimum Payload Disclosure **77**
4.1 DPI Optimization . . . 78
4.1.1 A Simplified Analogy . . . 78
4.1.2 Sampling Policies . . . 80
4.1.3 DPI Modules and Cost Analysis . . . 82
4.2 Related Work . . . 84
4.3 Assessment of DPI-oriented Sampling Scheme . . . 87
4.3.1 Per-packet based Sampling . . . 89
4.3.1.1 Experimental Results . . . 89
4.3.2 Per-flow Based Sampling . . . 91
4.3.2.1 Experimental Results . . . 93
4.3.3 Comparison of Packet and Flow Truncation Sampling Policies . . . 97
4.3.4 Combined Sampling Schemes . . . 101
4.3.4.1 Non-Contiguous Sampling Results (Method I) 103
4.3.4.2 Contiguous Sampling Results (Method II) . . . 104
4.4 Comparison of Combined Sampling Policies . . . 106
4.5 Optimized DPI Sampling . . . 111
4.5.1 Proposed DPI Sampling . . . 112
4.5.2 Advantages of the Proposed DPI Sampler . . . 113
4.5.3 Discussion . . . 114

5 Network Traffic Application Identification Based on Message Size Analysis **117**
5.1 Related Works . . . 120
5.2 Protocol Methods and Messages: an Insight . . . 124
5.2.1 Protocol Methods . . . 124
5.2.2 Sequence of Methods . . . 125
5.2.3 Visibility of Message Sizes from Transport Headers . . . 127
5.2.4 Message Size Scaling . . . 127

5.2.5 Potential of Message Size Analysis for Identifying TCP Applications . . . 128
5.3 Flow Classification Based on Message-Size Vectors . . . 128
5.3.1 System architecture . . . 129
5.3.1.1 Features Extraction . . . 130
5.3.1.2 Sequence Evaluation . . . 131
5.3.1.3 Classification . . . 133
5.4 Training of the System . . . 134
5.4.1 Metric and Normalization of Message Sizes . . . 135
5.4.2 Gaussian Mixture Model . . . 136
5.4.3 Profiling the Applications . . . 137
5.4.4 Clustering and Estimation of Gaussian Mixtures . . . 138
5.5 Experimental Results and Assessment . . . 139
5.5.1 Test bed . . . 139
5.5.2 Experimental Results . . . 140
5.5.2.1 Training Phase Results . . . 140
5.5.2.2 Classification Phase Results . . . 143
5.6 Complexity Analysis . . . 147
5.7 Conclusions and Future Work . . . 150

6 Host-Based Multi-label Classification using Motifs 153
6.1 Related Work . . . 155
6.2 Fundamentals of Graphs and Motifs . . . 156
6.2.1 Basic Definitions and Concepts in Graphs . . . 156
6.2.2 Graph Modeling using Motifs . . . 162
6.2.2.1 Graph Randomization . . . 167
6.2.2.2 Motif Enumeration . . . 169
6.2.2.3 Motifs Selection . . . 171
6.3 Vertex Representation using Motifs . . . 173
6.4 Motif-based Traffic Classification: the Reference System . . . 176
6.4.1 Preprocessing: Network Graphs Construction . . . 177
6.4.2 Motif-based Parametrization . . . 181
6.4.3 Classification . . . 183
6.5 Reference System Results . . . 185
6.5.1 Experimental Results . . . 186
6.5.1.1 Preliminary Results . . . 186
6.5.1.2 Motif Enumeration and Selection Results . . . 187
6.5.1.3 Classification Results . . . 190
6.5.2 Analysis and Discussions . . . 192
6.6 Improved Motif-based Classification Method . . . 196
6.6.1 Evaluation of the Improvements . . . 198
6.6.2 Analysis and Discussions . . . 200
6.7 Extended Multi-label Motif-based Classification . . . 201
6.7.1 Classifier Target . . . 201

6.7.2 Multi-label Classification Input 203
6.7.3 Multi-label Classification Technique 204
6.7.4 Evaluation Metrics 208
6.7.5 Experimental Results 210
6.7.5.1 Multi-label Classification Results 210
6.7.6 Analysis and Discussions 212

7 Conclusions and Future Work 215
7.1 Conclusions 215
7.2 Future Work 218

Appendices 221

A1 The Customized DPI Tool 223

A2 Resumen 227
A2.1 Introducción 227
A2.2 Clasificación del tráfico de red 229
A2.2.1 Aproximaciones básicas 230
A2.2.2 Retos 231
A2.3 Objetivos y contribuciones 231
A2.4 Estado del arte en clasificación de tráfico 233
A2.5 Escenario experimental 234
A2.6 Optimización de las técnicas DPI mediante muestreo 235
A2.7 Clasificación de tráfico basada en el tamaño de los mensajes . . 236
A2.8 Clasificación basada en hosts mediante motifs 237

Acronyms 239

Bibliography 258

List of Figures

1.1 Evolution of Internet traffic [Brodkin, 2012] 2
1.2 Search and data download traffic for: a) BitTorrent and b) Gnutella 11

2.1 A multilevel taxonomy for traffic classification methods characterized on three levels 18
2.2 Generic diagram of machine learning based traffic classification 31
2.3 Examples of graph based traffic classification approaches: a) Graphlet, b) TAG, c) TDG, and d) Motif 41

3.1 Acquisition of datasets of real traffic for experimental purposes 56
3.2 Scheme and tools for reassembly, ground truth classification and parametrization of PCAP files 57
3.3 Monitoring point emplacement inside the network used for traffic capture 60
3.4 Comparison of the volume of traffic for the main traffic datasets 63
3.5 Protocol distribution in the main traffic datasets 68
3.6 Protocol distribution in the subsets: a) CS-A1 and CS-A2 (Chapter 4), b) CS-A3 and PS-1 (Chapter 5), c) CS-A4 and CS-B1 (Chapter 6) 69
3.7 Basic evaluation metrics for a binary classification scenario (classes A and B) 74

4.1 Analogy between DPI classification and table keyword search (table representation of a flow) 79
4.2 DPI packet classification model 83
4.3 Average detection percentage results, DP, as a function of the truncation length of the payloads (per-packet sampling) 90
4.4 Per-packet sampling detection percentage results for various protocol groups as a function of the truncation length for packets and flows: a) Web, b) Instant messaging, c) DNS and d) P2P . 91

4.5 Per-packet sampling detection percentage results for various protocols as a function of the packet truncation length: a) BitTorrent, b) eDonkey, c) Gnutella, d) Mail_IMAP, e) Mail_POP, f) Mail_SMTP, g) NTP, h) SNMP and i) STUN 92
4.6 Results for flow detection percentage as a function of the flow detection number 94
4.7 Flow detection percentage results for various protocol groups as a function of the flow detection number: a) Web, b) Instant messaging, c) DNS and d) P2P 95
4.8 Histogram showing the number of detected P2P and non-P2P flows as a function of the flow detection number for flow truncation 96
4.9 Average flow detection number for each individual protocol (per-flow sampling) 97
4.10 Percentage of detected flows for selected protocols as a function of the *flow detection number*: a) SIP, b) FTP, c) SSL, d) iMESH e) BitTorrent, and f) HTTP 98
4.11 Flow identification number (Flow_ID) v.s. flow detection number (HTTP deviators for $N_{min} > 30$) 99
4.12 Total inspection time t_i for dataset CS-A1 as a function of: a) the packet truncation length (per-packet sampling), b) the flow detection number(per-flow sampling), v.s. full payload 99
4.13 Detection percentage as a function of the inspected payload length using non-contiguous sampling for: a) P2P and b) HTTP 104
4.14 Detection percentage as a function of the inspected payload length using non-contiguous sampling for all protocols: a) three dimensional, b) two dimensional 105
4.15 Non-contiguous sampling: Payload size to be inspected per protocol for DP above 90% 106
4.16 Non-contiguous sampling: Results for various protocols as a function of the of the inspected payload length: a) BitTorrent,b) Gnutella,c) Mail_POP,d) Mail_SMTP, e) NTP, f) DNS, g) SSL, h) SSH and i) FTP 107
4.17 Histogram of DPI classification detection percentage for P2P protocols as a function of the inspected payload size using contiguous sampling 108
4.18 Results for various protocols as a function of the inspected payload length for contiguous sampling mode: a) BitTorrent, b) Gnutella, c) Mail_POP, d) Mail_SMTP, e) NTP, f) DNS, g) SSL, h) SSH and i) FTP 109
4.19 DPI classification detection percentage as a function of the inspected payload size using contiguous sampling, for all protocols (except for DNS): a) three dimensional and b) two dimensional 110

4.20 Contiguous sampling: Payload size to be inspected per protocol for DP above 90% 111
4.21 Total classification time, t_{fc}, for dataset CS-A1 as a function of the inspected payload size for: a) Method-I and b) Method-II, v.s. full payload 112

5.1 Diagram of the proposed system 130
5.2 Extracting application layer messages from a TCP session . . 132
5.3 Model topology for a single application. 132
5.4 Function used to normalize the message sizes (B=350). 135
5.5 Mean message size (normalized) for the clusters obtained for BitTorrent ($L = 6$). 141
5.6 PDFs associated with a) BitTorrent and b) HTTP for $l = 2$. . . 142
5.7 Sequences of application message sizes in both directions for sample BitTorrent, SSL and Gnutella flows. 143
5.8 Average classification recall for TCP and UDP applications. . . 145
5.9 Classification recall as a function of L: a) TCP b) UDP. 147
5.10 Precision and recall values for each application ($L = 3$): a) TCP, b) UDP. 149

6.1 A simple graph composed by six vertices and seven edges . . . 157
6.2 A labeled undirected graph composed by eight vertices and eleven edges 159
6.3 A labeled, directed and colored graph with 3 vertices colors, 8 vertices and 11 directed edges 159
6.4 Motifs of order: a) $k = 3$ and b) $k = 4$ 164
6.5 Schematic view of motif detection 167
6.6 Edge-switching process with regard for a colored directed graph for generating random networks 168
6.7 An example of a vertex profile using multiple motifs/graphs . 175
6.8 Traffic classification system based on motifs 177
6.9 Modules for motif-based traffic classification including training stages 178
6.10 Graphical representation of exchanges between two network hosts 179
6.11 Motif extraction from network traffic application graphs 182
6.12 Building the vertex profile for each host 182
6.13 KNN classification example 184
6.14 Application graph visualization using Graphviz for: a) BitTorrent and b) Gnutella 185
6.15 Ground truth results for hosts in the test dataset (CS-B1) . . 187
6.16 Accuracy average for different values of K 190
6.17 Accuracy results for the reference system, for the CS-B1 dataset trained on CS-A4 using K=3 193

6.18 The 24 selected significant and persistent motifs in CS-A4: order-3 (1 through 16) and order-4 (17 through 24) 199
6.19 Improved v.s. native method per protocol accuracy results for the CS-B1 trained on CS-A4 dataset 200
6.20 Building the vertex profile in: a) single-label mode vs. b) extended Multi-label 205
6.21 Per protocol accuracy for native, improved and multi-label models for CS-B1 dataset trained on CS-A4 211
6.22 P2P v.s. non-P2P classification accuracy for CS-B1 trained on CS-A4 212

A1.1 Flow diagram of the process and tools used to label and parameterize flows 224

List of Tables

2.1 Comparison of selected state-of-art methods 51
2.2 Features of future classifiers, defined at the three levels 52
2.3 Requirements for the best future traffic classification method, defined at three levels 53

3.1 Characteristics of the main traffic datasets 62
3.2 Characteristics of the traffic subsets used throughout the thesis 64
3.3 List of traffic parameters useful for blind classification 70
3.4 Confusion matrix for multi-classification scenario (classes A, B and C) 76

4.1 Confusion matrix for sampling-based DPI classification decisions 88
4.2 Comparison between best case scenario results for per-packet and per-flow sampling methods 100
4.3 Comparison between best case scenario results for sampling methods: Method-I and Method-II 111

5.1 Packet parameters obtained for the message analysis based method 140
5.2 Number of sample flows for applications in the CS-A dataset . 141
5.3 Confusion matrices for TCP applications for $L = 3$: a) Absolute number of samples, b) Percentage of samples relative to the total number of application samples (row) 146
5.4 Confusion matrices for User Datagram Protocol (UDP) applications for $L = 3$: a) Absolute number of samples, b) Percentage of samples relative to the total number of application samples (row) 148

6.1 Example of vertex profiling using a single motif attribute . . . 175
6.2 Number of selected hosts per application for training in the CS-A4 dataset 186
6.3 Fanmod-tool Settings and parameters for significant motif selection 188

6.4 Number of occurrences for each of the 36 significant motifs, obtained with the native method, across the 14 tested protocol (Training dataset CS-A4) 189
6.5 Examples of host vertex profiles (columns H_i) from the training dataset (CS-A4) 191
6.6 Confusion matrix for reference motif-based classification system (K=3, dataset CS-B1) 192
6.7 Main differences between the motif-based classification in single vs. multi-label modes 203
6.8 Protocol indexes and example of the use of the customized ML-KNN 207

Chapter 1

Introduction

1.1 Introduction

Today's networks are governing the way we live: network applications (e.g. social networks, e-learning, e-commerce, etc.) are changing the way in which social, educational and commercial interactions take place. Existing network applications are diverse and are ever evolving in order to continuously cover new users' requirements and provide new services. Ranging from legacy Internet applications, such as Web, e-mail, file transfer, etc., to dynamic complex applications as video streaming, Peer-to-Peer (P2P) file sharing, etc., the underlying network technologies must be adapted and/or improved to provide the new services. Thus, new network paradigms (e.g. overlay networks, Software Defined Networks, Internet of Things, etc.) are being developed and deployed.

As a result of the great impact and usage of Internet, the network traffic is constantly growing [Brodkin, 2012, Kende, 2012] and the network itself is becoming more and more complex. As shown in Figure 1.1, the available bandwidth for the final users has grown up to 2 orders of magnitude in the last decade while, at the same time, the number of users is also rising exponentially. From the network management perspective, this continued increase in the requirements and complexity represents a big challenge in many aspects. Among them, the rise in traffic may affect the Quality of Service (QoS). The influence of the traffic on the network infrastructure itself and on the hosted services is evident. If it is not properly managed, the network might become overwhelmed with huge amounts of uncontrolled traffic which would affect the overall network performance, cause congestion and even a disruption in the services. Thus, at some extents, services hosted by the network risk to become completely unresponsive or denied. On the other hand, as the volume of traffic increases, it will be more difficult to differentiate illegitimate traffic, which may represent a security problem.

To properly deliver application data between network endpoints, that is, to provide for the necessary QoS, the underlying network infrastructure and

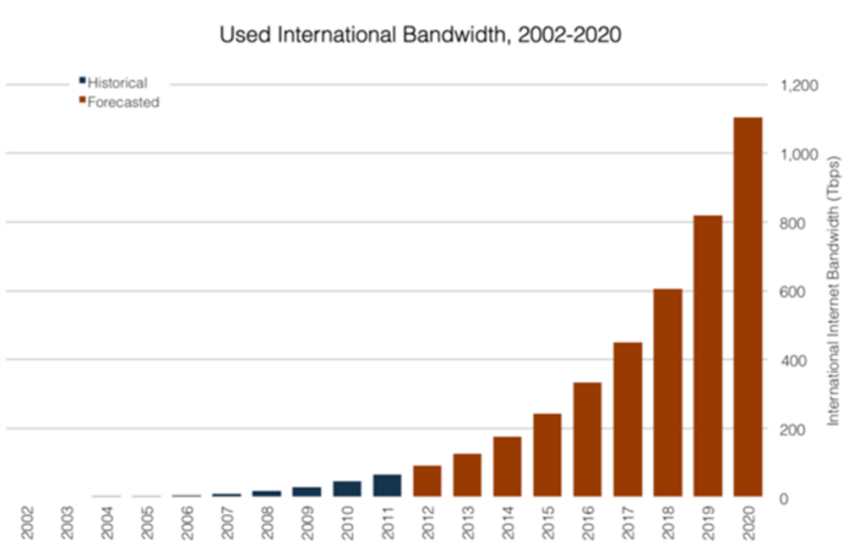


Figure 1.1: Evolution of Internet traffic [Brodkin, 2012]

interconnection devices should have a lot of attention at the management level. What the management and security solutions of today's network should cover depends on many factors related to the network type and services' criticality. However, a very common task for most of these solutions is to constantly monitor the traffic stream and demystify the nature of each traffic flow in real time, which is relevant both from security and performance standpoints. This way, it will be possible to provide each service with the required QoS or, if necessary, even to filter it from the network. In this sense, describing the network traffic by the amount of consumed bandwidth, though important, is still not enough. Thus, an important step in any traffic analysis is to identify what the user is really doing through each session, that is, which is the application in use.

Identifying network applications, that is, attributing network traffic to the application or kind of application generating it [Callado, 2009], is a corner stone in any of today's network management and security solutions. The differentiation of the nature of the flows in the network will be of major importance from a traffic engineering point of view. From this, it will be possible, for example, to provide different QoS to different services, to disable the traffic for a given application or to plan for network upgrading as a response to a new high consumption service. Furthermore, illegitimate traffic can use obfuscation techniques in an attempt to pass over usual policy enforcement systems such as firewalls. In this scenario, traffic identification is a must to apply the corrective actions needed.

In a naïve approach, the nature of the traffic can be attributed to an application or service based on the associated communication ports for the service [IANA, 2013]. In the early years of traffic classification, the port-based

technique [IANA, 2013] used to be typically the fastest, simplest and most accurate one. To classify flows and packets, it simply relies on Internet Assigned Numbers Authority (IANA) registered list (e.g. Hyper Text Transfer Protocol (HTTP) uses Transmission Control Protocol (TCP) port 80). However, the use of port obfuscation, address translation, port forwarding and protocol tunneling, together with the use of unregistered ports and multichannel applications, have deprecated the use of this technique. Therefore, more sophisticated techniques are nowadays required for this. Thus, traffic identification may involve the analysis of many traffic characteristics as the traffic payloads, the general properties of flows and packets, the anomalies in application protocols, the behavior of the applications and hosts or even the hosts' interactions. All of these are examples of various types of traffic characteristics that can assist network administrators in deeply understanding the effect of traffic associated to each application on the network functionality, security and performance. As an example, statistically huge sized packets and/or long duration flows might be indicators of large file transfer which may, in turn, lead to network overconsumption; a large number of connections might point to the use of P2P applications; etc. In a similar sense, malformed packets or incompletely established flows may point to protocol anomalies and suspicious activities, inquiring thus special intervention.

Consequently, identifying network applications is not solely driven by the simple need to associate traffic to applications. Though it is not sufficient for network management and security, it can assist in providing a deeper insight to understand the effect of different applications' streams on the network infrastructure. Most importantly, traffic identification is motivated by the direct security and performance implications one would have on the network behavior, which validates its relevance for network management and security solutions.

Literally speaking, the need to identify network applications is relevant for many network management and security tasks such as security measurements, QoS assurance, traffic shaping and traffic engineering. For example, as different applications have different network requirements in terms of packet latency and resource allocation, QoS assurance mechanisms strongly depend on application identification.

Tackling the intensive use of network resources among different applications is essential for bandwidth shaping and prioritization mechanisms used to optimize network resources which are common traffic engineering tasks used by Internet service providers and corporate networks.

Moreover, a preliminary step prior to enforcing any network access policy is to identify the application in use so that policy actions can be accordingly applied on network security devices. Namely, policies applied on firewalls, web filters, intrusion detection systems, etc. often involve conditions on the used application type, based on which, further actions are taken. For example, a web filtering security policy might consist on blocking P2P traffic; an intrusion detection policy might need to specifically inspect web traffic for known

attack types and so on. However, network based attacks that are hidden inside unknown traffic, are evidently problematic and harder to detect. From this standpoint, unknown traffic is regarded as one of the real sources of security threats which might be carrying malicious activities.

Many other reasons motivate traffic identification. Some applications, whose impact on the network is critical from network security and management perspectives, are regarded as crucial application types for traffic identification. The most relevant example is P2P (Section 1.3), for many reasons including their high bandwidth consumption, their associated system openness and security threats, and particularly, for the copyright concerns they raise.

In this context, the steady emergence of new Internet applications, together with the advances in obfuscation mechanisms used to bypass network controls, are keeping application identification a hot research topic. Identifying the wider scope of contemporary and newly emergent applications (e.g. P2P, mobile applications) is a challenge for most classifiers, as their accuracy figures may differ from one application to another. Even more, it is not only a question of new and possibly more complex applications with their associated protocols, but also the intentional use of techniques to hide the nature of the traffic, that is, of obfuscation techniques, which makes this problem a challenging one in nowadays networks.

After briefly introducing the traffic identification field, in the next section some basic definitions and terminology are presented, as well as a description of the basic approaches. Section 1.2 introduces a discussion regarding the need for a proper taxonomy and a benchmarking scenario in order to be able to properly compare and assess the multiple exiting techniques, which makes it necessary for every research team to use their own datasets for the development and testing of their proposals. Next, being considered one of the most challenging kind of applications for its classification, Section 1.3 presents a comprehensive overview of P2P traffic, and how it introduces new needs (P2P identification for security, copyright protection, high volumes of traffic) as well as new challenges (port obfuscation, distributed nature, traffic encryption and complex host interaction schemes). Section 1.4 describes the scope and the objectives of the current thesis and its contributions. Finally, Section 1.5 presents an overview of the thesis and the structure of the current document.

1.2 Network Traffic classification

1.2.1 Definition and Goals

As stated earlier, traffic classification [Callado, 2009] consists on attributing traffic instances or elements to the applications or kind of applications that generated them. According to the element to be classified, it can be made at three levels [Callado, 2009]: the flow level, the packet level and the host level. In the first case, the elements to be classified are the flows traveling across the

network, while in the second case these elements are the individual packets. On the contrary, host based classification consists in labeling each host by the application/s in use by that host. The most common case is flow classification, as opposite to packet or host based ones, as it is the most directly related to traffic shaping and QoS, and flows are the natural unit for communications between applications. Furthermore, they are more information rich than packets from a feature extraction standpoint. Anyway, host based classification, despite being a more complex problem, is also an interesting scenario for network planning and even security related issues, as it allows the identification of misbehaving hosts.

Traffic identification is a similar term also used in this context, although it refers to a slightly different target, that is, it is used when targeting at a more granular level. For instance, one flow can be classified as HTTP and identified as http get. Nevertheless, it is also usual to use both terms indifferently, which will be the case in the present thesis.

According to the number of classes to which traffic instances have to be associated, variants of the classification problem can be considered. Thus, in binary classification scenarios, the problem is deciding whether an instance to be classified belongs or not to a given category. On the contrary, in multiclass classification, an instance should be assigned to one of the multiple existing classes. Furthermore, there exist scenarios in which a multilabel classification is required. In these cases, typically related to host-based classification, each element to be classified has to be assigned to different classes. This is the hardest scenario from the classification point of view, as usually the number of classes (labels) to assign to each element are also unknown. In this regard, it is important to remark the differences between multiclass and multilabel classification. Obviously, in both approaches more than one class exist and, therefore, more than one different labels can be attributed to the objects to be classified. Nevertheless, the difference is that in the multiclass case, a classified object is to be annotated with just one class label among the different existing ones, while in the multilabel case, the annotation is done with a set of labels.

1.2.2 Basic Approaches

Although a detailed analysis of the state of the art in traffic classification will be presented in Chapter 2, next we present an overview of the existing traffic classification approaches and the major challenges.

In brief, the methods found in the literature analyze payload bytes [Erman, 2007a], packets [Sen, 2004a], flows [Zhang, 2013], or hosts [Karagiannis, 2005], being the flow-based classification the most used one. The scope of detected applications ranges from well-known protocols (e.g. HTTP) to challenging applications like P2P or encrypted traffic (e.g. Secure Sockets Layer (SSL) in [McCarthy, 2011]) and encrypted network tunnels [Mujtaba, 2009]).

On the other hand, most current existing approaches fall into one of two

categories: payload [Sen, 2004a] or non-payload (or in-the-dark) based classification. Historically, earlier classification methods relied on port numbers [IANA, 2013] by simply associating the application protocol type to the transport port protocol number. With port obfuscation, this method is obviously deprecated.

On the other hand, port-based classifiers will have an unrealistic view of the types of traffic being exchanged over the network, particularly for new emerging mobile applications [Choi, 2012], most of which rely on HTTP and Hypertext Transfer Protocol Secure (HTTPS). This renders port-based classification almost useless nowadays. Nevertheless, it can be still useful for legacy applications [Callado, 2009, Aceto, 2010] such as Domain Name System (DNS) or Simple Mail Transfer Protocol (SMTP) that use their default assigned port numbers, or for contexts where accuracy is not a major concern (e.g. traffic monitoring).

The historical evolution [Dainotti, 2012] of traffic classification methods shows that payload inspection [Sen, 2004a] has emerged after port number classification became unreliable. Payload based classification rely on payload inspection, being the most accurate methods currently found in the literature. They identify applications by matching well-known strings or signatures inside the packets' payloads, which raises, however, considerable concerns related to performance, encrypted traffic and user privacy. Thus, a lot of interest is deposited in non-payload based (also called blind) classification, as it is theoretically able to analyze encrypted traffic and it is not invading user's privacy. In this case, some features are extracted from the communications, being they flows or any other unit, without inspecting application layer payloads. Thus, each unit subject to analysis is represented through a set of parameters which is subsequently used as the input for the classification method of choice. In this regard, many techniques are described in the literature, including statistical [Yildirim, 2010], Machine Learning (ML) [Nguyen, 2008] and graphical [Iliofotou, 2007], besides hybrid [Keralapura, 2010] and a few miscellaneous [Trestian, 2010] approaches. All of these approaches will be detailed in Chapter 2.

1.2.3 Challenges

Traffic classification is a challenging task to accomplish. Many reasons make this a hard problem. Particularly, some applications that are intrinsically hard to detect might be using complex and dynamic protocols (e.g. videostreaming and P2P) and including advanced traffic obfuscation techniques [Zink, 2012] (e.g. port obfuscation, Network Address Translation (NAT), tunneling and encryption [McCarthy, 2011], etc.). Furthermore, as previously mentioned, transport layer port number can be easily obfuscated or NAT can be used to establish connections behind a firewall. Additionally, encryption makes it very hard to classify the traffic based on payload inspection mechanisms. And applications may even use closed proprietary protocols (e.g. Skype [Sen, 2004a]) to

disguise network control devices. Obviously, using closed proprietary protocols further complicates the task of traffic classification, as the protocol specificities and the message exchange patterns are undisclosed. Therefore, with the proliferation of new applications and the advances in obfuscation techniques, accurately classifying network traffic becomes more and more challenging.

Other considerations can be challenging for traffic classification research. Particularly, protecting the user privacy and coping with high link speeds, that is, scalability, are important concerns to be considered in modern classification systems. The user privacy concern prohibits the use of payload inspection techniques, despite of the high level of accuracy they provide, and is even illegal in many countries. On the other hand, coping with high link speeds has severe implications on the classifier design, which should be able to classify huge amounts of traffic in real-time without affecting the overall network performance.

On the other hand, although much research has been devoted in the last decade by the research community to the traffic classification problem, the current literature still lacks for recent surveys and for appropriate comparisons, despite the significant amount of papers in the field. Most existing classifiers lack for generality as they are often evaluated within special contexts (specific applications and/or network environment). Many different techniques were explored and deployed, as previously mentioned, ranging from simple statistical to advanced ML techniques. Besides inspecting the payload, quite different traffic characteristics were mined including packet, flow and host characteristics (packet size, flow duration, number of connections per host, etc.). Furthermore, the classification output, that is, the kind of the provided classification information, depends on the classifier design and target, some of them focused on packet or flow classification, while others are targeted at classifying hosts and host communities. In this context, with the diversity in the deployed techniques and data formats, the current literature still lacks for appropriate benchmarks relying on reference traffic captures and expressed in a consistent terminology. Furthermore, it is documented the lack of a consolidated input format, consistent method definitions, and common network application sets [Khalife, 2014]. The same applies for the definition of evaluation metrics and comparison procedures.

Moreover, most of the existing classification approaches need to be revisited for enhancements. Their intrinsic weaknesses are mostly related to the classifier's performance, the user privacy, the lack of generality, and even sometimes to inapplicability in real scenarios. As an example, while payload inspection methods are well-known and provide high classification accuracy, they are privacy invasive. Although relying on traffic characteristics instead of the payloads, complex ML techniques would arise performance questions related to the overall classification time, and to the required resources. Most often, low classification performance is related to the complex techniques and large traffic characteristics and input sizes used. Hybrid solutions were proposed in the

literature, however, a clear integration methodology that takes into account the capabilities of each classification technique and the overall system performance is still missing. Trade-offs among classification accuracy, user privacy and classification performance should be carefully studied. Apparently, there exists an ad hoc in the way methods are proposed, categorized and evaluated. As a result, an optimal traffic classification model has yet to be defined by the research community. In this regard, the preference of one method over another should be based on appropriate comparisons and benchmarks, which in turn requires a lot of missing steps and actions that has to be completed by the research community. In particular, it is relevant the lack of reference databases or frameworks for the proper assessment of the proposed techniques and systems. In this scenario, Chapter 2 will be devoted to a comprehensive review of the state of the art in which not only a taxonomy for current approaches is presented but also the challenges and requirements are detailed.

1.3 Classification of P2P Applications

As mentioned earlier, most existing blind classifiers lack for generality as they are often evaluated for a reduced set of specific applications. With the proliferation of many different novel applications, one of the questions that arises regarding the evaluation of the classification methods is the preferred set of reference applications on which classifiers should be tested and evaluated. This choice reveals relevant for the research community in order to allow comparisons and benchmarks to converge efforts toward selecting the optimal classification model. In the current literature, classifying applications which are intrinsically hard to detect, such as P2P applications [Shen, 2010] (e.g. BitTorrent [Bittorrent, 2013], Gnutella [Gnutella, 2013]) are considered a key indicator of the classifier's capabilities [Nguyen, 2008, Zhang, 2009]. In fact, P2P applications refer to traffic obfuscation techniques, encryption and tunneling and often use closed proprietary protocols to disguise most network control devices. As such, P2P application identification has been cited as part of the core classifier evaluation process in most of the traffic classification works we surveyed. Assessment results in most of these papers usually point to P2P classification together with the overall classification results. Following this approach, the subsequent experimental chapters addressing classifier's enhancements and proposals focus on P2P classification results. To get a deeper understanding of what makes P2P applications highlighted by most traffic identification works, in the following, we explain P2P main characteristics and their implications in traffic identification.

Understanding an application's protocol and its associate architectural components can be significant for the application identification task, as the classification algorithm should be able to detect each application specificities and traffic patterns that are discernible from other application types. In the fol-

lowing sections, we try to reveal some of the P2P discriminative features that are relevant for traffic identification. We start first by highlighting on the particular benefits and challenges for P2P identification.

1.3.1 Relevance of P2P Traffic Identification

Nowadays, P2P applications are commonly used over the Internet and cover various user needs, from file sharing or Voice Over IP (VoIP), to media streaming and Peer-to-Peer TV (P2PTV) applications. Identifying P2P applications is particularly relevant for network security and management, due to many reasons. On the one hand, tackling the intensive use of network resources associated to P2P applications, which supposedly constitute a substantial proportion of today's Internet traffic, commonly represents a challenge for Internet Service Provider (ISP)s. Although P2P usage for sharing purposes is somehow decreasing, P2P applications are still considered on the top of bandwidth consuming applications [Bittorrent, 2013]. Additionally, the traffic symmetry characterizing some P2P applications is considered today by some ISPs an emerging problem for which their networks are not prepared for, as they were designed under the assumption of traffic asymmetry. On the other hand, P2P generates security threats that are mainly due to the openness and distributed nature of the system, making them vehicles for the exploitation of several security flaws and facilitating the spread of worms and viruses. Furthermore, they are becoming increasingly used as the infrastructure backing up some botnets. These facts are particularly problematic in the P2P case, since the traffic they generate is now intentionally disguised to avoid monitoring systems. Thus, the shared contents and transferred files are even harder to control, effectively decreasing the security level of a host or even of the entire network. Finally, files can be downloaded with no respect to any copyright or legal issues using P2P.

As stated previously, P2P systems are characterized by their openness, high decentralization and dynamicity. P2P applications use multiple obfuscation techniques [Zink, 2012] as payload, flow and port obfuscation, tunneling, and encryption [Mujtaba, 2009]. Moreover, P2P applications are still relatively recent and do not conform to any standards or rules in the sense that closed proprietary P2P protocols can be mutated and new protocols can appear, which further complicates the traffic tracking. For instance, some P2P applications, like Skype, use encrypted traffic through a closed and proprietary algorithm, which turns its analysis and detection harder to perform. For these reasons, P2P applications are intrinsically hard to detect and are thus considered, as mentioned earlier, a key indicator in evaluating the capabilities of the traffic classifiers.

1.3.2 P2P Architectural Model and Discriminative Characteristics

In contrast to the most widely adopted model by Internet applications, i.e. the client-server model, there are no fixed client or server roles in the P2P model, as any host can act as client or server in any exchange of information. Any host in a P2P network can directly exchange information with any other one, after finding the location of the desired piece of information and the address of the information holder. Therefore, one of the core problems in a P2P network is the addressing and search of the information, for which various approaches are considered. Thus, P2P applications may be built on structured or unstructured overlays [Shen, 2010], following a centralized, a decentralized or a hybrid model. Although structured overlays are developed to improve the efficiency of data lookup, most P2P Internet applications rely on unstructured overlays. In the unstructured P2P overlays, the search mechanism is performed through central servers (centralized P2P systems), or through message propagation strategies (decentralized P2P systems), while in hybrid P2P systems, a few nodes are selected to act as search servers (called super-nodes, ultra-peers or trackers) based on their computing capabilities.

Although various P2P applications exist and merit being analyzed, for the purposes of this work, we will be more concerned with unstructured P2P file sharing applications, mostly Gnutella and BitTorrent, as they are the most frequent P2P applications in nowadays traffic. Gnutella [Gnutella, 2013] is a P2P network available in both decentralized and hybrid versions, while BitTorrent [Bittorrent, 2013] uses a search mechanism closer to hybrid systems. According to Cache Logic [Schulze, 2009] BitTorrent represents up to 43% of Internet traffic.

The way in which P2P communications take place among peers and super-nodes and the nature of P2P traffic will have direct implications on the connection patterns and traffic characteristics, as seen on the host, host-community and even on the flow and packet levels. For example, a key concept for P2P systems is the decentralization [Shen, 2010]. In most P2P systems, the decentralization concerns only the signaling traffic, and not the download traffic, which directly occurs between peers. Thus, as shown in Figure 1.2, a requesting peer first searches a file through a specific tracker or super-node, or by propagating a search request to other peers. The requesting peer then connects to other peers holding different file chunks to start the download. This behavior is relevant from a traffic identification standpoint: in order to perform tasks such as answering content queries and sharing files, the P2P peers act both as a client and as a server opening a large number of connections. Fortunately, with different patterns in signaling and data traffic, P2P hosts reveal a lot of other discriminative traffic characteristics that are relevant for traffic classification (e.g. long flow duration, packet size and interarrival times distribution, etc.) as will be detailed in Chapter 2.

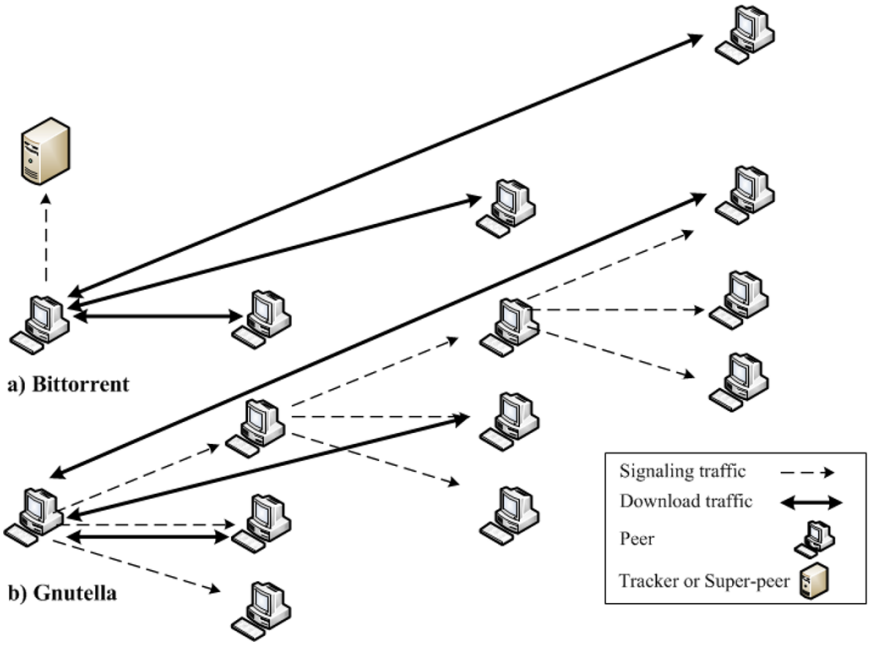


Figure 1.2: Search and data download traffic for: a) BitTorrent and b) Gnutella

1.4 Thesis Scope and Contributions

The main objective of the present work is to develop and evaluate novel methods for blind traffic classification both at the flow and host levels. For this, it is also necessary to set up an experimental framework including real traffic captures that should be labeled according to their classes and that will be used as the "ground truth" to train and evaluate the classifiers. Additionally, it is necessary to develop the programs and algorithms required to parameterize the traffic, according to the needs of the proposed classification methods.

The systems explored for traffic classification are based on message size analysis, for the flow level one, and on elementary graphs decomposition, using the so called motifs, for the host based classification. Prior to any of these proposals, an in-depth review of the state of the art and a proposal on a taxonomy is made. Additionally, during the process of classifying the acquired traffic using a customized Deep Packet Inspection (DPI) tool, some optimizations based on sampling are proposed and evaluated. Thus, the contributions of this thesis can be summarized as follows:

A) New multilevel taxonomy proposal: The number of papers in the literature covering traffic classification is tremendous. However, few are the number of surveys covering up-to-date papers and fewer are the ones that systematically categorize existing works. With the lack of existing

surveys and significant taxonomies, we provide a survey categorizing existing methods according to a proposed taxonomy at three different levels (input, technique and output). As a result of the study of the state of the art, the main challenges and the characteristics of the ideal traffic classifier are analyzed.

B) Framework for traffic classification research: Evaluating and comparing traffic classification methods should rely on significant traffic traces. With the lack of reference datasets, we collected real traffic sets in a significant volume after analyzing and discussing the requirements for a proper assessment. Additionally, a set of tools to handle the datasets and to extract all the parameters described in the literature for its potential use in the classification process is built. Moreover, a DPI classification tool is customized for its use in the assessment process, that is, to build the ground truth and compare the results.

C) Assessment and optimization of payload-based classification methods: The major limiting factors for the usage of DPI classification are breaching the user's privacy and the associated computational burden. After assessing the performance of DPI, we present an optimization based on various sampling policies for the payloads in each flow. The best trade-off for performance, accuracy and user privacy is experimentally demonstrated.

D) Novel blind flow-based classification method: The messages exchanged by peer entities at the application layer hold discriminative information that could enrich the classifier's capabilities. From this standpoint, we propose a novel blind method to explore the discriminative power of application layer messages. The novelty of our approach consists on relying on the size and direction of the messages instead of on the packets and on applying a Markov modelling together with multipeak Gaussians to represent the possible distributions in size from the different methods in each application layer protocol. The results evidences the goodness of the proposal and the existence of discriminative information regarding traffic classification in the sizes of the exchanged messages.

E) Assessment and improvement of blind host-based classification models: The current host-based classification models do not reflect real usage scenarios, where a host might be using more than one application. Thus, after selecting a host-based method, we explore multilabel classification. The method of choice is based on a graph description of the interactions between hosts, which are represented as elementary units, the motifs. This method is assessed and some improvements at the parametrization level are proposed and evaluated. Finally, we explore the extension of the method for multilabel host identification.

1.5 Thesis Structure

According to the previous discussions and contributions, the present thesis is structured in 6 additional chapters as follows.

First, in Chapter 2, we analyze the state of the art in a comprehensive way after proposing a taxonomy, which is consequently used to describe and organize the existing contributions. While in the last decade multiple research groups have been working in the traffic classification topic, which resulted in a huge amount of publications, no significant survey on the field existed. Thus, most of them were outdated while others were focused on one specific research trend. Therefore, this chapter aims to provide a systematic approach for categorizing and characterizing traffic identification methods through a comprehensive three-level hierarchical taxonomy. After discussing the taxonomy, we present the current literature in traffic classification by surveying most of the existing and recent papers in the field, which are described according to the terminology and categories established in the proposed taxonomy. From this, the requirements that should be met by the optimal model or method for traffic identification are discussed and the underlying related research hot topics are highlighted.

Then, before moving to each of the new proposals requiring experimental validation, we need to establish an appropriate testbed. Therefore, Chapter 3 is dedicated to explain the experimental setup and the testbed used throughout the remaining of this thesis. The chapter starts with a discussion regarding the requirements for the datasets and the scenario in order to be able to properly evaluate the methods. From this analysis, it is concluded that, among other properties, a proper dataset of real and labeled traffic, the so called ground truth, is needed. This represents a relevant challenge, as it is not trivial to label each of the instances in the real dataset. Therefore, after describing the considered networking scenario and the data acquisition procedures, the method and the tool used to label each of the samples are presented. The chapter continues with a description of the captured datasets and the different partitions established to train, evaluate and validate each of the traffic classification methods proposed in this thesis. Next, the feature extraction procedures and parameters are described. The chapter concludes with a presentation of the different metrics required to evaluate the performance of the traffic classifiers.

Once the testbeds are set, in Chapter 4 an assessment and optimization of payload-based classification is described. Throughout this chapter, we propose and evaluate a general methodology for optimizing DPI through sampling. The proposal is motivated by the potential to decrease the computational cost of the inspection of the payloads and to enhance the users' privacy by reducing the amount of analyzed payload bytes. Thus, various sampling schemes including payload-based sampling, flow-based sampling and a combination of both techniques, using different inspection approaches, are considered and experi-

mentally assessed. The classification results are then compared to show the gain that can be obtained with the proposed samplers. An analysis of the positions in the flows of the information which is relevant for the discrimination of the applications is also addressed.

Then, Chapter 5 presents and assesses a novel proposal for a blind flow-based classification method. After finding in Chapter 4 that most of the signatures used by DPI are located within the initial flow packets, the proposed method focuses on the first application layer messages exchanged in each flow. Thus, the procedures to parameterize each flow through the sizes and relative directions of these initial messages are described. Next, the basic modeling used to classify each parameter vector, that is, each flow, is presented. It is a classifier based on Markov models whose observation probabilities are obtained from class-dependent multipeak Gaussian distributions that are estimated through a training procedure. Therefore, the topology and elements of the models, as well as the method used to obtain these distributions are then described. Finally, before the experimental validation of the proposed method, the procedures to estimate the parameters of the models, that is, the Probability Density Functions (PDF)s, are described.

After proposing a novel flow classification method, Chapter 6 focuses on host-based classification. In particular, it is devoted to the assessment of a graph-based approach based on the concept of motifs. This approach was previously proposed and explored by other authors, although it presented some limitations. Therefore, in Chapter 6 we describe the fundamentals of the underlying graph theory and the extraction of the so-called motifs from a graph representing the interactions between hosts. Each host is then represented by a profile which accounts for the involvement of the host in each of the obtained motifs. Then, we assess and evaluate the results obtained from this method in a real scenario in order to identify the limitations of the technique. In this sense, some improvements targeted at the feature extraction and parametrization are proposed and evaluated. Next, in order to be really applicable in real scenarios, the classifier should be able to classify the hosts according to the set of applications in use instead of just one of them. Evidently, a host profile should depict the typical host behavior contributing in more than one application at the same time, which is rarely addressed in the literature and was not considered in the previous work. Thus, we extend this method to a multi-label host classification scenario.

Finally, Chapter 7 presents the conclusions from this work and some ideas to be explored in future work.

Chapter 2

State of the Art in Traffic Identification Methods

The basics of traffic classification and the motivations for this work were described in Chapter 1. This chapter is devoted to present and analyze the works described in the literature for this topic through a multi-level taxonomy.

Moreover, based on the surveyed works and comparisons, this chapter outlines the main requirements for future benchmarks and the optimal classifier design concept that we believe (and argument) to follow a multi-classifier approach.

In the last decade, the relevance of the traffic classification problem motivated multiple research groups, which resulted in a huge number of publications. Although each of the identification methods described in the literature is considered optimal or valuable from the perspective of the authors, in most of these works, only specific conditions and/or particular applications were considered. Furthermore, existing comparisons used to cover only few identification techniques, while the results were rarely verifiable and the main focus was on reducing error rates. Apparently, there are no clear research trends toward defining the best traffic identification model.

On the other hand, existing surveys, [Zhang, 2009, Callado, 2009]), are now outdated since the number of contributions has doubled in the last few years, including a lot of recent and promising approaches. Some of these surveys focused on specific research trends (e.g. [Nguyen, 2008] for machine learning methods). Moreover, a single criterion, the classification input or technique [Zhang, 2009, Callado, 2009] is usually considered for categorizing methods described in the literature. As a result, identification methods were often mapped to non-disjoint categories. To add to this fact is that most of the proposed methods rely on different techniques and on various input/output formats.

However, comparing methods without considering their differences both at the input and output levels would be of less significance.

Consequently, a comprehensive survey that can systematically categorize and characterize different aspects of existing identification methods is required. Obviously, a systematic taxonomy of exiting works is also relevant from a benchmarking perspective.

In this chapter, a survey of existing and recent achievements in traffic classification over the last years is presented, focusing on the methods that are the most relevant to the contributions in this thesis.

The aim is to provide a systematic way in categorizing and characterizing traffic identification methods. Therefore, a proposal for classifying and organizing traffic identification methods is contributed in the first part of this chapter. This taxonomy is then used as the vehicle to present major contributions in the field. Specifically, a comprehensive three-level hierarchical and systematic taxonomy which can assist in defining one consistent terminology that is useful for future benchmarks is proposed. According to this taxonomy, each method is categorized at three different aspect levels: the input type, the technique used and the output. According to this taxonomy, methods are grouped into disjoint category groups.

Finally, based on the surveyed works and comparisons, the optimal classifier design together with the challenges and future requirements to build such a system are discussed. As will be argued, it should follow a multi-classifier approach.

2.1 A Novel Multilevel Taxonomy for Traffic Classification Methods

This section is dedicated to presenting the proposed taxonomy that we follow in categorizing literature works in traffic identification.

As previously mentioned, hundreds of papers have been devoted by the research community in developing traffic classification techniques. However, some of the existing surveys take a narrow view [Nguyen, 2008] while most [Zhang, 2009, Callado, 2009] are currently outdated. Moreover, previous taxonomies focus mainly on the used technique (e.g. port-based, DPI, machine-learning) as the basic categorization criteria. Thus, a lot of relevant features that worth of being underlined at the input and output levels were missed.

On the other hand, rigorous benchmarks usually refer to compare methods within the same group before comparing methods across different groups. From this perspective, existing taxonomies are confusing since the same method may fall into different category groups. Therefore, comparing different categories as per the existing taxonomies might be of less significance.

For these reasons, and to surpass these limitations, a comprehensive and multilevel taxonomy that covers most existing and recent traffic identification works in the literature is proposed and presented in this section.

According to this taxonomy, each identification method is characterized at three different levels:

(i) The classification input: The input covers the analyzed traffic characteristics, which can be measured at different levels (e.g. packet, flow, host, etc.).

(ii) The classification technique: The technique describes the core of the identification process which may involve numerous means (e.g. payload inspection, statistical, behavioral, etc.).

(iii) The classification output: The output is described in terms of quantitative metrics (e.g. accuracy, precision, recall, etc.) as detailed in Chapter 3, that targets objects at different levels (e.g. packet, flow, host, etc.).

This three-level grouping provides richer information about each method while generating separate three-tuples categories. Figure 2.1 illustrates the proposed multilevel taxonomy by showing category groups at each of the three defined levels. The different categories in each level are defined after the review and grouping of the systems proposed in the literature, as will be described in the next sections.

In this taxonomy, an identification method is necessarily a member of at least three groups, one at each level. Nevertheless, existing identification methods may rely on multiple choices or on new non-categorized ones at each level, referred to, respectively as combination and miscellaneous groups in Figure 2.1. Thus, an identification method is described as belonging to a 3-tuple defined category (input, technique, output) that is associated with one distinctive path in the diagram of Figure 2.1. For instance, the classification methods proposed in [Sen, 2004a] and [Khalife, 2013b] are described by (payload, payload inspection, flow-accuracy).

However, it is important to note that choices at the three levels are not totally independent, as the classification technique may imply the form of the required input and the provided output. For example, the payload input would most probably imply the use of a payload inspection technique, providing flow accuracy at the classifier's output.

Nevertheless, with such a multilevel taxonomy, comparisons should become of higher significance once the 3-tuples category groups are appropriately chosen. For example, it would be much more significant to compare methods having the same output group (e.g. host-based [Karagiannis, 2005, Allan, 2009]), rather than comparing methods of different output groups (e.g. host-based [Allan, 2009] and flow-based [Sen, 2004a]).

Next, from the perspective of this taxonomy, existing methods will be described. However, instead of reviewing hundreds of individual methods, representative methods are chosen from each category.

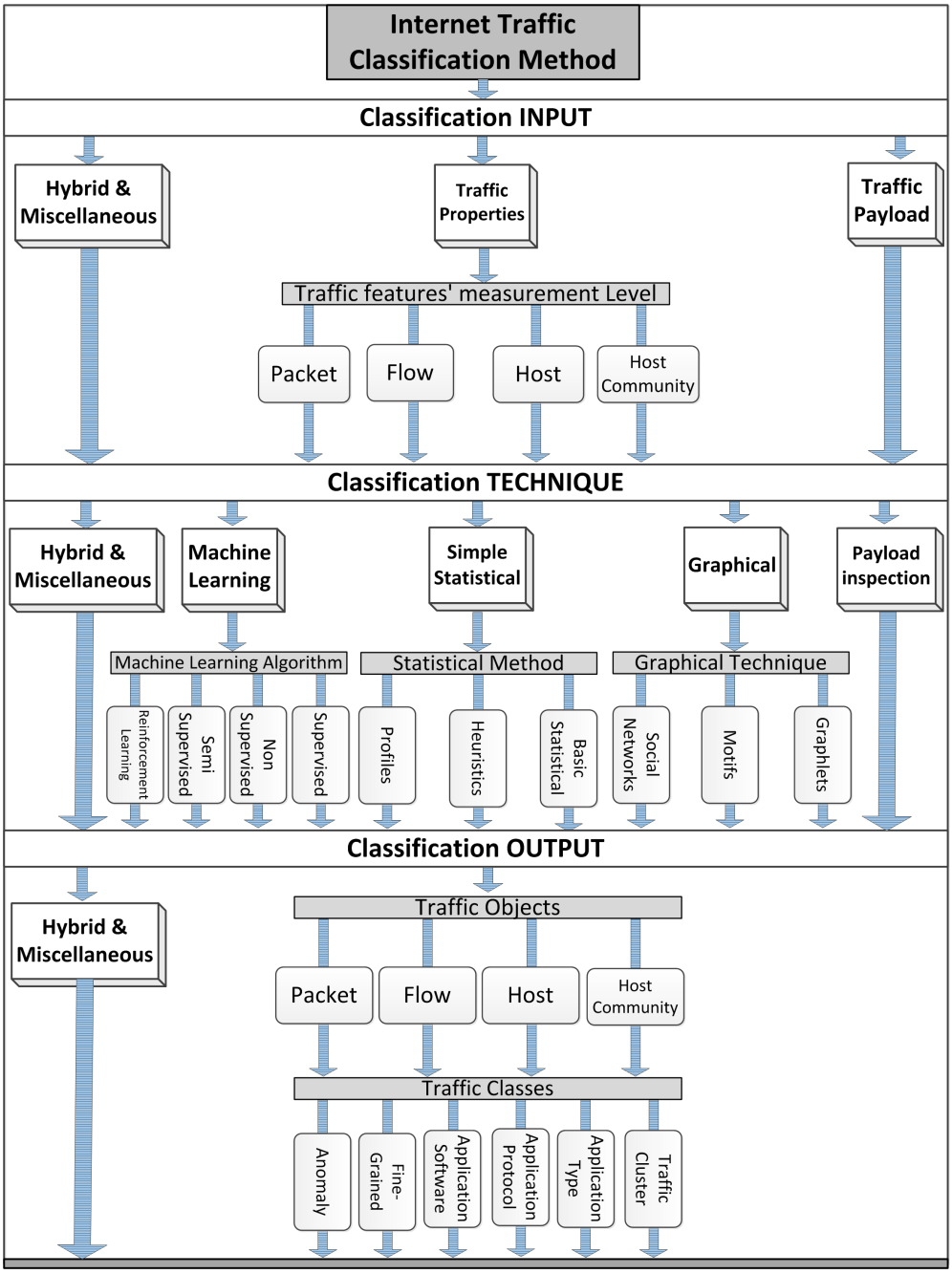


Figure 2.1: A multilevel taxonomy for traffic classification methods characterized on three levels

In order to highlight on the motivations behind traffic identification, we start by examining the kind of output level provided by each method. Choices at the technique and input levels will be subsequently discussed.

2.1.1 Categorization by the Classification's Output

In the following, a survey and categorization of different traffic classification methods is presented according to the nature of the classification output, as per the third level in Figure 2.1.

Basically, as shown in Figure 2.1, the classifier's target is to associate traffic objects to output classes. Thus, each of the considered traffic identification methods should define the traffic class or classes to which target objects are to be ascribed. Thereafter, the goodness of the classification process is evaluated in terms of some metrics which are usually dependent on the task, that is, the nature of the input and output classes and the target of the classification.

2.1.1.1 Traffic Objects

Traffic objects, as shown in Figure 2.1, range from fine-grained (bytes, packets) to macroscopic levels (hosts, host-communities). Namely, traffic objects include:

(i) Packets, in this case, each individual packet is to be labeled (e.g. [Sen, 2004a]).

(ii) Flows, where each flow is considered for classification. Usually, flow based classification (e.g. [Zander, 2005, Kim, 2011]), defines a flow as an unidirectional or bidirectional series of Internet Protocol (IP) packets having the same IP addresses, port numbers, and transport layer protocol. The flow concept can be extended to that of a Bag of Flows (BoF), (e.g. [Zhang, 2012, Zhang, 2013]), which consists on a correlated set of flows that are very likely to be generated by the same application.

(iii) Hosts, where the target is to classify the activity of a single host, (e.g. [Karagiannis, 2005, Allan, 2009, Keralapura, 2010, Jinsong, 2009]). In host based classification, hosts are represented by IP addresses and associated, in most of the cases, with one single application (e.g. DNS host, file transferring host, etc.).

(iv) Host-communities, in this case, it is the joint activity of a set of hosts which is to be classified (e.g. [Iliofotou, 2011]). For example, host-communities can be classified as P2P host-communities, SMTP host-communities, etc.

Although some classification papers describe bytes as the elements to analyze [Erman, 2007b, Erman, 2007a], this approach is usually adopted only for statistical evaluation, (e.g. monitoring volume consumption). They are not literally defined, nor can be targeted as traffic classification objects.

Contrarily to bytes or packets, flows are the top most adopted identification objects in most of the surveyed works. However, deciding upon the

preference of classification objects still depend on the classifier's intended use. For example, for VoIP signaling traffic, both high flow and packet accuracy are recommended, which is critical for controlling the whole voice session.

Ideally, a classifier should be able to identify traffic at the smaller possible granular level, which is supposed to be computationally intensive. However, coarse-grained identification objects are potentially more robust against network fluctuations that usually affect low level statistics [Allan, 2009, Jin, 2009].

On the other hand, except for few modeling works (e.g. [Chang, 2009]), most models treat each application independently to simplify the host classification problem. Most of the existing host and host-community classification methods assume that only one application is in use by each host. However, one host would simultaneously contribute in many applications, as per the normal user behavior that might be, for instance, surfing the web and making a VoIP phone call while running a P2P program in the background. Consequently, inferring fine-grained classification given coarse-grained annotations, and vice versa, is not a straightforward process. In other words, applications associated with packets and flows generated by one host cannot be implicitly inferred given that host's annotation.

For example, flows generated by a host classified as a P2P host, are not necessarily P2P flows. For this reason, coarse-grained identification approaches are used to assist or to validate fine-grained classification methods. For instance, after payload inspection fails at the packet level, [Keralapura, 2010] resorts to host-based detection.

2.1.1.2 Traffic Classes

According to the degree of the classification detail, traffic classes can be categorized (see Figure 2.1) as ranging from fine-grained (e.g. a web application function) to wider macroscopic levels (e.g. traffic cluster). As described in the literature [Park, 2011], up to six major schemes can be identified for traffic classes:

(i) Traffic clustering, (e.g. [McGregor, 2004]), which defines broad classes of traffic (e.g. bulk, small transactions, etc.).

(ii) Application-type, (e.g. [Trestian, 2010]), which classifies the traffic according to the kind of application generating it (e.g. game, browsing, chat, P2P, streaming, email, etc.).

(iii) Application protocol, (e.g. [Keralapura, 2010]), which targets at the protocol associated to the traffic (e.g. Post Office Protocol (POP3), SMTP, etc.).

(iv) Application software, which tries to determine the specific program for a given piece of traffic (e.g. specific FTP or BitTorrent client software).

(v) Fine-grained traffic classification, as in [Park, 2011, Kim, 2012], for multi-channel applications, which takes a deeper insight trying to distinguish different constituents in the traffic from each application (e.g. Facebook login, Yahoo search, etc.).

(vi) Anomaly, (e.g. [Kim, 2006]), targeted at simply discriminating between normal and abnormal traffic.

Anomaly is one of the main classification targets of many monitoring security systems. Anomaly alarms are generated whenever a deviation from a pre-estimated "normal" behavior is detected (see *profiles* in Section 2.1.3.2). Traffic anomalies can include network, transport and application layer anomalies.

On the other hand, fine-grained traffic classification can be relevant for multi-channel applications that open multiple connections for different purposes. For instance, Skype [Bonfiglio, 2007] offers several services (user authentication, voice and video communications, file transfer, chat, etc.) over TCP and UDP connections. Although belonging to Skype at the software application level, these flows should be identified differently at a finer-grained classification level. Furthermore, fine-grained traffic classification reveals crucial for mobile applications [Park, 2011] as most of them rely on HTTP and HTTPS flows.

Another relevant question is related to the number of classes each traffic object can be assigned to. As previously mentioned, many of the contributions in the literature use a single label to classify the traffic (single-label classification), that is, each traffic object is attributed to just one class from one of the previously mentioned classification schemes. This can be a suitable approach depending on the nature of the traffic objects to be classified, as is the case for flows and packets, and the intended task. However, there exist some situations for which it would be wiser to have different labels from various schemes (or multi-label classification) working at different granularities for each traffic object [Schulze, 2009], and even from the same scheme, if the traffic objects are hosts or host communities.

Regarding this last situation, most contributions in the literature use a single label for hosts or host communities based classification, as they use the assumption, most of the times not explicitly stated by the authors, that a host is involved in only one application at a time.

As a direct result of this limitation, realistic cases where more than one application is simultaneously in use are obviously not considered. As an example, the authors in [Allan, 2009, Allan, 2008] relied on graph mining and used a single-label K-nearest Neighbors (KNN) algorithm to classify the host activity, testing hosts that used one application at a time, to find that most of the considered hosts (61%) properly matched the profiles of a single class, even in a narrow time window.

2.1.2 Categorization by the Input

In the following, a survey and categorization of different traffic classification methods is presented according to the nature of the input to the system, as per the first level in Figure 2.1.

In general terms (Figure 2.1), traffic identification methods may use data directly taken from the observations, as the traffic payload; processed information from traffic measurements and properties, as packet sizes; hybrid sets from both kinds and some miscellaneous types. The two main categories are:

A) Traffic payload: As mentioned in Chapter 1, one of the most used and successful methods nowadays, DPI, is based on the inspection of the payloads. Anyway, many other classification techniques also use the payloads of the packets as their inputs. In this context, packet payload is referred to in this thesis as the whole content of the packet beyond network and transport headers, or a part of it, that is, just the information from the application layer.

Despite their successfulness, major concerns are raised regarding the user privacy with payload based methods, as mentioned in [Sen, 2004a, nDPI, 2013, Lu, 2007, Yeganeh, 2012]. Though most DPI signatures require few packets to be disclosed [Khalife, 2013b, Aceto, 2010], payload based classification is still less preferred due to privacy breaching and the supposed computational burden it introduces.

On the other hand, the use of ciphering may make this kind of input as useless for traffic identification.

B) Traffic properties: Alternatively, non-payload based identification (also called "in-the-dark" classification) refers to extracting various traffic features based on information at the network and transport layers. These features can be gathered at different levels as packet, flow, host, and host-community (Figure 2.1). They are similar, although not necessarily the same, to the type of the traffic identification object (Section 2.1.1.1).

The nature and kind of the extracted attributes depends on the level being considered. Thus, without pretending to be exhaustive, they can be:

(i) Packet-level features: most common parameters are the headers from the packet at different layers (network and transport), packets' sizes, flags, etc.

(ii) Flow-level features: as the flow size, flow duration, average number of packets, interarrival times, etc.

(iii) Host-level features: as the number of connections, the number of distinct active ports, etc.

(iv) Host-community-level features: as the connection degree, graph diameter, etc.

Throughout most of the surveyed works, e.g. [Dainotti, 2008, Hu, 2012], the packet interarrival times and packet sizes (including some statistical values as the maximum, minimum, mean, variance, etc.) were mostly used in a flow level characterization.

Obviously, the nature of the input depends on the network being monitored and the available tools and resources. For instance, bidirectional packets' features might not be applicable in multi-homed networks relying on a single monitoring point.

From a different standpoint, measurements can be either passive or active. Passive measurements are carried out by parsing traffic characteristics, e.g. [Moore, 2005a], while active measurements, e.g. [Trestian, 2010, Dedinski, 2009], inject probe packets into the network to infer some of its properties. Most of the surveyed works use passive measurements inquiring, in some cases, the disclosure of packet payload.

A common and relevant practice regarding the input to the classifiers is the attempt to reduce the size of the input data in order to improve the performance. For this purpose, different techniques are described in the literature, although the applicable ones might depend on the nature of the data. Thus, the number of parameters in input vectors can be reduced through feature selection, as in [Nguyen, 2008]; flow data can be replaced by that from BoF [Zhang, 2013, Zhang, 2012]; etc. This kind of approaches can be considered a part of the classification method or a preprocessing step, and will be addressed, as needed, in the next section.

Nevertheless, there exist some approaches that use just a fraction of the potentially available data, independently on the used classification method and the category of the input. Two schemes are relevant in this context: early classification [Bernaille, 2007, Tabatabaei, 2012, Wang, 2009b] and sampling [Khalife, 2013b].

Early classification methods are targeted at classifying flows as soon as possible. For this, they only analyze the first few packets of the flow. The intuition behind the approach is that these packets are supposed to capture a pre-defined sequence of messages within the application negotiation phase. Similarly, the input can be simply reduced through different sampling modes. However, methods relying on much reduced input size should provide alternative sources when their input are lost or not captured by the classifier.

2.1.3 Categorization by the Technique

In this subsection, a survey and categorization of different traffic classification methods is presented according to the type of the classification technique used, as per the second level in Figure 2.1.

According to this criterion, existing methods can be categorized as belonging to the following major categories:

(i) Payload inspection: based on the direct inspection of the whole or part of the data in the application layer payloads of the packets [Sen, 2004a, Yeganeh, 2012].

(ii) Simple statistical: which uses general traffic features to build representative sets of parameters which are almost straight forwardly compared with the reference sets of parameters [Moore, 2005a, Crotti, 2007]. Simple statistical approaches include basic statistics [Yildirim, 2010, Alshammari, 2011], heuristics [Karagiannis, 2005] and behavioral techniques [Trestian, 2010]. In these cases, representatives for each of the classes to be classified are manually selected or set and some distance or distance-like measures are used to find the class of an element.

(iii) Machine learning: this category includes a variety of methods and systems from the Pattern Recognition discipline to classify the observations (e.g. [Dehghani, 2010, McGregor, 2004]).

(iv) Graphical: based on building graphs representing some interactions between the elements to be classified, which are subsequently analyzed using graph-based techniques. This set of techniques is mainly targeted at the host and host-community based classification (e.g. [Karagiannis, 2005, Iliofotou, 2007]).

(iv) Miscellaneous and hybrid: Hybrid approaches (Figure 2.1) integrating more than one of the previously stated techniques were proposed in the literature [Keralapura, 2010, Zhenxiang, 2011]. Moreover, various miscellaneous methods [Trestian, 2010, Zhang, 2011] can also be found in the literature and will be discussed in the next sections.

The historical evolution of traffic classification methods shows that payload inspection has emerged after port-based classification became unreliable [Dainotti, 2012]. Then, statistical and ML techniques were applied to overcome the limitations of most previous methods, while graph-based approaches are still in its initial stages of development.

Once the categories have been presented, they will be described with greater detail in the next following sections, including the respective subcategories and the most representative contributions associated to each one.

2.1.3.1 Payload Inspection Techniques

As previously stated, payload inspection techniques are based on the analysis of the information in the application layer payloads of the packets. Thus, most payload inspection techniques [Sen, 2004a, Yeganeh, 2012] rely on DPI

[Mochalski, 2009] which checks the packet payload against a set of known protocol signatures (e.g. 'GET' signature in Web traffic).

DPI is defined [Mochalski, 2009] as being "a computer networking term that refers to devices and technologies that inspect and take action based on the contents of the packet (commonly called the *payload*) rather than just the packet header."

The most important components of DPI are regular expression matching and signature based scanning. In the basic approach, the payload of all the packets is checked against the set of known protocol signatures, being the flows the objects to be classified, although it is also possible to classify individual packets. Thus, DPI will need first to parse the packet headers, up to the layer 4 (shallow packet inspection) which is essential to identify the flow to which the packet belongs to. To classify the packet and, subsequently, the flow, DPI will then need to inspect the entire payload, beyond the layer 4 header, to match applications' signatures.

Some variants and refinements are possible regarding the signature matching mechanism and the amount of analyzed information from each of the packets and/or flows. For instance, Statistical Protocol IDentification (SPID) [Choi, 2012] uses entropy-based comparisons of probability distributions, relying on the payload content. It measures, for example, the frequency at which all of the possible 256 byte values occur in a packet. Thus, the message type coded as 0x16 (SSL Server Hello packet) should have higher frequency in SSL traffic.

On the other hand, in order to reduce the high computational cost associated to the inspection of the whole payload for all the packets traversing the network, a lot of effort is devoted to studying the impact of sampling this data (see Chapter 4 for details and related work).

Some less relevant alternatives to DPI relying on traffic payload can be found in the literature. For example, the payload can be checked against the content redistribution characteristic of P2P applications [Lu, 2007], which is based on the likelihood that a P2P peer would usually redistribute the same content it receives to other peers.

Most importantly, DPI based methods are characterized by providing the highest accuracy level with the widest range of covered applications. In fact, if the existence of unlabeled output is accepted, the performance of DPI methods can approximate to 100% for the labeled flows/packets. Due to this, it is usually considered as the reference validation method (i.e. to build the ground truth) to compare other systems with.

However, DPI techniques have some limitations concerning the user privacy and the computational overhead associated with payload disclosure and inspection. Moreover, DPI methods require specific, updated and consistent applications' signatures to cope with the exponentially increasing number of Internet applications and floating signatures [Keralapura, 2010]. Although the process of signatures' extraction can be automated [Keralapura, 2010, Yeganeh, 2012], many of the existing DPI tools, e.g. [L7-filter, 2013, nDPI, 2013, Tstat,

2013], use inconsistent sets of manually updated signatures. In fact, applications' signatures might be differently located within the flow stream [Khalife, 2013b, Aceto, 2010] as different inspection algorithms might be also applied according to the DPI tool of choice.

In fact, DPI has been implemented in many open source classification tools, such as OpenDPI recently replaced by nDPI [nDPI, 2013]; Tstat [Tstat, 2013], an open source passive sniffer providing several insight on the traffic patterns; L7-filter [L7-filter, 2013], an open source application layer classifier for Linux's Netfilter; and Snort [Snort, 2013], an open source network intrusion prevention and detection system.

Nevertheless, and although DPI implementation might be tool-dependent, the same concept is usually followed in making the DPI classification decision and matching signatures within the inspected payload [Mochalski, 2009].

Moreover, the literature shows different DPI based tools that incorporate other joint technologies, such as behavioral [Zhang, 2010], statistical [Dehghani, 2010], port based [Aceto, 2010] and Deep Flow Identification (DFI) [Wang, 2009a].

2.1.3.2 Simple Statistical Techniques

Statistical methods are based on the underlying assumption that traffic at the network and transport layers may have some statistical properties which are unique for certain applications. This assumption proved to be valid for many applications and retained its validity for encrypted traffic [McCarthy, 2011, Bernaille, 2007, Yildirim, 2010, Alshammari, 2011].

Statistical based classification approaches use general traffic features to identify traffic applications without relying on payload analysis. In a simple statistical model, representative sets of parameters are compared to the observed ones [Moore, 2005a, Crotti, 2007, Wang, 2010].

In this context, a group of general flow discriminators relevant for traffic classification was early defined in [Moore, 2005a].

Each statistical method uses a separate set of statistical features and functions. Simple statistical approaches include (Figure 2.1) basic statistical parameters [Yildirim, 2010, Alshammari, 2011], heuristics and profile-based techniques [Karagiannis, 2005, Trestian, 2010]. Next, some details about them are discussed.

A) Basic statistical: Basic statistical techniques observe a set of selected traffic properties to build a fingerprint for each protocol. Observations are then compared against these fingerprints using basic statistical functions.

For example, protocol fingerprints were derived in [Crotti, 2007] based on packets' interarrival time by using simple PDF and normalized thresholds. Results provided up to a 91% [1] of accuracy for some known pro-

[1]Evaluation metrics are reported by authors for various protocol sets and under different

tocols such as HTTP, POP3 and SMTP. Results were, however, site-dependent as pointed out by the authors. Similarly, based on the packet size, a wider range of protocols was covered in [Wang, 2010] including File Transfer Protocol (FTP), Internet Message Access Protocol (IMAP), Secure Shell (SSH), and TELNET, while maintaining the overall accuracy at 87%. Some types of encrypted traffic (e.g. tunnels transporting Skype based VoIP traffic [Yildirim, 2010]) can be also identified using these basic techniques.

However, none of the basic statistical approaches [Yildirim, 2010, Crotti, 2007, Wang, 2010] has provided deeper insights into complex applications like P2P. For this purpose, enhanced statistical approaches were proposed under different terminologies such as heuristics, behavioral, profiling, and characterization.

Enhanced statistical methods construct a set of statistical rules concerning traffic properties to describe flows and hosts using specific applications. The statistical characteristics on which most of these methods rely were experimentally selected.

Some of the statistical derived terms were used interchangeably in the literature (e.g. traffic behavior, characterization and profiling [Bolla, 2008, Sen, 2004a]) and others were used in the method description (e.g. heuristics and behavioral [Zhu, 2010]). Since researchers are more concerned with the technical significance, approaches using statistical derived terms covering heuristics, behavioral, profiling, and characterization, were categorized under the statistical group at the technique level in Figure 2.1.

B) Heuristic: A heuristic is an approximation about some statistical traffic characteristics and attributes, generally defined as a set of rules [Karagiannis, 2005, Karagiannis, 2004, Wang, 2007]. An example of P2P heuristic [Karagiannis, 2004] supposes that the concurrent use of TCP and UDP sessions between two hosts with the same port number is an indicator of P2P activity. Port-based classification [IANA, 2013], mentioned previously, can be considered as another kind of heuristic mapping applications to IANA registered port numbers (e.g. web traffic uses TCP port 80).

C) Profiles: Profiling measures the heterogeneity level and formalizes it in a value that can be used to compare traffic identification objects. For example, an application can be profiled using the PDF of the payload length and direction of the first few packets in a flow [Wang, 2009b].

controlled scenarios. As such, they cannot be regarded as the major criteria for comparisons. For adequate comparisons, further efforts are needed by the research community as we recommend by the end of this chapter.

Then, an observed flow is classified by matching its characteristics to the closest application profile.

Although profiling and behavioral techniques are usually used for security analysis and anomaly detection [Kim, 2006], they can also be employed for traffic classification purposes.

Profile characteristics are often expressed using quantitative metrics such as entropy [Gomes, 2008], graphical means [Karagiannis, 2005], etc. In contrast to classifying the traffic based on simple heuristics, these techniques provide a higher level view of the application's characteristics [Karagiannis, 2005, Trestian, 2010, Keralapura, 2010, Lee, 2007, Wu, 2009, Dedinski, 2009, Wang, 2009b, Gomes, 2008, Zhu, 2010].

Profiles can be built for applications, hosts, and users:

-*Application profiles*. They are defined by a set of rules summarizing statistical properties shared by flows or packets that are generated by the same application. For instance, profiling the packets' size [Wang, 2009b, Gomes, 2008, Zhu, 2010] proved to be discriminative for P2P. Specifically, examining successive short equal-sized UDP packets in P2P application behavior achieved 96% of P2P flow accuracy [Zhu, 2010]. Similarly, profiling the payload length and direction of the first few packets in a flow (a similar approach is proposed in Chapter 5) achieved more than 90% of recall for many P2P applications (Maze, Thunder, PPLive and Feindian) by incorporating the longest common subsequence mechanism [Wang, 2009b].

Part of application profiling is to specifically characterize the signaling phase. The assumption here is that application signaling, regulated by the underlying protocol, should serve as an unique signature for identification purposes. For instance, e-Donkey and FTP signaling traffic [Dedinski, 2009] show differences both in the distribution of packets' inter arrival times and sizes. From a similar perspective, the DNS-query behavior [Wu, 2009] can be used to characterize P2P hosts. The intuition is that P2P applications are supposed to send no or fewer DNS queries than non-P2P applications, as most peers are usually registered by their IP addresses. With DNS-query profiling, 90% of accuracy is achieved for P2P protocols covering BitTorrent, eMule and PPStream.

-*Host profiles*. At a higher level of identification objects, attribute values may be aggregated to build host profiles which can be expressed in several ways. For instance, using graphical means, host profiles can be represented by graphlets [Karagiannis, 2005], revealing host interactions at the application layer reportedly providing up to 95% of host classification accuracy covering most known protocols and including P2P hosts.

Another way of building host profiles can be based on observing their generated flows' characteristics. For instance, by aggregating the number of unique source port numbers and produced flows [Lee, 2007], hosts can be ranked as busy servers, web-servers or P2P.

Flow characteristics used to build host profiles can be quantified. For instance, the *Discreteness of Remote Hosts* [Cheng, 2007] is a value formulating the high number of remote networks to which a P2P host is likely to connect. With this type of host profiling, results achieved more than 90% of accuracy in detecting BitTorrent hosts.

Unconstrained Endpoint Profiling (UEP) [Trestian, 2010] is another promising approach suggesting a fundamental change in host profiling. Key differences exist with state-of-the art approaches, such as BLINC [Karagiannis, 2005] (which also incorporates basic graphical techniques), although both methods are used for host profiling.

First, UEP is different in design by actively crawling Web-based information instead of exclusively relying on network traces.

Second, UEP is able to provide finer-grained host classes such as Kazaa or Yahoo chat, whereas BLINC provides generic host classes such as P2P or Chat. As per the conducted experiments in [Trestian, 2010], UEP was able to classify over 60% of traffic compared to 53% with BLINC [Karagiannis, 2005].

Finally, active host profiling [Trestian, 2010] sending probes to hosts to detect applications in use can be promising as a valid alternative for the current profiling approaches.

-*User profiles.* They analyze traffic characteristics associated with the user behavior [Ullah, 2012, Jinsong, 2009]. Long periods of connectivity and nightly download activities [Jinsong, 2009] are examples of P2P users's characteristics.

User profiles can be translated into host and flows properties for traffic identification purposes. However, characterizing user behavior was less explored in the context of traffic classification, although it might have a potential to offer new discriminative information.

Up to this level, selected examples of existing statistical methods are shown together with their discriminative power in traffic classification and the associated low computational costs. However, statistical based classifiers suffer from several limitations. First, only few applications can be detected with these approaches. Second, some hosts or applications may still exhibit similar profiles (e.g. P2P peers and highly active web clients [Cheng, 2007]).

In addition, most profiling approaches [Karagiannis, 2005, Lee, 2007, Cheng, 2007, Wu, 2009] assume that only one application is in use by each host, which is a non-realistic assumption, as stated earlier in Section 2.1.1.1. In other words, the statistical traffic properties characterizing each application in sep-

arate might become indiscernible when measured for a host running multiple applications simultaneously.

Fewer works [Lee, 2007, Chang, 2009] built host profiles by correlating behaviors for many applications, but they were proposed, however, in the security context. In this regard, active host profiling [Trestian, 2010] might be considered as a valid alternative for the current profiling approaches: Probing a host for each application in separate might lead to more accurate classification results, regardless of the number of running applications.

Another important aspect of statistical methods is that they need to rely on large datasets and multi-dimensional spaces of attributes in order to extend their classification capabilities. In expressing variations in such multidimensional space, simple heuristics, statistical or even behavioral rules might become complicated and poorly readable, requiring thus much higher computational power. In these conditions, tedious human interventions should be dedicated for creating and adjusting the huge number of required statistical rules, and their mapping to different traffic classes. Alternatively, this process could be partially or even fully automated through ML techniques, borrowed from the field of artificial intelligence, as shown next.

2.1.3.3 Machine Learning Techniques

ML techniques are used in various fields including traffic classification. In this case, classification falls into the Pattern Recognition [Theodoridis, 2009] scientific discipline.

In a first approach, statistical based ML algorithms were extensively used to mine relevant information from the huge amount of traffic characteristics.

Traffic classification [Nguyen, 2008] was posed in the context of ML as supervised, unsupervised and semi-supervised learning problems (see Figure 2.2).

The most relevant works and methods based on ML are:

A) Unsupervised Traffic Classification

Unsupervised learning [Theodoridis, 2009] classifies traffic objects without relying on any labeled instances for training. New unlabeled traffic objects, as depicted in Figure 2.2, are directly applied to the classifier at the input level.

In the context of traffic classification, simple unsupervised clustering algorithms were mostly applied though many other unsupervised clustering algorithms exist [Theodoridis, 2009]. Most unsupervised traffic classification [McGregor, 2004, Bernaille, 2006, Kurt, 2012] uses to group unlabeled instances (see Figure 2.2) into clusters, based on their feature vectors, and according to a given similarity function or distance measurement.

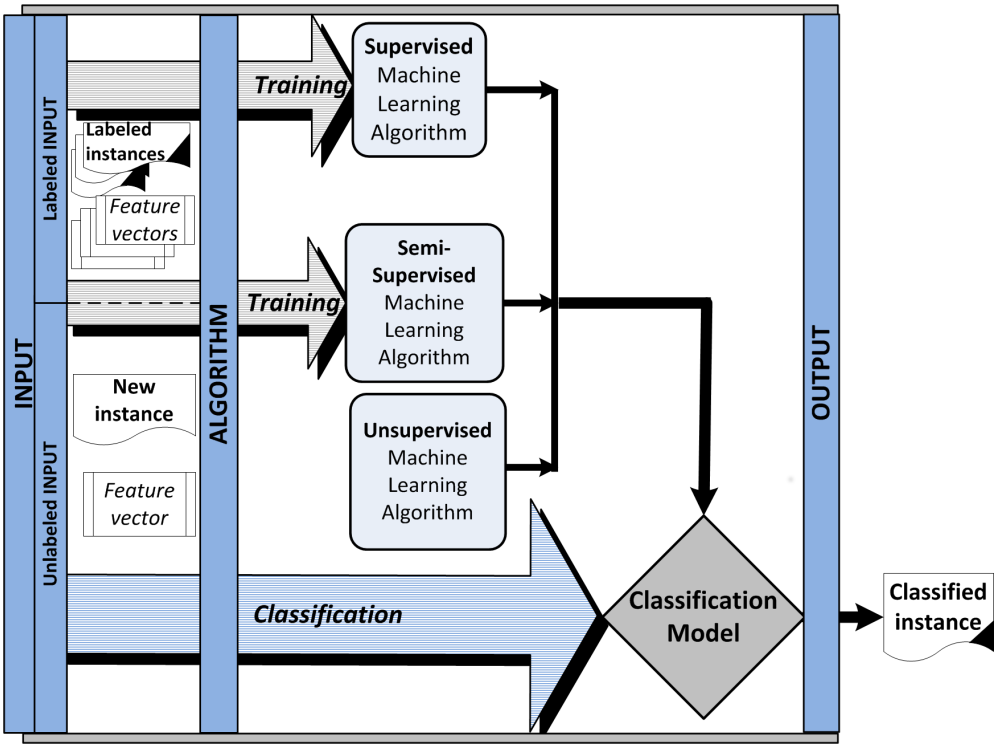


Figure 2.2: Generic diagram of machine learning based traffic classification

For example, in unsupervised flow classification [Bernaille, 2006], clusters are formed by measuring the Euclidean distance between points representing flows in a multidimensional space. A new flow instance will belong to the closest cluster (i.e. having the minimal distance to the cluster's representative point).

At the output level, the first historically applied unsupervised clustering methods [McGregor, 2004] map traffic instances into generic classes such as bulk transfer, small transactions, etc. Classes were so generic as these methods lacked for the derivation of new attributes to further discriminate between individual applications. These approaches relied on few traffic characteristics, such as the packets' order [Bernaille, 2006] and sizes [Bernaille, 2007] at the input level.

Subsequent clustering approaches, such as AutoClass [Zander, 2005], were based on more discriminative characteristics such as packet inter arrival time, packet length, flow size, and flow duration. With this approach, more granular results were provided in discerning applications such as HTTP, SMTP, Telnet and FTP with a reported 80% of accuracy.

P2P applications were identified using K-means clusters [Erman, 2007b] with 95% of flow accuracy. This approach was based on similar attributes

to that used in AutoClass [Zander, 2005], but focused on the server-to-client direction.

On the other hand, to improve the classification performance and speed up the convergence of the clustering process, a constrained variant of K-means was proposed [Wang, 2012]. This approach was motivated by observing correlations between flows, and used additional attributes such as the number of packets and the volume of transferred bytes. Results achieved over 90% of F-Measure for most known protocols, including P2P BitTorrent.

Another relevant issue concerns the input requirements for clustering methods. For some approaches, it was not shown how many flow packets are required to obtain a reliable identification [Zander, 2005]. However, for online deployment, it is important to reduce the size of the input features' vector. For this purpose, the clustering process in early identification methods [Bernaille, 2007, Bernaille, 2006] is based on the characteristics of the firsts flow packets.
For example, by using the size of the first five data packets of a flow [Bernaille, 2006], 95% of flow accuracy was maintained while enhancing the classifier performance. Results covered most known protocols including e-Donkey and Kazaa P2P applications. Similarly, and based only on the first three examined packets [Bernaille, 2007], early identification were able to cluster known applications including BitTorrent and e-Donkey P2P applications, in addition to SSL traffic with 85% of accuracy.

A relevant issue is to enhance the classification accuracy of the clustering process. For this purpose, many helper methods as port-based heuristic [Bernaille, 2007] or Markov models [Kurt, 2012] can be jointly used. For example, incorporating a Markov model [Kurt, 2012] provides a kind of context dependent classification (as detailed in the next section) into the clustering process. In this case, similar flows are assumed to belong to the same cluster already established from previous flows. This approach successfully grouped together flows (such as streaming, VoIP, and P2P flows) that were generated from the same Markov model.

As a result, it is documented that traffic classification through unsupervised learning and clustering is considered among the simplest ML mechanisms. Clustering performs much faster coping with emergent or unknown applications, compared to updating new signatures in DPI based methods.

However, critical challenges with clustering include identifying applications that do not dominate any of the classes and dealing with the cases in which too many clusters are obtained. Although clustering may provide an in-depth classification of similar traffic type generated by different

protocols, it may be only suitable as a first step of classification for those situations where the traffic is completely unknown.

Consequently, in order to build more complex classification models, a classifier should be trained on labeled traffic instances, as detailed next, by referring to supervised and semi-supervised classification algorithms.

B) Supervised Traffic Classification

Supervised learning [Theodoridis, 2009] learn from a set of pre-labeled training samples at the input level to automatically associate unlabeled instances with their corresponding classes. As depicted in Figure 2.2, supervised classifiers build a model based on the experience learnt during their training phase. This model is used during the classification phase to map new traffic instances to output classes, based on their input features' vector.

Supervised learning algorithms can be classified as context-free or content-dependent. Context-dependent classifiers assume the existence of interrelations [Wright, 2006, Dainotti, 2008] among different classes. The class to which a new instance is assigned depends on its own value and on the values of other successive or previous instances. Context-dependent supervised classification achieved 90% of general classification accuracy using the Viterbi algorithm [Wright, 2006] and more than 90% with Hidden Markov Models (HMM) [Dainotti, 2008] covering e-Donkey and PPlive P2P applications. Context-dependent classification approaches were less addressed in the literature. Thus, most supervised traffic classification models were context-free.

Supervised learning [Maglogiannis, 2007] was proposed in many traffic classification works [Dehghani, 2010, Wright, 2006, Moore, 2005b, Li, 2008, Hu, 2012] and includes many algorithms such as KNN, Naive Bayes and Bayesian Networks.

B.1) The k-nearest Neighbor KNN:

This approach is based on the consideration that instances within the same class label should exist in close proximity to each other (nearest neighbors), according to a given proximity function (e.g. Mahalanobis distance [Huang, 2009]). To determine the class of an instance, KNN locates the k nearest points and then identifies the single most frequent class label. Applied to traffic flow classification, KNN [Huang, 2009] provided 90% of accuracy for known applications including P2P BitTorrent and eMule.

The KNN algorithm has many advantages. First, KNN is an example of instance-based learning where the generalization process is delayed until the classification phase. This means that KNN does not require any training time because the training instances are

simply stored. In addition, KNN is relatively easy to tune with one single parameter, k. The performance of a KNN classifier can be improved by using fewer training samples and measuring averaged distances to BoF [Zhang, 2013] instead of individual flows. With this improvement, KNN leaded to 20% saving in the classification time with 90% of less training samples per class.

Despite of many enhancements, KNN is still suffering from several limitations [Maglogiannis, 2007] including its sensitivity to the choices of both the similarity function and the value of k which are crucial for the effectiveness of this algorithm. For instance, the value of k proved to be dataset-dependent in [Huang, 2009]. Other limitations of KNN include the relatively high storage and computational costs requirements. Moreover, KNN is deeply affected by the presence of noisy instances in the dataset and the use of irrelevant features which may distort similarity measures leading to misclassified instances.

B.2) Naïve Bayes

Naïve Bayes classifiers [Moore, 2005b, Auld, 2007, Zhang, 2012, Zhenxiang, 2011, Dehghani, 2010] are based on computing probabilities. Given an element, characterized by the observation of its features, these techniques estimate the probability that a class generates such an observation and label the element with the class that provides the highest probability.

The Naïve Bayes method is based on a robust algorithm that trains very quickly and which requires less storage space, dedicated mainly for the variances of the probabilities. Moreover, any missing features' values can be simply ignored and have no impact on the final classification decision

However, Naïve Bayes classifiers are based on the assumption that different features' vectors are independent and have standard Gaussian behavior, under the normality assumption, which cannot be easily generalized.

B.3) Enhanced Bayesian Approach

To overcome these limitations, the Naïve Bayes model has been in turn the subject to many enhancements through kernel density estimations [Moore, 2005b], neural networks [Auld, 2007], payload inspection [Dehghani, 2010, Zhenxiang, 2011], and BoF [Zhang, 2013].

Particularly, to address the problem of approximating every discriminator by a Gaussian distribution, the kernel density estimation theory [Moore, 2005b] was used with a Naïve Bayes classifier to obtain 96% of accuracy covering most known protocols and including Kazaa, BitTorrent, and GnuTella P2P applications.

A relevant issue is to incorporate dependence between discriminators in a Bayesian model. For this purpose, a Bayesian trained neural network was proposed in [Auld, 2007]. Covering the same set of P2P protocols, a 99% of accuracy was achieved.

On the other hand, to improve accuracy, payload inspection was used jointly with a Naïve Bayes classifier to achieve, for most known protocols [Dehghani, 2010, Zhenxiang, 2011], 80% to 93% of precision and recall results. P2P protocols were covered with up to 96% of flow accuracy for BitTorrent, eMule, PPLive, and Skype applications [Zhenxiang, 2011].

Finally, and in order to minimize the training set, BoF with feature discretization and flow correlation were used with Naïve Bayes classifiers [Zhang, 2012]. The classification accuracy was improved by 8% compared to the native classifier.

Nevertheless, enhanced Naïve Bayes classifiers are still based on hard to generalize assumptions, and might not be suitable for datasets with too many characteristics [Maglogiannis, 2007].

When dealing with multidimensional features, more complex models should be employed in order to fit data variations more readily. In this case, more complex supervised learning algorithms tend generally to perform much better, as detailed next.

B.4) Supervised Artificial Neural Network

Simply defined, Artificial Neural Network (ANN) [Theodoridis, 2009] is a mathematical model used for modeling complex relationships between inputs and outputs.

ANN approaches (e.g. [Chen, 2009a, Sun, 2010b, Ting, 2010]) were proposed for traffic identification. It was shown [Zhou, 2011, Ting, 2010] that ANN can outperform Naïve Bayes methods providing more stable classification performance.

In fact, P2P applications, for example, were easily identified using variants of ANN.

Back Propagation trained Neural Networks (BP-NN) [Chen, 2009a], for example, can identify P2P traffic with up to 96% of True Positive Rate (TPR). Similarly, using BP-NN [Gu, 2010] and Feed-Forward Neural Networks (FF-NN) [Zhou, 2011], various P2P applications such as BitTorrent, e-Donkey, eMule, PPstream, PPlive, Kazaa and Gnutella were identified at 85% of classification accuracy. Moreover, Probabilistic Neural Network (PNN) [Sun, 2010b] achieved over 91% of TPR and 87% of accuracy in discerning P2P and web applications.

To produce a low-dimensional and discretized map representation of the input space of the training samples, ANN can be trained using unsupervised learning to obtain Self-Organizing Maps (SOM).

With SOM [Ting, 2010], 90% of traffic classification accuracy was achieved.

As a result, it can be seen that ANNs algorithms proved to be efficient for traffic identification covering P2P protocols. However, the presence of irrelevant features can make ANN training very inefficient, even impractical. When dealing with learning tasks where the number of features is too large with respect to the number of training instances [Maglogiannis, 2007], other alternatives would be preferred, as detailed next.

B.5) Support Vector Machines

A Support Vector Machine (SVM) model [Theodoridis, 2009] represents training samples as points in a high-dimensional space and constructs hyper-planes used for the classification task. SVMs were proposed for P2P traffic identification in large networks, achieving up to 97% of accuracy [Yang, 2007].

Moreover, the decision function of SVM can be weighted [Liu, 2010] based on the False Positive (FP) and False Negative (FN) rates of every SVM when it identifies specific application traffic. With the weighted SVM approach, a wider range of P2P applications was covered including BitComet, Xunlei, PPLive and PPStream, at the cost of an accuracy degradation to 80%.

In addition, SVM classification was also customized for real-time deployments [Tabatabaei, 2012] by observing the first 7 packets in each flow. Accuracy was maintained at around 85% for many P2P applications including BitTorrent, Gnutella, live-streaming, and -Skype.

However, in contrast to Naïve Bayes classifiers, one of the obvious pitfalls of both SVM and ANN is overfitting [Maglogiannis, 2007]. These approaches are based on complex algorithms with poor interpretability and many parameters to tune. Their classification accuracy can be improved at the cost of larger training sets and increased training time. The training process itself would become impractical at some extent. In this context, decision trees, could be alternatively preferred due to their relatively smaller time complexity, as detailed next.

B.6) Decision Trees

Decision trees [Theodoridis, 2009] are easily comprehensible logic-based systems. A decision tree classifies instances by sorting them based on feature values, so that the one that best divides the training data would be the root node in the tree.

In fact, decision trees are reputed by being quite faster than neural networks and SVMs [Maglogiannis, 2007], while still being more resistant to missing or noisy attributes. Moreover, decision trees

are able of updating the identification model incrementally. This is crucial for handling the dynamic nature of, for example, P2P applications, where new communities of peers often attend and old communities of peers often leave.

In the literature, various types of decision trees were proposed for traffic classification. For instance, J48 and REPTree decision trees [Li, 2007] achieved more than 94% of classification accuracy in detecting P2P applications covering BitTorrent, eMule, PPlive, QQ, Kazaa, Gnutella and Skype. Specifically, the C4.5 [Li, 2008] achieved more than 90% of accuracy covering the same set of P2P protocols [Li, 2007] in addition to encrypted P2P applications, based only on the first five flow packets. The classification accuracy of C4.5 decision trees was improved to 97% for the same P2P application set in addition to covering the Xunlei application [Chunzhi, 2010]. The robustness of the C4.5 decision tree was assessed in many encrypted traffic classification works [McCarthy, 2011, Alshammari, 2011]. Moreover, services running over SSH encrypted tunnels were unveiled with 99% of accuracy based on C4.5 decision trees trained on simple statistical feature sets including temporal information. In addition, identifying SSL applications (including SSL BitTorrent) was also made possible using a modified version of AdaBoost and C4.5 decision trees [McCarthy, 2011]. Results achieved 98% of classification accuracy and 0.6% of SSL False Positive Rate (FPR).

Moreover, the literature shows many refinements of decision trees, applied to traffic classification. For instance, Fast Correlation Based Feature Selection [Hu, 2012] is used to remove any redundant features. The efficiency was improved for real-time traffic identification and 90% of classification accuracy was maintained for P2P applications.

However, decision trees are still relatively hard to be changed if new streaming data is acquired after the training is made [Verticale, 2008, Li, 2008]. As such, retraining points are required to offer the possibility of discovering network specific unknowns [Erman, 2007a].

Consequently, for large data sets, and particularly in cases where continuous, rapid data records are to be acquired, the retraining process becomes harder. Alternatively, incremental decision trees using stream mining techniques, such as Very Fast Decision Trees (VFDT), are able to be quickly updated and revised using new data instances. VFDT may be also adapted [Sun, 2010a, Yang, 2011], for limited memory computing environments by controlling the tree size while sustaining good prediction accuracy. As such, VFDT [Raahemi, 2008] outperformed traditional C4.5 decision trees in real-time deployments.

Up to this level, most supervised classification approaches are covered. Despite of their advantages, these methods still suffer from many persistent limitations, inherited from their basic design concept, namely, the training process. In fact, training might include irrelevant features which can deeply affect the classification accuracy of supervised classifiers.

Moreover, obtaining significant large training sets and the associated training cost are among the major limitations of supervised classification approaches. Controversially, when trained on small datasets, supervised classification models would become difficult to generalize and more dependent on the training samples.

To reduce the bias in supervised learning, training samples can be subject to iterative reweighting through meta-learning algorithms such as Adaboost [McCarthy, 2011, Alshammari, 2011].

In the ML context, ensemble learning [Dietterich, 2000] is a systematic method for combining and weighting classifiers to improve the overall classification performance. This concept can be applied in a wider context to cover other techniques, as discussed later in Section 2.3.

In the ML context, a simpler approach is to merge supervised and unsupervised learning processes, to obtain semi-supervised learning, as presented next.

C) Semi-Supervised Traffic Classification

Semi-supervised classification [Theodoridis, 2009] (see Figure 2.2) relies on a mixture of labeled and unlabeled input samples that are fed into the unsupervised clustering algorithm. The labeled samples within each generated cluster are used to associate the cluster with one of the existing classes. Semi-supervised traffic classification is preferred when the training data is scarce. Moreover, semi-supervised approaches in [Erman, 2007a, Yuan, 2008, Zhang, 2012] are mostly adopted for the sake of computational efficiency compared to complex supervised ML algorithms.

For instance, based on few labeled flows, the semi-supervised approach [Erman, 2007a] yielded over 90% of classification accuracy for a variety of applications including P2P. A simple probabilistic assignment is used to map the K-means generated clusters to different application types. This same approach was improved to reach more than 95% of classification accuracy, by automatically labeling unknown flows within the same BoF [Zhang, 2013]. Classification results were improved to 97% when the classified clusters were used to train a SVM classifier [Yuan, 2008]. Results covered many P2P applications such as Soulseek, Skype, BitTorrent, e-Donkey, and QQ.

D) Reinforcement Learning Traffic Classification:

Reinforcement Learning (RL) algorithms [Qiang, 2011] are developed to interact with dynamic environments where labeled samples and examples of optimal outputs are not explicitly provided, but must be instead discovered by a process of trial and error. Many of the existing RL techniques have their origin in different scientific disciplines. Particularly, RL's widest application is in the field of intelligent control and robots. To the best of our knowledge, RL has not yet been explored for traffic classification.

To the best of our knowledge, RL is not yet used for traffic classification problems. Up to this level, most ML based classification methods are covered. Although some ML algorithms raise performance and complexity concerns, a common aspect is underlined for most of the surveyed ML approaches: the classification process is automated while the high problem dimensionality is handled. For these reasons, ML techniques were the most reputed in the literature of traffic classification, especially when considering the number of the surveyed works using ML.

Moreover, ML is a rich science discipline with a variety of additional algorithms. Examples include: multi-label classification, RL, fuzzy clustering, genetic algorithms [Alshammari, 2011], template matching, polynomial algorithms, EC4.5 decision tree [Maglogiannis, 2007], Gaussian processes [Rasmussen, 2006], etc. Although many of these are designed for different purposes (e.g. speech recognition, genetic analysis), they are still worth being assessed in the traffic classification context. Applying genetic programming, for instance, to traffic classification [Alshammari, 2011] achieved 98% of accuracy covering encrypted Skype and SSH tunneled applications.

One of the relevant domains to explore is the multi-label traffic classification. Several algorithms have been adapted for multi-label classification (see Chapter 6). Examples include multi-label decision trees [Vens, 2008, De Comité, 2003], multi-label kernel methods [Boutell, 2004, Elisseeff, 2001], multi-label neural networks [Zhang, 2006] and lazy multi-label classification methods [Zhang, 2007, Spyromitros, 2008], which are the most relevant to our research (see Chapter 6). Particularly, lazy multi-label classification methods such as that in [Zhang, 2007] and BRkNN [Spyromitros, 2008] are derived multi-label versions of the KNN algorithm.

Choosing a preferred lazy multi-label classification method depends on the kind of evaluation metrics and the datasets used, which might be well-suited for one method rather than the other, as concluded by authors in the comparison study in [Spyromitros, 2008].

To the best of our knowledge, a considerable number of ML algorithms [Theodoridis, 2009] have not been systematically explored for traffic classification. The variety of existing pattern recognition algorithms should be an

incentive for future researchers to explore efficient algorithms that matches the best traffic classification requirements.

Beside ML complex algorithms, one of the research trends is to refer to graphical techniques in identifying network applications, as detailed next.

2.1.3.4 Graphical Techniques

With Graphical techniques, interactions among hosts in computer networks are illustrated using graphs whose properties are subject to further analysis and mining. For most graphical representations (e.g. [Allan, 2009, Iliofotou, 2011]), nodes represent IP addresses, and edges are observed exchanges that represent interactions of interest. Host interactions can be illustrated at different layers, mostly at the application and the network layer.

The underlying assumption of graph based classification approaches is that hosts involved within the same application are supposed to reveal specific patterns of interactions. For example, if a peer wishes to download a file using the BitTorrent application, each user must first establish a connection to a central tracker server, then, other connections are established with peers to download portions of the file.

It is important to note that visualizing graph patterns is generally suggestive [Lian, 2010], however, graph metrics (e.g. degree, diameter, etc.) make the intuitions precise, which allows for appropriate classification.

Graph based approaches may use different input types, as per the vertex and edge properties. With the graphical approaches, various objects can be targeted for identification such as flow, host or host communities. Graphical methods include graphlets [Karagiannis, 2005], motifs [Allan, 2009], Traffic Activity Graphs (TAG) [Jin, 2009], and Traffic Dispersion Graphs (TDG) [Iliofotou, 2011], as depicted in Figure 2.3, in addition to visual motifs [Lian, 2010].

A) Graphlets

One of the earliest graphical classification methods is based on graphlets [Karagiannis, 2005], which make part of the host profiling procedure. A graphlet is a graphical illustration used to portray interactions among hosts at the application layer. For this purpose, it consists on showing the relationships between source and destination IP addresses and transport layer port numbers. A sample of P2P graphlet is shown in Figure 2.3a) (taken from [Karagiannis, 2005]) portraying the peer functional role.

Having a library of similar graphlets associated with different applications, is theoretically supposed to, classify any host by specifying the closest matching behavior.

However, similar behaviors may be exhibited in some cases (e.g. P2P server side, web, and games) and only few application types can be detected with this approach. Moreover, this approach requires capturing

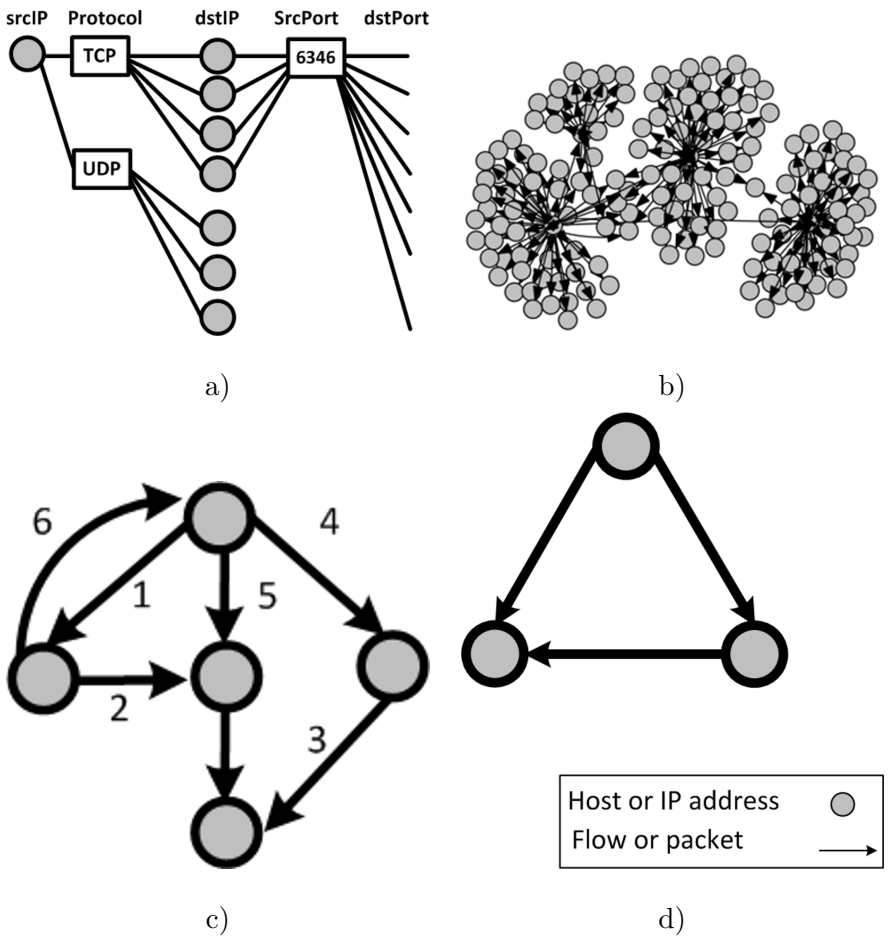


Figure 2.3: Examples of graph based traffic classification approaches: a) Graphlet, b) TAG, c) TDG, and d) Motif

both directions of traffic flows which makes it suitable only for single-homed or edge networks.

In addition, the graphlet based approaches lacked the capability of coping with the dynamic nature of network interactions along with their evolution over time. Nevertheless, several graphical methods were inspired by this concept.

Specifically, dynamic activity and social network graphs are able to provide deeper insights into traffic interaction graphs, as detailed next.

B) Social Networks

Social graphs are usually used to analyze the human social behavior in social network applications. Similarly, social network graphs [Iliofotou, 2007, Jin, 2009, Iliofotou, 2009] suppose that graph patterns of social

interaction among network hosts are specific for each application. A set of nodes (or hosts-community) forming a distinctive graph pattern will be properly associated to the corresponding application.

Social network graphs include TAG [Jin, 2009], which depict host interactions for a specific application. An example of P2P TAG is shown in Figure 2.3b) (taken from [Iliofotou, 2011]), where the main P2P host characteristic (i.e. connectivity with many other peers) is captured by the high degree of connectivity.

Moreover, the P2P decentralized architecture is captured by the high diameter in the TAG graph. TAGs [Jin, 2009] proved to be able to discern HTTP, e-mail, AOL, BitTorrent and DNS host communities.

To incorporate temporal relationships between connections, TAGs are simply referred to TDGs [Iliofotou, 2007]. TDGs include dynamic properties [Iliofotou, 2007, Iliofotou, 2009] that measure how frequently nodes and edges change over time, which gives additional information about the dynamic evolution of interactions for the application in use. This is crucial for detecting applications, such as P2P, which are dynamic in nature. This ability is due to the fact that edges in a TDG are labeled in the order in which the corresponding node interactions were observed, as depicted in Figure 2.3c) (taken from [Iliofotou, 2007]).

A TDG based framework, named Graption [Iliofotou, 2011] has been proposed for traffic classification covering P2P protocols. The TDG based approach [Iliofotou, 2011] showed promising results with 95% of accuracy covering Gnutella, e-Donkey, FastTrack, Soribada, MP2P, and BitTorrent P2P applications.

However, it is important to note that, by examining nodes in isolation, TAGs or TDGs do not provide any information regarding flows between two specific hosts. Instead, TDGs provide an indication regarding the type of application that generated these flows. Moreover, social graphs require capturing a lot of host interactions which are mostly available at the service provider level.

While social graphs assimilate host interactions within computer networks to human social behavior, motif based approaches assimilate host interactions to living cells behavior, as detailed next.

C) Motifs:

Motifs are a small number of specific patterns that occur repeatedly across network types and are thus supposed to be the basic structural elements for broad classes of networks. Motif based analysis is an advanced graph mining technique [Milo, 2002]. The underlying concept is that many of the complex interconnections that occur within a network

are built up from frequently recurring patterns of basic structural elements that is, the motifs. Figure 2.3d) (taken from [Allan, 2009]) shows a motif composed of three nodes.

Primarily, motif analysis has been used mostly in biological networks. However, authors in [Milo, 2002] state that the appearance of network motifs at high frequencies suggests that they may have some specific functions in the information processing, and thus the application interactions, performed inside the network.

A motif based traffic classification model was presented in [Allan, 2009, Allan, 2008] (detailed in Chapter 6). Motifs are detected by mining hosts' interaction graphs, filtered for each application in separate. While social graphs rely on traditional graph measures [Iliofotou, 2007, Jin, 2009], motif based traffic classification [Allan, 2009] rely on binary metrics to measure the host participation in each application based on its contribution in the set of motifs associated with that application [Allan, 2009].

Motif based classification has shown the ability of accurately identifying 85% of the hosts, assuming one host contributes in a single application, over a protocol set covering AIM, DNS, HTTP, MSDS, NETBIOS, SSH and Kazaa. Authors compare their results to traditional graph measures and show that class recall was improved for most of the protocols (except for AIM).
However, the default classification model proposed in [Allan, 2009, Allan, 2008] is based on a single-label KNN algorithm, that is, assuming each host contributes into one single application, as mentioned earlier. Moreover, motif based classification proved to be computationally very expensive in terms of training time, especially for order 4 motifs.

Some additional miscellaneous graphical classification methods are worth of being mentioned. Thus, although using a similar terminology to [Allan, 2009], visual motifs [Lian, 2010] are based on a completely different concept. A visual motif refers to representing a series of client-server interactions based on the sizes and timing of packets.

Other graphical classification approaches [Gallagher, 2010, Kim, 2011] make use of flow similarity intuitions (e.g. for analyzing hosts' interactions graphs). Inspired from context-dependent classification, the concept is that flows sharing the same IP host address [Gallagher, 2010], or the same transport layer port number within a large connected component [Kim, 2011], are supposed to have the same application type.

As a result, it has been shown that most graphical techniques rely on network layer addresses and transport layer port numbers as their basic traffic input properties. Consequently, these methods are less affected by network dynamics compared to statistical approaches relying, for example, on packets interarrivals times.

However, Graphlet, motif, TAG and TDG based methods are examples of host based identification approaches. As previously mentioned in Section 2.1.1.1, these methods are not applicable for cases where multiple applications are simultaneously in use. In addition, dealing with complex graphs results, in most of the cases, in a poor classification performance.

For these reasons, most of these graphical techniques are used as a secondary classification technique within a classifier design. In this regard, incorporating more than one technique is an important research trend in the literature for which, many papers were published, as shown next.

2.1.3.5 Hybrid and Miscellaneous Techniques

An active research trend in traffic classification is the use of hybrid techniques. The idea is to combine different classification techniques that complement each other in making the final decision. These are referred to as multi-classifiers or hybrid classification systems.

For example, a multi-classifier may resort to host-based heuristics after payload inspection has failed in identifying a traffic flow [Keralapura, 2010].

Another relevant example is that of SPID [Choi, 2012], a hybrid technique integrating DPI with statistical analysis. Its traffic model contains a set of fingerprints represented as probability distributions of different traffic attributes. Payload measurements at the application layer include byte frequencies and offsets for common byte values. SPID showed promising results with an average 92% recall in identifying standard and P2P application protocols. Enhanced SPID [Qiang, 2011] can identify additional applications in real-time by using a smaller size fingerprint database through a modified set of attributes such as the number of direction changes and the first payload size. With this enhancement, 17 standard application protocols are identified including Real-time Transport Protocol (RTP), real time messaging protocol, Internet Relay Chat (IRC) together with progressive tunneled video download protocols.

In the context of ML, fusion of multiple classifiers can be performed at different levels. The score level fusion [Ichino, 2010] is the most preferred one, as argued by the authors. Similarly, meta-learning algorithms like Adaboost [Alshammari, 2011] and ensemble classifiers [Dietterich, 2000], construct a set of classifiers by taking weighted vote of their predictions in classifying new instances.

A relevant observation is that many of the surveyed works (e.g. [Jin, 2009, Keralapura, 2010, Alshammari, 2011, Szabo, 2007, Aceto, 2010, Zhenxiang, 2011, Ichino, 2010]) reveal a small bias in the current research trends toward the multi-classifiers approach.

However, and to the best of our knowledge, ML-based multi-classifiers are not yet used in traffic classification problems. They are mostly applied to speech and image recognition and other disciplines. In the context of network traffic classification, researchers have only referred to simplified approaches

[Keralapura, 2010, Szabo, 2007]. Examples included resorting to host-based detection after payload inspection fails [Keralapura, 2010] or relying on simple decision-making systems when combining multiple techniques [Szabo, 2007].

In the wider context of traffic classification, a complex decision system (e.g. a multi-classifier [Szabo, 2007]) has to be designed to combine various techniques. In the decision making process, methods with higher accuracy are usually assigned higher priorities [Szabo, 2007].

Multi-classifiers are supposed, not only to inherit the advantages of different combined techniques, but also, to complement and inter-validate their results. These systems should be robust against changes affecting one population of the input features. However, the increased input size, the complexity of the decision mechanism and the multi-classifiers' efficiency should be considered.

On the other hand, few approaches were proposed with less measurable traction in their directions. Examples include active measurement [Trestian, 2010], active networking [Dedinski, 2009], distributed architectures [Keralapura, 2010, Xu, 2009, Bo, 2009] and process-based classification [Szabó, 2008, Zhang, 2011, Miruta, 2012]. Fewer approaches incorporate new technology trends into the classification context such as virtualization technology [Ban, 2011] or cloud computing [Huang, 2012].

Future methods have to assess the real potentials of deploying such techniques in traffic classifiers. For most of these, handling the administrative and budgetary aspects for large scale Internet deployments may be practically unrealizable. For instance, process-based approaches [Szabó, 2008, Zhang, 2011, Miruta, 2012] consist on deploying special middleware programs on end systems in order to inject extra information inside outgoing packets' headers. Distributed architectures [Keralapura, 2010, Xu, 2009, Bo, 2009] recommend multiple subsystems in the classifier design in an attempt to improve the classification efficiency.

2.2 Comparison and Evaluation of Traffic Classification Methods

In order for the research community to reach an optimized traffic classification method, experimental efforts should be performed in order to compare the most promising methods in the literature.

In this context, comparisons should be performed referring to well defined processes in order to provide network administrators with a certain level of confidence in the classifier's results.

Requirements for appropriate comparisons and benchmarks are presented later at the end of this section, but first, let us start by showing what should the evaluation and comparison of traffic classification methods consist of.

Comparing traffic classification methods consists basically on defining a set of procedures and data formats, covering the three levels of the defined

taxonomy, namely:

(i) A set of traffic properties extracted from a traffic capture dataset used for test, at the input level.

(ii) A validation method whose results are used as reference and, evidently, an experimental setup of the classifier in question, at the technique level.

(i) A set of evaluation metrics to quantify the classification capabilities and the comparison processes, targeting traffic object(s) and class(es) of interest, at the output level.

To evaluate a method, the input format should be defined at first, as the set of traffic parameters chosen during the classifier design phase (see Section 2.1.2). Then, the evaluation and comparison should be based on measuring one or more evaluation metrics of interest (detailed in Chapter 3). Nevertheless, most of the literature works focus on only evaluating the accuracy of the classification.

Apparently, the most crucial part of the comparison procedure is the validation method, which should be commonly defined. Comparisons should be performed in reference to one validation method, not one against the other, which would make it easier for the research community to follow up on the progress of works in the literature, as will be shown at the end of this section.

In this regard, the validation method is mainly used to obtain the validation results, referred to as the ground truth, that is, the set of annotated objects, used as reference. As mentioned previously in Section 2.1.3.1, by the time of this writing, the validation method is usually based on DPI [Mochalski, 2009], reputed with the highest classification accuracy.

However, with the existence of many DPI tools as L7-filter [L7-filter, 2013], nDPI [nDPI, 2013]), etc. the DPI algorithm and signatures are not yet consolidated. Nevertheless, most of the comparison works found in the literature referred to DPI as the technology of choice for the validation, as shown next.

2.2.1 Comparison and Limitations of Existing Methods

In this section, key features are compared for main methods in the literature. Instead of studying an exhaustive list of all existing techniques, selected methods are compared for each category. Basically, we focus on desirable classification features, as highlighted in the few existing comparison works [Nguyen, 2008, Maglogiannis, 2007, Yildirim, 2010, Erman, 2007b, Verticale, 2008, Li, 2008, Hu, 2012, Erman, 2006].

Classification features are detailed in Table 2.1 for each method at three different levels. The overall classification accuracy results are shown as presented in the original papers. Most features are qualitatively presented as high (H), medium (M), low (L) or not applicable (N), where H indicates that the

feature is strongly offered by a method; L indicates the opposite case; M corresponds to moderate cases where neither L nor H applies (M cases require additional future assessments); and N indicates that a feature is not applicable for a method. For example, graphical methods rely on static fields in the TCP and IP headers that are more immune to dynamic network changes (H) than methods using packets' interarrivals (L).

In brief, Table 2.1 shows that payload based methods are usually characterized by their high accuracy covering the largest scope of detected applications.

However, non-payload based approaches are still preferred due to privacy protection and to their potential to detect encrypted applications.

Particularly, ML based approaches are the most reputed in traffic classification. Unsupervised ML techniques provide fast classification that does not depend on training sets and has the ability to classify unknown applications. These techniques [Erman, 2006] are usually compared according to their ability for producing a minimal number of clusters that contain the majority of the classification objects, with the highest predictive power of a single traffic class. Particularly, AutoClass [Zander, 2005] outperforms other algorithms such as K-means [Erman, 2007b].

However, it was demonstrated in earlier comparisons [Nguyen, 2008, Maglogiannis, 2007, Verticale, 2008, Li, 2008, Erman, 2006] that supervised classifiers are more accurate than unsupervised ones, despite of being dependent on the training sets. Particularly, supervised decision trees [Li, 2008] outperform most clustering algorithms, and are easy to train and update, while still offering fast classification.

Supervised learning algorithms [Maglogiannis, 2007], are usually compared according to the speed of learning and classification, tolerance to errors in attributes, etc. Each algorithm has specific strengths and weaknesses. For instance, Bayesian [Zhenxiang, 2011] classifiers may be preferred for simplicity and memory saving. Context-dependent classifiers [Dainotti, 2008] are preferable in describing inter-class dependencies.

Also SVM [Tabatabaei, 2012] and ANN [Gu, 2010] provide complex classification models that are suited for large sets of traffic attributes.

Recent comparison works emphasize on the preference for supervised decision trees for real-time [Yang, 2011] and encrypted [Alshammari, 2011] traffic classifications, particularly, the C4.5 decision tree, which is able to outperform both Bayesian [Hu, 2012] and SVM [Yildirim, 2010] classifiers. Although pointing to some preference for decision trees [Verticale, 2008, Li, 2008], these comparison results have yet to be validated through appropriate benchmarking, which requires, in turn, standardizations at different levels, as pointed out in the previous chapter.

2.2.2 Vendor Classification Engines

This section presents an overview of classification techniques used in commercial products (e.g. [Juniper, 2013, Sourcefire, 2013]) used for traffic-management and network-security purposes. These include routers, firewalls, Intrusion Prevention System (IPS), Secure Web Gateways and traffic shapers. Unfortunately, there is very little information available about the protocol classification performed in most of these systems. Furthermore, although relying on common techniques, commercial products often rely on proprietary methods.

An example of a proprietary algorithm used in TippingPoint systems is Protocol Identification via Statistical Analysis (PISA) [Gomes, 2013]. PISA creates a 10-dimensional representation of each fingerprint for each protocol, based on a training set of captured traffic. PISA uses simple average and standard-deviation values of general flow attributes (packet size and inter arrivals) in both directions, in addition to the Shannon entropy of the data at the application layer. It uses K-means to cluster flows for standard and P2P applications including Skype. However, one of PISA's main limitations is the required number of packets to be analyzed before a flow is identified. For example, Skype results stabilize after the 600^{th} packet.

Another example is Juniper's DPI mechanism [Juniper, 2013] that matches patterns in the first packet of a session using a deterministic finite states automata. It has the ability to chain signatures and to specify a maximum number of transactions wherein the signature must occur to be a match.

Network-based Application Recognition (NBAR) [Cisco, 2013], used by Cisco routers, relies on DPI and many application-specific attributes. It is a state-oriented classification mechanism that supports applications with dynamically negotiated port numbers, such as RTP. It is able to support subclassifications, such as HTTP user agent, content-type and Uniform Resource Locator (URL). NBAR2 is an extended version of NBAR that supports evasive applications such as Skype and Tor, cloud-based applications such as Office-365, and even mobile applications such as FaceTime. Cisco Service Control Engine (SCE) is a dedicated hardware DPI appliance that incorporates protocol state analysis together with behavioral and heuristic analysis.

In most of these commercial products, the increasing requirement for content awareness and application visibility explains the DPI integration with statistical and ML techniques together with SSL decryption. Namely, DPI helper techniques included port heuristic in Juniper [Juniper, 2013], behavioral in IPOQUE [Mochalski, 2009] and SVM in Websense [Gartner, 2011].

The future requirements that are driving the market for next-generation products and the key features of future classifiers are considered next.

2.2.3 Requirements for Future Benchmarks

In the following, basic requirements for future assessment and benchmarking of traffic classification methods, are listed and discussed from a research standpoint.

In fact, despite of the remarkable advances in traffic classification, a significant portion of today's network traffic is still being unknown and the best classification model is not yet defined. The best ever traffic classification method has not yet been defined. Obviously, one of the main obstacles for advancing the research in the field is the lack for appropriate benchmarks.

Existing comparisons (e.g. [Yildirim, 2010, Nguyen, 2008]) were ad hoc in the sense that methods to be compared were randomly chosen. Many authors compare their obtained results with obsolete or weaker methods (e.g. port based heuristic) to underline the supposed capabilities of their classifiers.

A core step toward the best traffic classification model is to perform valid comparisons. Appropriate comparisons should take into consideration both the used input types and output formats. First, techniques providing the best output results, while using the same input set, should be elected for each category group at the technique level. Then, representative techniques for each category should be assessed.

However, making such rigorous comparisons requires that the Internet research community promotes convergence on common standards covering the used terminologies, procedures and policies. Table 2.3 shows some basic standardization requirements on the three levels: the input, the technique and the output.

In brief, Table 2.3 highlights on:

(i) The lack of one common platform for traffic classification, one standard validation method and well-defined comparison procedures, at the technique level.

(ii) The lack of standard formats, metrics and application sets, at the output level.

(iii) The lack of publicly available traffic traces and defined traffic attributes, at the input level. At the input level, we suggest creating a standard list associating discriminative characteristics and protocol signatures for each application. In the long term, such a list would replace the existing IANA list, associating registered port numbers to existing applications.

As a result, these requirements are more than crucial for making systematic comparisons leading to the best future traffic classification model. Nevertheless, most of these problems are complex policy rather than purely technical.

2.3 Open Challenges in Traffic Classification

As mentioned in Chapter 1, this thesis will address part of the open challenges in the traffic classification literature. Many challenges will be left for future work as various problems are steadily emerging in face of researchers. In the following, challenges that were acquired throughout the surveyed works are listed from the perspective of the proposed taxonomy.

Future traffic classification models, as per Table 2.2, should be able to cope with various challenges, mostly related to advances in obfuscation techniques, link speeds and emergent applications.

Basically, future methods should be robust, fast, accurate and easy to update, while protecting the user privacy. Most contemporary applications should be classified, including P2P, encrypted and possibly all unknown traffic.

Nevertheless, an essential question has to be answered: *is it technically possible to offer all of these features (as per Table 2.2) through one single method?*

Theoretically, defining the best traffic identification method consists on determining the best path in the taxonomy diagram of Figure 2.1. Practically, most comparison works [Verticale, 2008, Li, 2008] have shown that no single ML algorithm is able to uniformly outperform all other algorithms and for all the applications. This same conclusion applies in the wider context of traffic classification.

In fact, it was acknowledged [Callado, 2009] that the traffic classification decision requires a lot of trade-offs in robustness, reliability, performance, and privacy protection.

From this perspective, it seems reasonable to assert that these trade-offs can be obtained through the multi-classifier model. This model has to be generalized to cover various techniques (e.g. DPI, Auto-class, decision trees, graphical approaches, etc.). In the same direction, semi-supervised learning and ensemble classifiers are valid attempts but are, however, restricted to ML-based techniques.

Unfortunately, the potentially high computational cost associated with multi-classifiers might become a major limitation to further advances in this direction. For this purpose, future methods might also refer to novel architectural designs which might include specialized hardware [Zhou, 2012].

Future methods should answer the need for modeling hosts running multiple simultaneous applications or traffic for multi-channel applications. In multi-label classification [Tsoumakas, 2006], an instance can belong to more than one class at the same time.

This discipline should be more deeply explored for traffic classification problems although it is usually applied on other fields as image processing [Tsoumakas, 2006].

Table 2.1: Comparison of selected state-of-art methods

Method description				General features															
REF	**TECHNIQUE**	**INPUT**	**OUTPUT**	**TECH.**						**INP.**			**OUTPUT**						
			[Identification Object] [Evaluation Metrics] [P2P Protocols]	Immune to missing or irrelevant attr.	Suitable for many traffic attr.	Simple to update	Fast to train	Independent of training sets	Can describe inter-class depend.	Simple classification model	Protects user privacy	Immune to network dynamics	Immune to obfuscation mech.	Overall classification acc.	Provides fast classification	Memory saving	Ability to detect encrypted apps.	Ability to classify unknown apps.	Scope of detected applications
Zhenxiang2011	**Machine Learning** / **Supervised** / Bayes	Flow size, duration, packet size and packet rate	[Flow] [Accuracy: 96%] [BitTorrent, eMule, PPLive, Skype]	H	L	M	M	L	L	H	H	M	L	M	M	H	M	M	M
Huang2009	**Machine Learning** / **Supervised** / kNN	TCP header fields, packet payload size	[Flow] [Accuracy: 90%] [BitTorrent, eMule]	M	L	M	H	L	L	H	H	H	L	M	M	L	M	M	M
Tabatabaei2012	**Machine Learning** / **Supervised** / SVM	Flow duration, idle time, packets, bytes, size and t interarrival times	[Flow][Accuracy: 84%] [BitTorrent, Gnutella, live-streaming, Skype]	L	H	L	L	L	L	L	H	L	L	M	L	M	M	M	M
Gu2010	**Machine Learning** / **Supervised** / ANN	Packet number, size, interarrival time in both direction	[Flow][Accuracy: 86%] [BitTorrent, e-Donkey, eMule, PPstream, PPlive]	L	H	L	L	L	L	L	H	L	L	M	L	M	M	M	M
Hu2012	**Machine Learning** / **Supervised** / Decision Trees	Packet number, size, interarrival times, transferred bytes	[Flow] [Accuracy > 90%] [Kazaa, BitTorrent, GnuTella]	H	M	H	H	L	L	H	H	L	L	M	H	M	H	M	M
Dainotti2008	**Machine Learning** / **Supervised** / Markov	Packets' sizes and interarrival times	[Flow] [Accuracy > 90%] [e-Donkey, PPlive]	M	M	M	M	L	H	M	H	L	L	M	M	H	M	M	M
Zander2005	**Machine Learning** / **Unsupervised**	Flow size, duration, packets' sizes and interarrival times	[Flow] [Accuracy: 80%] [Napster]	M	M	M	N	N	L	H	H	L	L	M	H	H	M	H	M
Zhang2012b	**Machine Learning** / **Semi Superv.**	Number of packets and packet size	[Flow] [Accuracy > 90%, F-measure 60%-90%] [BitTorrent]	M	M	M	M	M	L	M	H	H	L	M	M	M	M	H	M
Karagiannis2005	**Graph.** / Graphlets	IP addresses, port numbers, number of flow packets and bytes	[Flow] [Completness:80%-90%, Accuracy: 95%][P2P]	M	M	M	N	N	L	H	H	H	M	M	M	H	M	L	L
Allan2009	**Graph.** / Motifs	IP addresses; port numbers	[Host] [Accuracy: 85%] [Kazaa]	M	M	L	N	N	L	L	H	H	M	M	L	M	M	L	L
Jin2009	**Graph.** / Social graphs	IP addresses	[Host-community] [-] [BitTorrent]	M	M	M	N	N	L	M	H	H	M	M	M	M	M	L	L
Jinsong2009	**Stat.** / Heuristic	Netflow TCP flags	[Flow] [Accuracy> 83%] [P2P]	M	M	H	N	N	L	H	H	H	L	M	H	H	M	L	L
Zhu2010	**Stat.** / Behavioral	UDP packet size, IP addresses, port numbers, traffic volume in both directions	[Flow][Accuracy: 96%] [eMule, Skype, BitTorrent]	M	M	H	N	N	L	H	H	H	L	M	H	H	M	L	L
Yeganeh2012	**Pay.** / Payload inspection	First 100 payload bytes	[Flow] [precision, recall > 90%] [BitTorrent, Gnutella]	M	M	M	N	N	L	H	L	H	L	M	M	M	L	M	H
Aceto2010	**Hyb.** / Stat. & payload inspection	IP addresses, port numbers, first 32 payload bytes	[Byte] [Accuracy: 97%] [BitTorrent, e-Donkey]	M	M	M	N	N	L	M	L	H	L	M	M	M	L	M	H

Table 2.2: Features of future classifiers, defined at the three levels

INPUT	• Should include a minimal set of discriminative traffic attributes with less or no payload. • Attributes should be difficult to obfuscate, immune to network dynamics and adapted to new technology trends (e.g. IPv6).
TECHNIQUE	• Should train quickly, with less dependence on the training data. • Should be easy to update with low complexity; Should provide accurate (minimizing error rates) and online classification (minimizing computational costs). • Should handle multi-label host classification. • Might integrate more than one technique built on intelligent multi-classifier algorithms • Should be adapted to new technology trends (e.g. IPv6, CCN, SDN, SaaS, etc.).
OUTPUT	• Should identify a wide scope of contemporary application protocols, controlled at granular level, including multi-channel, Web 2.0, P2P, encrypted tunnels, mobile and social networking services based applications.

Moreover, future methods should be customized as well for next-generation and revolutionary network architectures (IPv6, Virtualization Desktop Infrastructure (VDI) [Kochut, 2009], Content-Centric Networks (CCN), Software Defined Networks (SDN), etc.). For instance, with VDI deployment in corporate networks, the traffic is subject to implicit and legitimate tunneling between an end station and the central virtualization server, using vendor specific protocols. With VDI, a new paradigm of traffic identification has thus to be considered.

In the upcoming years of research, we believe that the multi-classifiers approach should remain a valid candidate for the future traffic classification models, and an active research trend in the traffic classification field. This fact should hold at least until a superior alternative emerges. Future revolutionary network architectures should bring innovative ideas opening eventually newer dimensions in network management and traffic classification fields.

In the next chapters, some of the open challenges in the literature of traffic classification will be addressed. For this purpose, an experimental testbed has to be setup to provide valid results and inferences.

Table 2.3: Requirements for the best future traffic classification method, defined at three levels

INPUT	• To explore new information sources (e.g. application layer attributes, cloud-based reputation analysis [Juniper, 2013]-[Gomes, 2013] etc.). • To have one publicly available, free and open source tool for attributes extraction (e.g. similar to [Tstat, 2013]). • To define a standard list of discriminative traffic attributes and protocol signatures associated with each application (similar to [IANA, 2013], the list for registered port numbers). • To offer public repositories of recent traffic traces with enough payload obtained from various real operative networks. • To establish entities that offer execution on their traces (Move-code-to-data) as to have minimal privacy sensitivities.
TECHNIQUE	• To select main state of art techniques and algorithms for comparison; To accomplish community driven and validated benchmarks based on standard formats and procedures. Techniques that provide the best output results while using the same input type should be selected from each category. Techniques representing different categories should be compared. • To define evaluation metrics and thresholds associated with accurate and online classification; To have one publicly available, free and open source traffic classification platform (e.g. TIE [Dainotti, 2009]) and validation tool (e.g. [L7-filter, 2013],[nDPI, 2013]), with well-defined algorithms and consistent signatures. Multi-label [Vens, 2008, Elisseeff, 2001] classification and multi-classifier [Zhang, 2007, Zhang, 2006] algorithms should be explored in future works. • To propose new techniques adapted to roaming users (e.g. Software as a Service classification), high link speeds (e.g. special hardware [Rao, 2010]), and new technology trends (e.g. SDN).
OUTPUT	• To define standard traffic classification objects and classes.

Chapter 3

The Experimental Setup

One of the main objectives of this thesis, as previously stated, is to develop and assess new methods to improve current traffic identification systems. The completion of this objective requires the accomplishment of some additional objectives related to the handling and acquisition of network traffic data which, in turn, is essential for developing and evaluating the classification system.

Thus, this chapter is dedicated to the description of the experimental setup used throughout this thesis, including the details of the captured datasets. The presented testbed and procedures are targeted at handling raw network traffic as input and obtaining a suitable representation for the elements that will be the subjects of the classification. For this, two major questions need to be addressed for a proper assessment of the classification methods to be developed and tested: the quality of the captured data and the labeling of the whole or part of the traffic. According to these concerns, the data acquisition, the obtention of the ground truth and the parametrization of the traffic will be described next. Furthermore, the evaluation of the performance of the new proposals requires of some metrics that will be also introduced and discussed at the end of the chapter, as there exist different possibilities. Nevertheless, some details about the datasets and the parametrization that are exclusively related to a single proposed method are left for those subsequent chapters in which the associated proposal is described (Chapters 4 through 6).

3.1 Overview of the Data Acquisition

The target of the experimental setup are two-fold. First, it will be used to obtain the datasets of real traffic and the associated ground truth to be used for evaluating classification methods; and second, it will be used to extract a large set of traffic parameters, that are potentially relevant for blind classification methods and that will be considered as the inputs for the traffic classification methods.

For this, three main steps (Figure 3.1) are needed:

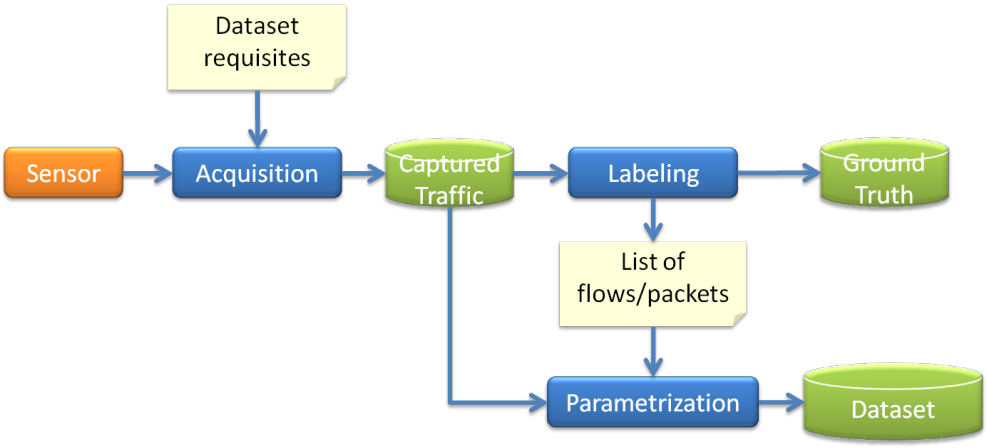


Figure 3.1: Acquisition of datasets of real traffic for experimental purposes

1) Traffic capture: First, a definition of a method and scenario for capturing real network traffic is required. The properties of this traffic are relevant from the point of view of the significance of the results and to enable the obtention of a reference set of labeled traffic (i.e. ground truth). Once defined, a capture of traffic with the selected characteristics will be made by using a set of sensors conveniently deployed at selected points in the network being monitored.

2) Labeling and classification: Once the traffic is captured, it is necessary to classify it or a part of it with an accurate enough method as to be able to compare the results of the proposed methods. This labeled dataset will be the ground truth. For this, DPI will be used to assign a class to each packet/flow in the captured traces. Additionally, during the classification process, the existing flows and their associated packets will be identified and listed.

3) Parametrization: Third, each flow in the traffic is parameterized according to a set of selected parameters, which will be described later in this chapter.

The appropriate implementation of the previous steps requires handling different formats for the data and the development of some tools as well as the adaptation of others. The files, formats and tools involved in the obtention of the datasets are illustrated in Figure 3.2 and will be detailed in Appendix A, although a brief description is presented next. As shown, the whole process is based on the processing of the traffic captured in the selected real scenario, from which both the ground truth and parameters are extracted. For this, *tcpdump* [Jacobson, 1989] is used to capture the traffic from the network in various sequentially ordered files.

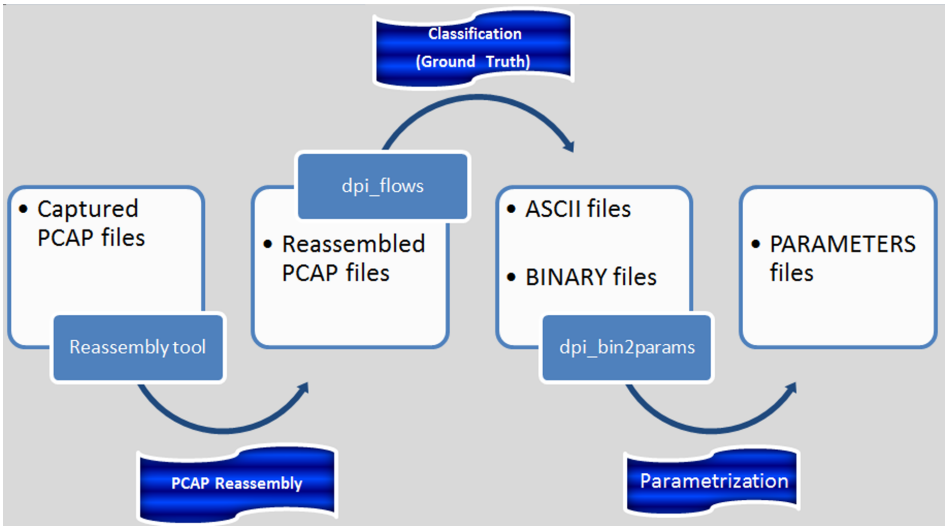


Figure 3.2: Scheme and tools for reassembly, ground truth classification and parametrization of PCAP files

Due to the high volume of traffic required and the use of flow based identification methods, captured traffic files are preprocessed and rearranged so as to group all the packets from an IP in the same file, which also groups all the flows from the same source IP in the same file. This preprocessing is made through a reassembly tool specifically developed to analyze an arbitrary number of consecutive pieces from the capture.

Next, a customized version of *openDPI* [nDPI, 2013] is used to label each individual flow/packet according to a DPI method and to identify all the packets in a flow for further individual processing of each flow. This way, the output of this step will be a list of flows with its assigned protocol, i.e. the ground truth, and the sets of packets in each flow.

Finally, the flows are parameterized through a vector of multiple characteristics by using a tool specifically developed for this purpose that analyzes all the packets in each flow.

In the following sections, each of the modules in Figure 3.1 will be described after analyzing the desirable properties of the captured data.

3.2 Acquisition of Traffic

The procedure for the establishment of an appropriate experimental setup starts with the acquisition of real traffic (Figure 3.1) following various requirements related its final purpose. Thus, before acquiring any dataset it is necessary to analyze the desirable characteristics of that traffic in order to be usable for the training and assessment of the traffic classification methods to

be proposed. From this set of requirements, an scenario and the placement of the sensors are selected, proceeding with the effective acquisition of the packets according to the specifications.

3.2.1 Analysis of Traffic Requirements

The requirements and contents for the data to be captured can be mostly deduced from the targets of this work, as explained in Chapter 1. The two most relevant aspects to consider are the representativeness of the data and the need to evaluate the performance of the methods. Regarding the first point, traffic captures should be representative enough as to be able to accurately infer the statistics or models associated to the classification methods.

On the other hand, the evaluation part requires of a labeled dataset. Thus, in a first analysis, the data should be captured in a real typical environment in a significant volume and including the full payload at least for a part of the data to enable the use of DPI-based methods.

A more detailed analysis reveals the following questions:

- Headers: It is necessary to acquire network and transport layer headers within the capture to be able to handle individual packets and flows when analyzing the traffic stream.

- Payload: The payload beyond transport layer headers is the most crucial part of the capture for the purposes of this work. This information is required in order to check the contents of the packets against the application signatures used by the validation method, i.e. DPI. In most of the works described in the literature, only an initial part of the payload or no payload at all is included in the datasets due to privacy concerns. Obviously, the use of DPI as the validation method requires the obtention of a part of the payloads. However, to obtain more accurate validation results, full payload captures are preferred.

- Real traffic: The traffic capture should be generated from a real network environment. Generating artificial traffic sets is considered of less significance, as the obtained data will lack of the required representativeness and generalization rendering the system useless in real scenarios. In fact, an experimental testbed should resemble, as much as possible, the production environment where the traffic classifier is to be finally deployed and should include a rich set of protocols and services. From this standpoint, the network infrastructure where the traffic is captured should be policed by traditional access control devices (firewall, web filter, etc.). Moreover, real traffic captures should encompass different types of Intranet and Internet services (e.g. e-mail, Web, DNS, etc.) and sessions generated by different user profiles (mobile, desktop, etc.).

- Volume of data: An additional concern is related to the volume of traffic required, which should be representative enough as to contain a mixture of up-to-date application protocols, including challenging ones as P2P related protocols. As stated earlier, P2P identification is considered as a key indicator of the classification capabilities of any given method. On the other hand, there should be enough data from each class so as to be able to properly model them, possibly through training, and to proceed with the evaluation and validation. For this, some partitions of the data are needed. Furthermore, to add a new dimension in this problem, the relative proportions of each of the classes in the dataset should be that of the real scenario in order not to bias the results. All these conditions are difficult to met unless huge volumes of traffic are considered.

- Number of different datasets: Classifiers and classification models trained on a single dataset risk of being dataset-dependent. To prevent any result bias, more than one dataset should be captured and tested, and where possible, from different network environments. This results in a testbed capable of evaluating the resiliency of a classification model over various network environments and its independency on the tested captures.

- Duration and timing: Another relevant question is related to the durations of the captures, that is, for how long should the data be acquired, and the times at which the captures are made. As it is well-known, network traffic is mostly auto-similar, which means that there exist some repetitiveness at different time scales. In order to incorporate most of the variabilities related to this behavior, traces should be captured over extended periods of time, at different times of the day, different days of the week, etc. On the other hand, as the network and consequently, the associated traffic are greatly dynamic and evolve in time, it is advisable to acquire different datasets that are sufficiently spaced in time. Spacing traffic captures in time is important to assess the resilience of the methods to the natural evolution of the networks, and consequently, their generalization capabilities can be better evaluated.

- Sensors placement: The placement of the sensors used to capture the data can be critical, as the quantity, quality and completeness of the observed traffic directly depend on the position at which the traffic is monitored. Thus, the data acquisition should be preferably performed at a border or WAN router as to be able to monitor all traffic in both directions: incoming and outgoing. It is also better to capture the traffic before any NAT or proxy is applied to ensure the visibility of individual host activities.

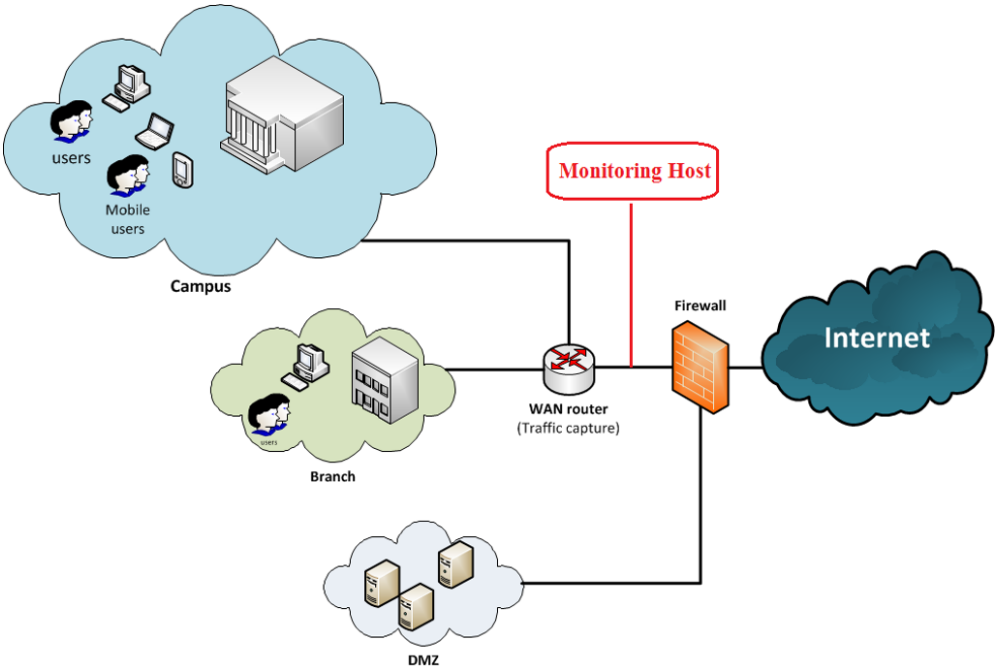


Figure 3.3: Monitoring point emplacement inside the network used for traffic capture

3.2.2 Scenario and Obtention of the Traffic

The capture process was made after a series of choices that were appropriately taken, in order to accomplish with the above requirements, considering the network environment, the sensor emplacement, the number of captures, the size and properties of each capture and the capture duration, among other less relevant factors.

First, to obtain real captures in the required volume, the network of a large-sized academic institution has been chosen as the scenario. Apart from accomplishing with the requirements, this choice presents some operational advantages, as the accessibility and availability of the infrastructure by the researchers involved in this work and the cooperation of the network operators during the procedures. The network under study uses an hierarchical architecture consisting of several remote branches and campuses networks connected to one main site in which most of the services reside, both public and private, as depicted in Figure 3.3. This whole network is connected to internet through an ISP using a single link. The network infrastructure is policed by traditional access control devices allowing the access for P2P applications.

In this network, the optimal placement of the sensor to monitor and capture the traffic is at the main site between the WAN router and the firewall, as shown in Figure 3.3. This way, the sensor is able to observe the whole traffic from the

inner nodes to both Internet and the local servers before any NAT or proxy is applied. Therefore, full flows in both directions reaching any of the inner nodes can be observed.

The sensor used is a PC-like host (monitoring host in Figure 3.3) configured with *tcpdump* [Jacobson, 1989], a well-known and widely used powerful command-line packet analyzer and sniffing tool. This tool allows the use of widely configurable filters and options including the ability to capture full packet payload and to split huge captures into separate smaller files, characteristics which are highly desirable for the purposes of this work. To monitor all the traffic, port mirroring is used in the WAN router.

The full packet payload is captured, as specified. Moreover, this choice is also mandatory for the DPI assessment study in Chapter 4, where we analyze the effects of inspecting sampled payload portions instead of the whole payload on DPI classification performance.

However, as the data to be captured almost surely contain some personal details and sensitive information, some issues regarding its usage and handling need to be addressed. For these reasons, IP addresses were randomized using the method described in the Internet Measurement Data Catalog [Khalife, 2013a] to ensure that no association can be made between content and users (or IP addresses) while preserving the relationships between IPs and the relevant properties from the point of view of traffic classification. To add a greater level of confidentiality, in any further processing of the packets made in this thesis the headers are treated only in a statistical sense and the payloads are examined through automatic means only to determine the nature of the traffic.

Moreover, it is important to note that the user payload is not analyzed by any of the proposed classification methods (in Chapters 5 and 6) which are based on blind classification techniques. The only exceptions are the obtention of the ground truth (later in this chapter) and the DPI assessment (Chapter 4), which strongly rely on the payload for the classification.

3.2.3 Captured Datasets

Next, a description of the raw datasets to be used is presented. Beside the datasets obtained in the previous scenario, and in order to evaluate the classification proposals under different network environments, a review of the publicly available traffic datasets has been carried out.

Although many network traffic captures are available in CAIDA [CAIDA, 2013] and CRAWAD [CRAWAD, 2013], by the time we initiated our work, none of them met some of the critical requirements as the inclusion of full payloads or a significant volume of data. In fact, as mentioned in Chapter 1, most of the current works in the literature related to traffic classification use datasets with limited capabilities in terms of the number of samples and the considered protocols. As will be shown later in this same subsection, the volume of the captured data is orders of magnitude bigger than those in the

Table 3.1: Characteristics of the main traffic datasets

Feature \Dataset	**CS-A**	**CS-B**	**PS-1**
Size [GB]	177	225	4
Unique IPs	656,553	86,938	1981
Total flows	6,430,886	1,144,603	231,493
IP packets	98,548,928	24,541,212	4,394,116
Labeled flows	4,977,123	508,958	199,422
Unknown flows	1,453,763	635,645	32,071
% of Unknown Flows	23%	56%	14%
Duration	3 days	1 hour	31 days
Start Time	Wed 2-6-2010 06:07:12	Tue 29-1-2013 09:22:25	Fri 13-11-2009 12:25:17
End time	Fri 4-6-2010 05:48:47	Tue 29-1-2013 10:22:18	Mon 14-11-2009 13:54:35
Data Source	*Locally Captured*	*Locally Captured*	*Publicly Available* [Corpora, 2009]
Used for proposal	Graph-based	All methods	Message size

publicly available datasets.

Furthermore, only a few datasets including the full payload are available and most of them are from scenarios not relevant for our purposes. Nevertheless, in order to check the proposals in a different scenario, we have considered the [Corpora, 2009] despite its limited size. This dataset includes all the traffic observed during the first four weeks of corporate history of the M57-Patents company.

Two separate traces, referred to as Capture Set (CS), CS-A and CS-B, were collected in the selected scenario during the capture process, over an extended period of time spanning around six months and totaling around 400 GB of real traffic. Additionally, the M57-Patents dataset, sizing around 4 GB, is referred to as Public Set (PS) PS-1. The most relevant characteristics of these datasets are summarized in Table 3.1.

As previously mentioned, it is noticeable from this table that the size of the collected traces is relatively large compared to common capture sizes used in the literature. Figure 3.4 shows the proportions of the sizes of key elements for the three datasets, evidencing big differences between PS-1 and the other two datasets. Therefore, the high number of unique IP addresses and the number of flows for the captured datasets support their representativeness and significance for evaluating traffic classifiers, especially when compared to other works in the literature.

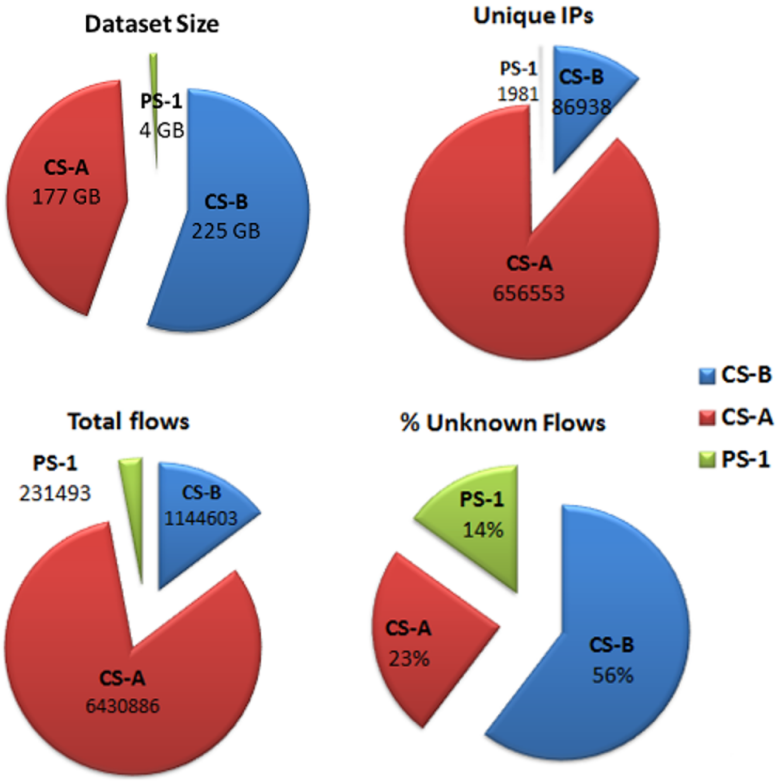


Figure 3.4: Comparison of the volume of traffic for the main traffic datasets

Another interesting observation can be obtained from this table concerning the high capture sizes, which are not necessarily proportional to the capture duration, to the number of IP addresses, nor to the number of flows and packets. This is a clear indicator of the differences in traffic properties when different scenarios are considered and even for the same scenario after some time. As shown, there is a significant increase in the throughput for CS-B when compared to CS-A despite the network is the same.

The way these captures were used throughout subsequent chapters differs according to the experiment purposes and conditions. Most of the experiments are based on the CS-A set which was the first dated capture. Then, for comparison and capture longevity purposes, other captures were used subsequently. Namely, Chapter 5 uses PS-1 in addition to CS-A, and Chapter 6 uses CS-B dataset.

However, due to their huge size, the main datasets shown in Table 3.1 were partially used in most of the cases. Smaller subsets were often extracted to fit with the experimental purposes and to bypass some technical limitations (e.g. memory handling of large file sizes). Moreover, only a subset of the protocols in the main datasets were of interest according to the classification

Table 3.2: Characteristics of the traffic subsets used throughout the thesis

Feature \Subset	**CS-A1**	**CS-A2**	**CS-A3**	**CS-A4**	**CS-B1**
Size [GB]	0.2	3.5	80	82	8
Unique IPs	504	29,516	294,672	301,356	35,419
Total flows	7,337	135,202	3,241,395	3,304,819	323,935
IP packets (kpackets)	137	5,211	126,414	134,773	13,700
Known flows	6,655	67,047	3,241,395	3,304,819	323,935
Unk. flows	682	68,155	972,419	925,349	74,505
% Unk. Flows	9%	50%	30%	28%	23%
Used in Chap.	4	4	5	6	6

scenario. For example, protocols with insufficient or highly unbalanced number of representative flows were omitted (Chapter 5). Similarly, protocols that are not found in both the training and test datasets (Chapter 6) were excluded from the set of targeted application protocols. Therefore, once the traffic is labeled and processed according to the next steps, a division of the datasets in smaller subsets is made. Obviously, at this point in the processing of the capture files, the classification of the flows is still undone. Nevertheless, for completeness, these subsets are described in this subsection.

As a result, a total of 6 different subsets from the captured traffic are considered for selected experiments (Table 3.2). Thus, up to 4 subsets are set from CS-A, being named as CS-A1 to CS-A4. Only a subset is obtained from CS-B, thus being named as CS-B1.

3.2.4 Preprocessing of Capture Files

Once the traffic is captured according to the specifications, the second step in building the experimental setup (Section 3.1) is to identify and label the flows in the dataset to obtain both the list of existing flows, and their respective packets, and the ground truth.

However, as the raw data, as acquired, is composed by a set of limited size Packet Capture (PCAP) files captured in sequence, it is possible that flows are split in two or more of these files. As such, before running any classification tool on the native PCAP files associated with these datasets, and in order to facilitate the handling of all of this information, a preprocessing phase has first to be accomplished.

The target of the pre-processing phase is to avoid flows being split in various files and, whenever possible, to somehow organize the flows sharing a common address, while keeping the file size below 2 GB in order to be able to handle

it in most common computers. In fact, the input data consist of 2 GB files as generated by the used tool (tcpdump). To carry out the preprocessing, an specialized tool (Figure 3.2) has been developed (Appendix A). This tool parses all the input files in sequence searching for all the packets with the same lower IP, being this defined as the lower address, in the numerical sense, from source and destination IPs. The size limit for the file is taken into account to organize the packets in such a way that more than one lower IP can be considered in a single file. In case that all the packets involving the same lower IP exceed the 2 GB limit, those packets are rearranged so as to include all the packets involving an upper IP in the same file.

It is worth to mention that after preprocessing, the database shrank around 20% when compared to the original size. This effect is due to non-IP packets (e.g. Internet Control Message Protocol (ICMP)) and orphan IP fragments being dropped during the reassembly procedure.

The set of files obtained after preprocessing will be the input to the next step: labeling and classification.

3.3 Labeling and Classification

In order for the obtained datasets to be useful for the experimental setup, two major questions need to be addressed next. The first one is related to the identification of the existing flows, that is, to obtain a list containing all the flows.

Furthermore, in order to ease the parametrization, also a list of the packets in each flow is of interest.

On the other hand, ideally each of the flows in this list should be classified as associated to one protocol using an error-free method, as this will be used to evaluate the classification performance of the proposed schemes. For this purpose, a common approach in the literature is to use DPI (see Chapter 2) as the reference method. This procedure is adopted under the assumption that DPI, by the time of this writing, is the most accurate traffic classification method with the minimum possible number of errors.

As previously mentioned, the tool of choice is OpenDPI [nDPI, 2013], a widely used tool based on DPI with demonstrated high accuracy, although at the cost of dubious flows not being classified. Due to the relevance of OpenDPI for a correct assessment of the results, its main characteristics and specifications are described next.

3.3.1 OpenDPI Classifier

OpenDPI, currently nDPI [nDPI, 2013], is a package of software available under the GPL license for the classification of flows and packets according to DPI based methods. To the best of our knowledge, nDPI is the best currently available open source tool for traffic classification.

Originally, OpenDPI derived from the popular L7-filter tool [L7-filter, 2013], including a richer signature library and enhanced classification techniques apart from pure DPI. OpenDPI was the open source version of IPOQUE's DPI engine before it evolved into PACE [Mochalski, 2009], a widely used commercial traffic classifier.

After being discontinued as an open source product, the former OpenDPI is maintained by ntop [nDPI, 2013] as part of their networking tools under the name nDPI [nDPI, 2013], which is released under the LGPL license. The main novelties are that nDPI extends the original OpenDPI library to support new protocols that are otherwise available only on the paid version of OpenDPI and that it is able to integrate with ntop tool.

More than 150 different protocols can be identified with the current version of nDPI, including some P2P and encrypted application protocols. To achieve this, nDPI (and previously OpenDPI, which recognized up to 101 protocols) incorporates, in addition to basic signature detection through pattern matching, various additional techniques such as application behavioral and statistical analysis, described in Chapter 2.

Thus, the used techniques, as stated by IPOQUE [Mochalski, 2009], are:

- Pattern matching, by scanning for strings or generic bit and byte patterns anywhere in the packet, including the payload portion. This way, DPI searches for signatures of known protocols.

- Behavioral analysis, by searching for known behavioral patterns of an application in the monitored traffic. The data used include absolute and relative packet sizes, per-flow data and packet rates, number of flows and new flow rate per application.

- Statistical analysis, by calculating some statistical indicators that can be used to identify transmission types, as mean, median and variation of values used in behavioral analysis and the entropy of a flow.

Therefore, nDPI is not a pure-DPI product as it is not only signature-based but also incorporates information from other sources. This way, the classification accuracy is improved (no miss-classification according to IPOQUE's claims), although some packets and flows still remain unclassified. In fact, unclassified or unknown traffic is still one of nDPI's major limitations.

Nevertheless, and according to its functioning, the capabilities of nDPI are mainly limited by the need to analyze the whole payload of all the packets in a flow in search of signatures (DPI behavior) and to extract the behavioral and statistical information from the flows. Therefore, it is basically a full payload / full flow analysis which imply a high computational cost.

OpenDPI provides a library for the classification of flows and packets. Based on this library, we have developed a customized tool (named dpi_flows, see Figure 3.2) which is able to follow and differentiate the packets in each flow

and provides both the list of flows and packets and the classification of the flows according to OpenDPI signatures.

Specifically, the most relevant customizations in dpi_flows consisted of generating two types of results: flow-based (a list of labeled flows) and packet-based (a list of labeled packets). Labeled flows and packets are then interrelated by simply identifying packets within each flow. On the other hand, dpi_flows operates in batch mode and, once the protocol of a flow is known, all the packets associated to that flow are relabeled as belonging to the identified protocol. The details on the customized tool and the input and output formats are presented in Appendix A.

3.3.2 Results and Analysis

Following the global procedure for the acquisition of the final datasets (Figure 3.1), the classification tool was used over the set of preprocessed files as generated in the first phase to obtain the lists of labeled flows, i.e., the ground truth, and the lists of packets in each flow.

The results obtained for each of the aforementioned datasets and subsets are presented in this section. Figure 3.5 shows the protocol distributions, in terms of the number of detected flows, in logarithmic scale.

It is important to note that unknown traffic, i.e. unlabeled flows, was observed in each capture (Table 3.1). Its relative high percentage can not be attributed to a deficiency in the data capture itself, but rather, to the limitations of the classification tool. The fact is that, as previously stated, the native OpenDPI tool from which the tool is derived was unable to classify all the flows with its current library of application signatures and inspection methods. From a research point of view, the presence of unknowns is problematic since validation results should be 100% accurate. To handle this situation, unlabeled flows will be omitted from the evaluation process, as will be detailed in Section 3.5.

Another less relevant observation concerns big differences among the properties or frequencies at flow and packet levels. For instance, Figure 3.5 evidences, for all detected protocols in our dataset, the difference between the proportions of detected number of packets and flows.

Obviously, the results provided by dpi_flows shows different number of present protocols per dataset. Also, protocols have different distributions across different sets. Nevertheless, HTTP, SSL and DNS were the most predominant protocols for all the datasets, which was expected in a real scenario where most applications and network services are web-based.

As previously stated, an important indicator of the classifier's capability is to be able to detect P2P protocols. Figure 3.5 shows the existence of some P2P protocols in the captured datasets consisting mainly of BitTorrent and a few Gnutella and e-Donkey flows. Although the P2P set is reduced when considering the size of other protocols, in fact the associated traffic volume is

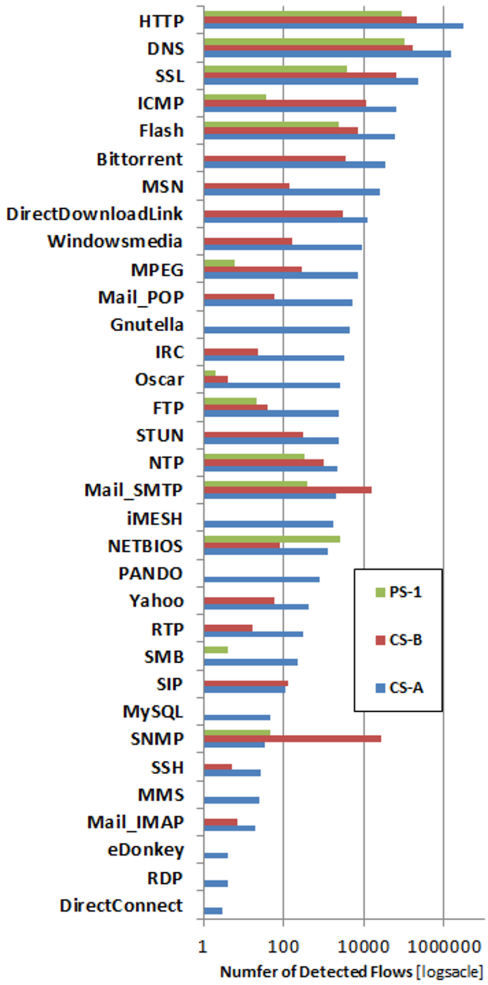


Figure 3.5: Protocol distribution in the main traffic datasets

large enough for experimental purposes due to the large size of the datasets.

As mentioned earlier, only a subset of the protocols detected in the main datasets (Figure 3.6) was of interest for each of the classification methods proposed in subsequent chapters. Besides standard known protocols (e.g. HTTP, DNS, FTP, etc.), most of the selected sets includes P2P protocols.

Another relevant issue concerns unknown flows that were remarkably found in the ground truth results. According to IPOQUE, the non-commercial version of OpenDPI is unable to classify encrypted protocols. In fact, based on our experiments, we validated that most unknown flows (e.g. 98.6% in CS-A2) belong to encrypted applications using TCP ports 22, 80, and 443. Since OpenDPI relies on advanced techniques that go beyond simple port-based classification, these flows were not classified as SSH (for TCP port 22), Web (for TCP port 80) or secure web traffic (for TCP port 443).

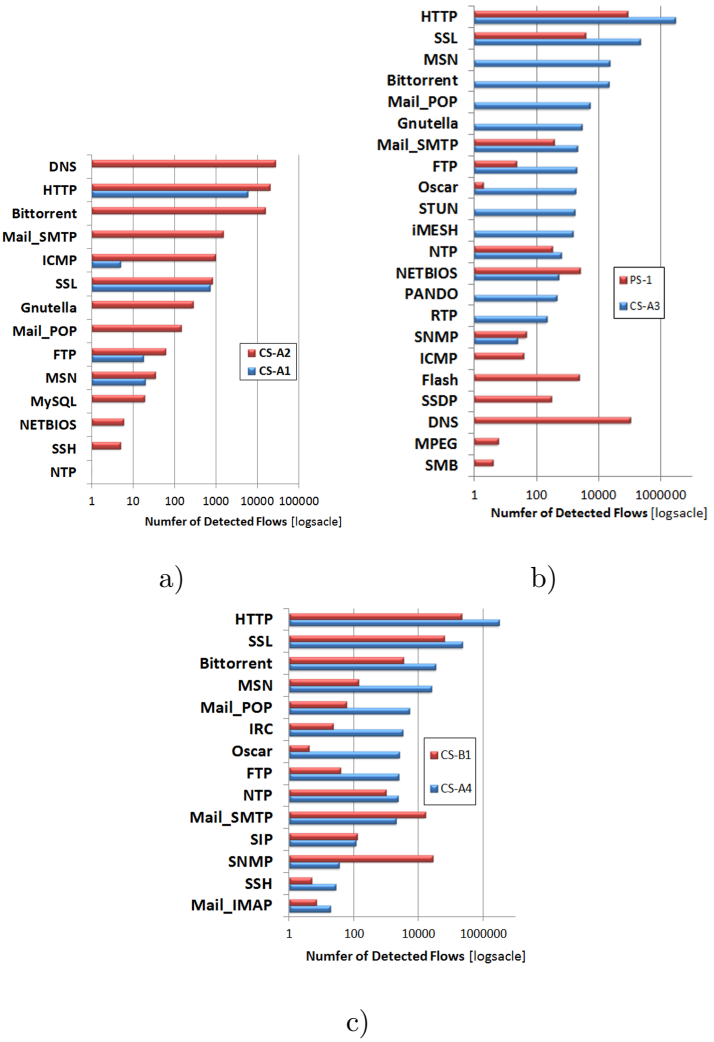


Figure 3.6: Protocol distribution in the subsets: a) CS-A1 and CS-A2 (Chapter 4), b) CS-A3 and PS-1 (Chapter 5), c) CS-A4 and CS-B1 (Chapter 6)

As shown in this section, complete ground truth results were obtained by analyzing the payload in the traffic captures. However, with blind classification methods, various traffic properties should be extracted to be used as the classifier's input, which constitutes the next step.

3.4 Traffic Parametrization

The blind classification methods to be proposed and tested should base their decisions on the analysis of traffic properties that are independent on the content of the payloads (e.g. packet size, flow duration, etc.). Thus, after complete

ground truth results were obtained, a parametrization of the flows existing in the data is made (Figure 3.1) in order to represent each of them as a vector or set of numerical properties. The resulting data will be the input for each of the classifiers.

The first decision to be made at this step is related to the particular set of properties to be considered. As explained in Chapter 2, existing blind classification methods consider a wide range of parameters from each of the elements to be classified. At this stage, the target of the parametrization process in the experimental setup is not to make a condition on the possible methods to be explored. Consequently, we extracted all the parameters that are potentially useful for blind classification as to have a large enough pool of traffic properties for subsequent experiments.

In particular, we considered all those parameters described in the literature [Moore, 2005a] and a few additional ones related to higher order statistics. The set of considered parameters is listed in Table 3.3, which includes up to 60 variables of different natures.

Table 3.3: List of traffic parameters useful for blind classification

Name	Type	Description
Identification set		
FLOW_ID	ui32	Number of the flow (in the file)
ID_PROT	ui32	Protocol as identified by dpi_flow
IP_LOW	ui32	Minor IP address in the session tuple
IP_UPPER	ui32	Greater IP address in the session tuple
PORT1	ui16	Port associated to lower IP (IP_LOW)
PORT2	ui16	Port associated to upper IP (IP_UPPER)
PROT	ui8	Transport protocol (TCP/UDP or as in header: 6/17)
DIR	ui8	Direction of the first observed packet (0 or UP if IP_LOW → IP_UPPER 1 or DOWN otherwise)
FIRST_TIME	ui64	Timestamp for the first observed packet (microseconds)
LAST_TIME	ui64	Timestamp for the last observed packet (microseconds)
Basic data and statistics (Netflow like)		
NPACKETS	ui64	Number of packets in the flow
NPACKETS_UP	ui64	Idem UP direction
NPACKETS_DOWN	ui64	Idem DOWN direction
PACKETS_SIZE	u16	Total size of the exchanged packets
PACKETS_SIZE_UP	u16	Idem UP
PACKETS_SIZE_DOWN	u16	Idem DOWN
PAYLOAD_SIZE	u16	Total size of payloads in exchanged packets
PAYLOAD_SIZE_UP	u16	Idem UP
PAYLOAD_SIZE_DOWN	u16	Idem DOWN
DURATION	u64	Duration of the flow (in microseconds)
MEAN_PACKETS_SIZE	float	Mean size of the packets in the flow
MEAN_PACKETS_SIZE_UP	float	Idem UP
MEAN_PACKETS_SIZE_DOWN	float	Idem DOWN
MEAN_INTERARRIVAL	float	Mean time among consecutive packets in flow
MEAN_INTERARRIVAL_UP	float	Idem only for UP packets
MEAN_INTERARRIVAL_DOWN	float	Idem only for DOWN packets

Table 3.3: List of traffic parameters useful for blind classification (cont.).

Name	Type	Description
N_SIGNALING	u16	Number of packets with flags
N_SIGNALING_UP	u16	Idem UP
N_SIGNALING_DOWN	u16	Idem DOWN
SHORT_PACKETS	u64	Number of short packets in flow (Default: $<$ 100 packets)
SHORT_PACKETS_UP	u64	Idem UP
SHORT_PACKETS_DOWN	u64	Idem DOWN
LONG_PACKETS	u64	Number of long packets in flow (Default: $\geq$ 100 packets)
LONG_PACKETS_UP	u64	Idem UP
LONG_PACKETS_DOWN	u64	Idem DOWN
MAX_INTERARRIVAL	u64	Maximum time among consecutive packets in flow
MAX_INTERARRIVAL_UP	u64	Idem only for UP packets
MAX_INTERARRIVAL_DOWN	u64	Idem only for DOWN packets
MIN_INTERARRIVAL	u64	Minimum time among consecutive packets in flow
MIN_INTERARRIVAL_UP	u64	Idem only for UP packets
MIN_INTERARRIVAL_DOWN	u64	Idem only for DOWN packets
MAXLEN	u16	Maximum packet size
MAXLEN_UP	u16	Idem UP
MAXLEN_DOWN	u16	Idem DOWN
MINLEN	u16	Minimum packet size
MINLEN_UP	u16	Idem UP
MINLEN_DOWN	u16	Idem DOWN
NACKS	u64	Number of packets with ACK flag active
NFIN	u64	Idem FIN
NSYN	u64	Idem SYN
NRST	u64	Idem RST
NPUSH	u64	Idem PSH
NURG	u64	Idem URG
NECE	u64	Idem ECE
NCWD	u64	Idem CWD
NACK_UP	u64	Number of packets with ACK flag (only UP)
NACK_DOWN	u64	Idem DOWN
NFIN_UP	u64	Idem FIN and UP
NFIN_DOWN	u64	Idem FIN and DOWN
NRST_UP	u64	Idem RST and UP
NRST_DOWN	u64	Idem RST and DOWN

The values considered in the parameter vector include both basic and advanced statistical measures and flow properties. Two directions are considered for most parameters: UP, for the packets traveling from the lowest IP to the upper IP (when representing IP addresses as integers), and DOWN for the opposite direction. Among the parameters are also the usual ones included in most NetFlow-like flow analysis as average packet size, flow duration and number of packets [Moore, 2005a], while at the same time a more detailed description at temporary and signaling levels (e.g. interarrival times and number of URG packets) is included. The complexity of the evaluation is low, as

only maximum, minimum, count and average values for each parameter are considered.

Additionally, and in order to ease the assessment of the flows, the categories of each flow, as assigned by the DPI tool, are also included in the parametrization. Obviously, this will not be used for determining the class of the flow but just to account for the correctness of the class as provided by the method being explored.

By analyzing the nature of the parameters, we can consider a feature vector as composed by four main parts:

- An identification vector, which includes all the information required to univocally differentiate each flow and its identification according to the customized DPI tool.
- A transfer related vector, which considers all the parameters related to the number of packets and their sizes.
- A time related vector, including parameters related to temporary characteristics of the flow, as duration and time between consecutive packets.
- A signaling vector, that accounts for the number of packets with signaling information and the associated signals.

As can be deducted from Table 3.3 the values for the parameters are obtained from the list of packets in a flow by analyzing just their sizes, timestamps, TCP flags, if any, and the direction of the packets. This way, no inspection of the payload beyond TCP/UDP headers is made, thus preserving the privacy of the users at the application layer.

The parametrization of all the flows in the captured datasets was made through a tool specially developed for this purpose (Figure 3.2). This tool (dpi_bin2params) takes as inputs the files generated by the classification tool (dpi_flows) containing the list of flows and the list of packets per flow. The details about the tool and the format of the files are provided in Appendix A.

It is important to note that, for the classification scenarios addressed in this thesis, not all of these parameters are used. In fact, at the parametrization phase, it was too early to determine, a priori, which traffic parameters are going to be used specifically. This had to be answered in regard to the research and experimental progress. As such, for each of the proposed methods in subsequent chapters, a different subset of parameters will be analyzed. Moreover, and despite of the large set of parameters obtained after parametrization, additional properties were generated in some cases (e.g. in Chapter 5).

3.5 The Evaluation Method

Once the experimental setup to evaluate and compare classification methods is complete, evaluation metrics should be defined and selected from those de-

scribed and used in the literature.

An evaluation metric is used to quantify the classification capabilities of a given method, being used as the main indicators for the choice of one classification method over the other. For this purpose, besides the experimental testbed used for evaluating or comparing classification methods, a critical choice is to specify which evaluation metrics are going to be measured. They play and important role in the comparison of techniques and even in the tuning of the parameters and/or models used by each method. As an example, if the selected metrics do not take into account the relative percentages of the members in each class, the effect on the metric of completely mistaken classes for the less frequent one can be almost void. This is the imbalance problem in pattern recognition [Theodoridis, 2009].

The most simple metric is the *Percent Correct* (PC), which accounts for the total number of elements correctly classified, N_c, related to the total number of elements to classify, N,

$$PC = \frac{N_c}{N} \tag{3.1}$$

Unfortunately, in imbalanced datasets [Provost, 2001] as is the case with the captured traffic, this measure is meaningless. Therefore, other metrics are required to assess classifiers' performance when imbalanced datasets are present.

In the literature, basic and composed evaluation metrics can be found in the context of traffic classification [Callado, 2009], although most of them are also useful in other applications. For simplicity reasons, these metrics are explained next for a two-classes scenario, although the same ones can be used for multi-class and multi-label classification scenarios [1].

Consider an scenario in which the elements to be classified belong to classes A or B and the target of the classifier is to identify elements belonging to A, that is, the classifier is a detector targeted at selecting elements from class A. Four possible situations can be identified:

- True Positive (TP): an element in class A is correctly classified as belonging to class A,
- FP: an element in class B is incorrectly classified as belonging to class B,
- True Negative (TN): an element in class B is correctly labeled as class B,
- FN: an element in class A is incorrectly labeled as class B.

This situation is illustrated in Figure 3.7. Perfect circles represent the correct classification, i.e. the ground truth. The ellipsoid forms, deviated and

[1] Performance evaluation of multi-label classification systems is more complex than for mono-label systems, and is discussed in Chapter 6.

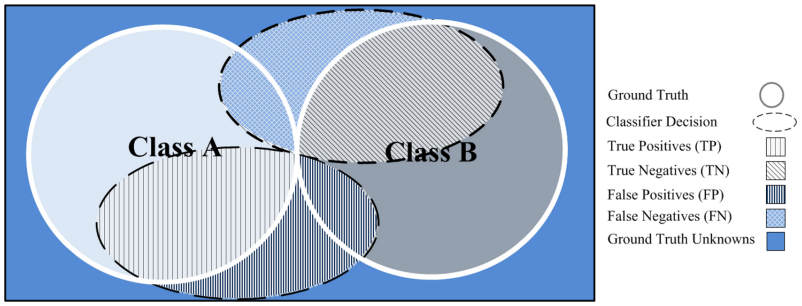


Figure 3.7: Basic evaluation metrics for a binary classification scenario (classes A and B)

deformed in relation to the original circles, represent the classifier's decision. Unknown classification decisions are illustrated by the zone outside both circles (for the ground truth) and outside both ellipsoids (for the classifier).

In this case, PC becomes

$$PC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

Ideally, classifiers should maximize TP and TN while minimizing FN and FP to increase the value of PC. In relation to the illustration in Figure 3.7, this refers to restoring the two ellipsoids to the original circle places and reshaping them to near perfect circular shapes. In all cases, unknowns' zones should be eliminated as much as possible. But, although the target is clear, the effects of modifying one of the measures on the overall performance are not that clear. That is, the impact of increasing TP could not be the same as that from increasing TN in an imbalanced scenario. Therefore, more complex measurements are recommended.

Composed evaluation metrics provide deeper insights into the classifier's performance. Many papers [Nguyen, 2008, Callado, 2009, Dainotti, 2012] used composed evaluation metrics to rank the performance of different classification algorithms. Some of these depend on the used technique (e.g. ML metrics), and sometimes on the classification context (e.g. Bytes' accuracy preferred for differential pricing).

Examples of composed evaluation metrics include precision or accuracy [Nguyen, 2008, Callado, 2009], completeness [Callado, 2009], correctness [Dehghani, 2010], and sensitivity or recall [Nguyen, 2008, Dehghani, 2010], among others. They can be defined as follows:

- Accuracy (Precision): The ratio of instances correctly classified (TP) to the total number of instances classified positively as belonging to the class

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

- Sensitivity (Recall): The percentage of TP patterns that are correctly detected by the classifier

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

- Completeness: The ratio of instances detected to the number of actual instances

$$Completness = \frac{TP + FP}{TP + FN} \tag{3.5}$$

- F-measure: A measure that combines precision and recall. It is the harmonic mean of recall and precision

$$F - measure = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} \tag{3.6}$$

Nevertheless, analyzing a single metric in separate is not a sufficient indicator of the classifier's performance. For instance, both precision and recall should be optimized in order to obtain significant classification results. F-measure [Dainotti, 2012, Iliofotou, 2011] is an example of a single performance measure, expressed in function of both precision and recall. In addition, helper tools, such as Receiver Operating Characteristic Curve (ROC) [Nguyen, 2008], can be used to correlate and to visualize trade-offs between evaluation metrics.

The overall classification accuracy [Nguyen, 2008, Maglogiannis, 2007] averaged for all classes is the most commonly used measure in the literature. The accuracy is calculated –Equation 3.3– as the percentage ratio between the number of instance objects (e.g. flows, hosts, etc.) labeled as belonging to a protocol with the assessed method, over the total number of instance objects labeled as belonging to the same protocol in the ground truth. However, as previously mentioned, when the tested dataset is imbalanced, the overall accuracy results might be biased by the most dominant applications.

Alternatively, a confusion matrix is able to illustrate the classifier performance on a per class basis. It shows the number of instances recognized by the classifier for each class, given the actual class. Table 3.4 shows an example of a confusion matrix for a three classes scenario (A, B and C), where N_{ij} denotes the number of instances with actual class i classified as j. The diagonal of this matrix represent the correct decisions for each class. Values outside the diagonal represent the number of classification errors, which should be minimized. Precision and recall values can be obtained from this matrix for each class.

It is clear that the literature lacks of any common framework for evaluating traffic classification methods. Most of the existing works in the literature consider the classification accuracy as the main evaluation metric, while computational and memory costs are not commonly defined.

Table 3.4: Confusion matrix for multi-classification scenario (classes A, B and C)

$$
\begin{array}{cc}
Actual \backslash Assigned & \begin{array}{ccc} A & B & C \end{array} \\
\begin{array}{c} A \\ B \\ C \end{array} & \begin{pmatrix} N_{aa} & N_{ab} & N_{ac} \\ N_{ba} & N_{bb} & N_{bc} \\ N_{ca} & N_{cb} & N_{cc} \end{pmatrix}
\end{array}
$$

To evaluate the classification methods proposed throughout this thesis, and based on the previous considerations, we select both basic and composed evaluation metrics that we measure for different classification scenarios.

Particularly, we focus on the flow rather than packet accuracy (Chapters 4 and 5), and the host rather than host-community based accuracy (Chapter 6). In fact, we consider that classifying hosts and flows are semantically more significant and more adequate to most traffic engineering tasks than classifying packets (too fine-grained level) or host-community (too coarse-grained level).

As for the computational cost metrics, we choose to measure the processing time in microseconds, consumed by different tested classification modules. For this purpose, we use time related function calls inserted into the classifier's code at the proper places. Classification programs were compiled with GCC v4.4.3 with -O3 optimization level. The server's hardware specifications include 16 GB of memory, 2 Intel(R) Xeon(R) 2.66 GHz processors with 4 cores each.

In the next chapters, the experimental testbed defined here will be used to assess existing traffic identification methods and proposed enhancements. As per the taxonomy defined in Chapter 2, methods of two main categories, at the technique level, will be tackled, namely, non-payload and payload based methods. We start with the assessment of payload-based traffic identification methods, which is the topic of the next chapter.

Chapter 4

DPI Optimization Through Minimum Payload Disclosure

In nowadays networks, performance and privacy issues are considered among the main concerns. However, as mentioned in previous chapters, the native DPI technique is theoretically based on the disclosure and examination of the full payloads of the packets. The user privacy and the classification performance are thus considered among the weaknesses of DPI classifiers.

The work to be presented in this chapter is motivated by the potential of enhancing the user privacy level while decreasing the inspection overhead associated with DPI classification. In this context, traffic sampling is a priori an interesting approach for DPI optimization. It is supposed to decrease the required computational cost by only inspecting payload samples, instead of the complete payload data for all the packets in a flow. In addition, sampling can improve the level of user privacy by disclosing less information from the payloads.

However, to the best of our knowledge, the literature lacks of analysis studies of sampling mechanisms customized for DPI, although various sampling policies can be found [Chen, 2009b]. Moreover, the research community currently lacks of one commonly defined DPI algorithm where a set of application signatures is well defined. This fact makes it more difficult to assess sampling schemes results applied to different DPI classifiers. From a research point of view, this is problematic especially when the need is to generalize one single sampling method for DPI optimization.

In this chapter, a general sampling methodology for DPI, leading to minimum payload disclosure, is presented and analyzed. Rather than recommending a single sampling scheme, a general sampling methodology is proposed, which can be then customized according to the classification context (e.g. DPI tool of choice, protocols of interest, etc.).

To accomplish this, an assessment study of DPI classification used jointly with various sampling schemes is conducted throughout this chapter. The aim

of the experiments is to statistically localize payload sections within a flow stream where application signatures are frequently matched. These portions of the payload will be referred to as the *classification bytes* and will be targeted for sampling.

Specifically, sampling is applied at four different levels: per-packet, per-flow, and combined sampling, both with with contiguous and non-contiguous modes. Complex applications, usually used as key indicators of the classification ability (e.g. P2P), are highlighted in the results.

The aims of this chapter are thus summarized as follows:

1) First, to prove that application signatures, the most important factor for DPI matching process, are usually found at regular locations within the flow payload stream.

2) Second, to localize application signatures within the flow stream with the highest level of granularity.

3) Third, to infer one general methodology for DPI sampling, with minimum payload disclosure to decrease the inspection time, while maintaining the classification results.

Next, the concept of DPI optimization through sampling is presented and formalized.

4.1 DPI Optimization

Understanding the operation of DPI methods and their associated costs is relevant before applying to DPI any sampling or optimization mean. To clarify it, a simple analogy is presented, thereafter, an estimation of the computational costs and the potential gains are evaluated in terms of the classification and inspection times.

4.1.1 A Simplified Analogy

DPI classification is based on the search of any of the elements from a set of predefined application signatures into the payloads[1]. In case a match is found, the packet is classified as belonging to the protocol or application associated to the signature that provides the match and, in most cases, also the flow to which the packet belongs is classified accordingly. If none of these signatures is found, the packet and/or flow will be labeled as "unknown".

[1]Throughout this work, a packet payload is assumed to include the full application layer content of the packet, thus not including layer 3 (IP) and layer 4 (TCP or UDP) headers, but including any application specific header. Moreover, TCP flows are considered to include the 3-way handshaking and the pure acknowledgement packets.

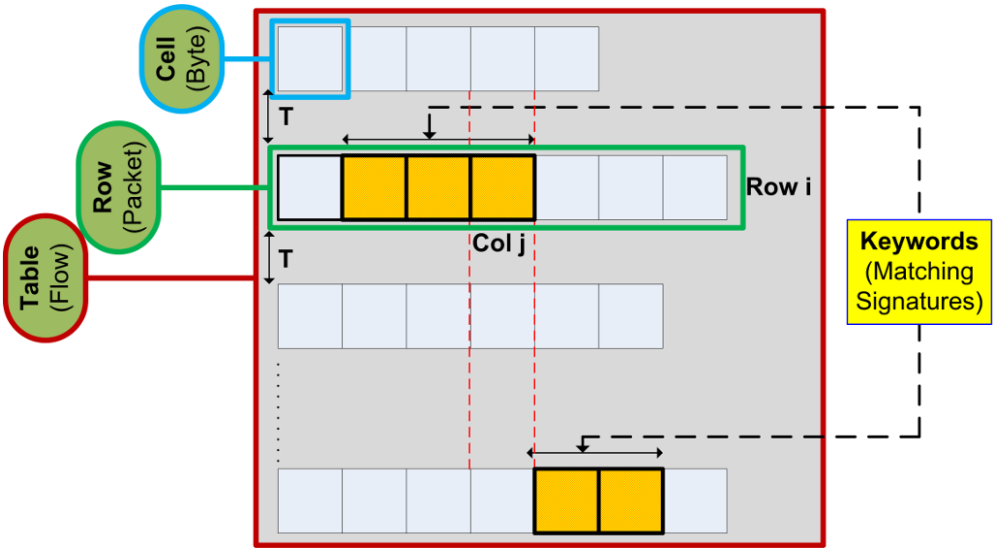


Figure 4.1: Analogy between DPI classification and table keyword search (table representation of a flow)

Based on these considerations, the mechanism used to match signatures in the flow payload stream presents many similarities with the linear search of a keyword in a bi-dimensional data table [Knuth, 1998].

In this analogy, the complete flow payload is simply represented as a data table or matrix, as depicted in Figure 4.1. One full packet payload corresponds to a single row in this table (numbered with row index i), and the payload bytes of a packet correspond to the cells within that row (numbered with column index j). Thus, a flow, F, can be considered as a matrix, $\mathbf{M} = I \times J$, being I the number of packets and J the maximum payload length in the flow. Row i contains r_i non-null elements.

Two perspectives are of interest. First, from a sampler point of view, the table representing a flow can be regarded as vector, $\overrightarrow{p}$, that contains the payloads of all of the packets, serialized row by row, and composed by P elements (bytes).

$$\begin{gathered}\overrightarrow{p} = \{m_{11}, m_{12}, \cdots, m_{1r_1}, m_{21}, m_{22}, \cdots m_{2r_2}, \cdots, m_{Ir_I}\} \\ P = \sum_{i=1}^{I} r_i\end{gathered} \tag{4.1}$$

This vector will be referred to as the *flow payload* vector in the following.

In any defined sampling scheme, the flow payload vector is subject to partial inspection, which is equivalent to analyzing selected parts of the table. On the other hand, from a classifier point of view, matching a signature within the flow is similar to matching a keyword within the table.

Nevertheless, these two processes –signature and keyword matching– have some key differences regarding their usual implementation. First, DPI usually

searches simultaneously for a set of signatures in order to label a flow to one of many well-known application protocols (i.e. multi-classification), which can also be accomplished with keyword matching using suitable algorithms. Second, unlike a data table where rows and cells are usually sequenced and loaded into memory with predefined sizes, the packets payloads should be analyzed in real time and have complex timing characteristics (i.e. packet interarrival time, denoted as T in Figure 4.1) and variable sizes. This imposes some restrictions to the way in which the signature matching should be done, although these differences are mainly operational and are not considered of major importance to this work.

Based on this analogy, the locations of application signatures will be defined simultaneously by the range of packet numbers within the flow holding the signatures and by the range of bytes position numbers where these signatures are found inside these packets. According to Figure 4.1, signature locations are simply defined by the value ranges of (i, j).

Throughout this chapter, the term *classification bytes* is used to refer to regular signatures' positions within a flow or to payload sections where application signatures are frequently matched. Thus, the classification bytes are basically the "unknown" that DPI is looking for, similarly to keywords' locations in the table search process.

In this context, an important question, relevant for both processes, is to be answered: Are the classification bytes (respectively, keywords) statistically located at regular positions within the traffic flow (respectively, the data table)?

In table search scenarios, this is a too general question to be answered as it strongly depends on the nature of the data being searched. However, for the considered protocol classification problem, some regularity in the classification bytes within a flow can be expected due to the presumable similarities in the signalling phase among flows generated by the same application.

In fact, the flows generated by the same applications might have similar content (at the application level messages), sequencing and timing, especially for the first signalling packets. However, this assumption has to be proved by experiment. Proving this regularity will be the first step toward DPI optimization through sampling, as it will point to the most relevant portions of the flows that, consequently, should be the target for the sampling procedures. This will be detailed in the experimental sections of this chapter (Section 4.3).

4.1.2 Sampling Policies

For DPI assessment, despite of the diversity of the existing sampling policies [Chen, 2009b], the schemes to be analyzed are targeted at the following four levels:

1) Per-packet level sampling, where each packet payload is partially inspected.

2) Per-flow level sampling, where a subset of packets are fully inspected per flow.

3) Combined level sampling, where a subset of the packets are partially inspected per flow. Here, two modes of operation can be applied for inspecting the flow payload vector:

   3.a) Non-contiguous inspection mode, where inspected payload chunks are obtained from a subset of sparse packets in the flow. In this mode, flow payload vector parts are omitted between the analyzed samples of the flow payload vector.

   3.b) Contiguous inspection mode, where the payload vector is continuously inspected, starting with the first payload byte, until a predefined size is reached. In this mode, the flow payload vector is continuously inspected without omitting any part of the payload. In other terms, the payload from a packet cannot be partially or fully considered unless the payloads for all the previous packets have been considered.

The main reasoning behind the choice of these sampling policies is related to the behavior and nature of the packets exchanged by most protocols, as most of them uses a signalling phase at the beginning or at some points of the communication to select/set some operative parameters and/or modes of operation. During this phase, commands related to the protocol and their functioning are exchanged, including specific protocol-related keywords mostly at the beginning of those packets. These signalling phases are supposed to be the most representative for each protocol and, consequently, should be the target for a sampling method oriented to identify the protocols.

As mentioned earlier, DPI optimization aims to minimize payload inspection. As such, the primary objective is to specify the minimum payload size and the location of the bytes that should be sampled without deeply affecting the classification results. For this purpose, sampling policies were applied in an increasing complexity order, starting with the simplest one (per-packet sampling), which is also that providing the largest sample size, and down to reach the one providing the minimum possible size. The main aim behind this ordering is to be lead to the minimal possible sample size that could provide the most granular insights location of the classification bytes within a flow.

Another relevant issue is that, at each step where these sampling modes are applied, the order of subsequent experiments is driven by the obtained results. In fact, at the beginning of the experiments, no prior idea could be obtained on the location of the signatures nor on the sampling scheme providing the best results. The order in which the sampling experiments were conducted aims to:

1) First, to prove the regularity of the locations of application signatures, and to compare sampling methods, both per-flow and per-packet sampling were assessed separately.

2) Second, to get more granular insights on the signatures' locations, and based on the previous results, per-flow and per-packet sampling techniques were combined using a non-contiguous inspection mode.

3) Third, to get the smallest possible payload sample size while maintaining the classification results, per-flow and per-packet sampling techniques were combined using a contiguous inspection mode.

By comparing the results of these sampling schemes, a customized DPI sampling scheme can be inferred. For such comparisons to be significant, a study of DPI modules and computational cost analysis is shown next.

4.1.3 DPI Modules and Cost Analysis

An important question for the assessment and comparison of the sampling methods is how to evaluate the gains from each of them. In this sense, sampling is supposed to optimize DPI, in terms of computational costs, as less input is to be analyzed. This should be evidenced through a reduction in the classification time, that is, in the time it takes to analyze and classify a flow.

Therefore, for the purposes of this work, the main parameter to be considered is this classification time. In order to have an insight on the computational costs involved in DPI classification, its operation is briefly presented next. As proposed in [Cascarano, 2009], a DPI classifier can be modeled as composed by 5 processing blocks, as shown in Figure 4.2:

1) The *SessionID Extraction* block extracts the parameters related to the identification of the session a packet belongs to (i.e., IP addresses, transport-level protocol, port numbers).

2) The *Session Lookup* block checks whether the session the packet belongs to is already present in the session table, which is a list of all the active sessions in the network being monitored. This block and the previous one process the packet header only for every incoming packet.

3) The *Pattern Matching* block is devoted to the classification of the packet. This block implements the pattern matching algorithm that looks for the presence of a signature in the application payload. This module handles both the header and the payload of the packet.

4) The *Session Update* block is executed by the DPI classifier only when the Pattern Matching block returns a positive match, in which case it stores the outcome of the classification in the session table.

5) The *Correlated Session* block looks for data that may lead to the identification of a correlated session. This block is executed only in case the application-level protocol may originate correlated sessions (e.g., Session Initiation Protocol (SIP) or FTP).

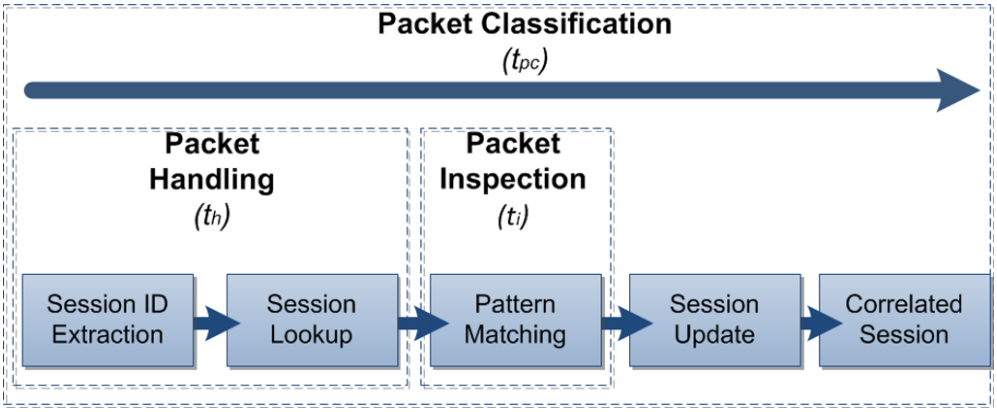


Figure 4.2: DPI packet classification model

According to this model, every packet entering the system is processed at least by both the SessionID Extraction and the Session Lookup modules, independently of the current state of the associated flow as already classified or not. On the contrary, Session Update is executed once per flow, just in case a category is assigned to it, while the Correlated Session module is executed after the classification of the flow.

Therefore, three contributions to the processing time of an individual packet, t_{pc}, can be identified (Fig. 4.2):

$$t_{pc} = t_h + t_i + t_a \tag{4.2}$$

where t_h is the *packet handling* time associated with session identification and lookup (i.e. SessionID Extraction and Session Lookup); t_i is the *packet inspection* time, related to the pattern matching procedures; and t_a is the *annotation* time, related to the storage of the flow classification label and the search for correlated sessions[2]. Thus, as previously mentioned, t_h is always non zero, t_i is incremented while the search for signatures is in process and t_a counts, at most, once per flow.

In the normal usage, t_a can be considered negligible in the classification of a flow when compared with the total contributions from the other two terms. Thus, when no sampling is applied, all packets are analyzed searching for the signatures and the average time for analyzing a flow, t_{fc}, composed by N_p packets can be estimated as:

$$E[t_{fc}] = N_p \cdot E[t_{pc}] \simeq N_p \cdot E[t_h + t_i] \tag{4.3}$$

where $E[]$ is the expected value operator.

It can be easily argued that both the packet handling and packet inspection times are independent magnitudes. In fact, the packet handling time is

[2]In this study, we are concerned in analyzing both t_i and t_h, illustrated in Figure 4.2.

expected to be similar for all the packets, as the processing will be the same. Its major dependence is related to the size of the active sessions list. On the other hand, the packet inspection time will be proportional to the size of the payload and to the number and the length of the signatures to be matched. Therefore, the average flow classification time can be expressed as:

$$E[t_{fc}] = N_p \cdot (E[t_h] + E[t_i]) \tag{4.4}$$

Consequently, the optimization of the DPI procedures can be targeted at reducing the packet handling time, which can be achieved through various means, as explained in Section 4.2, by improving the performance regarding the table lookup and the extraction of the packet headers. Nevertheless, this work is mainly targeted at optimizing the packet inspection time by using sampling methods, that is, by reducing the number of inspected packets and/or the volume of analyzed data per packet.

In fact, according to [Cascarano, 2009], the most significant portion of the flow classification time is related to the pattern matching block, which has a linear dependence on the number of input characters. A deeper insight into this question is addressed in Section 4.3.

For the purposes of this work, only the previously mentioned times will be considered for the comparison of the sampling mechanisms and the derived gains. In this sense, the classification times are considered proportional to the involved computational cost.

4.2 Related Work

Before analyzing each of the proposed sampling schemes, the related works found in the literature are presented in this section in order to place them in context.

First, the choice of DPI tool is validated in [Bujlow, 2014, Bujlow, 2015]. These recent studies present an evaluation of the accuracy of the most well-known DPI-based classifiers (i.e., PACE, OpenDPI, L7-filter, nDPI, Libprotoident and NBAR) and show that PACE is, for the majority of protocols, applications, and web services included in their dataset, the most reliable solution for traffic classification. Among the open-source tools, nDPI and Libprotoident present the best results. Authors provide accuracy per application, thus, the robustness of the evaluation methodology is independent of the applications selected.

Although the study in [Bujlow, 2014, Bujlow, 2015] is complete, the continuous evolution of the network applications and the DPI-based techniques allows a periodical updated of the evaluation. For instance, this evaluation can be updated by adding new applications and web services to the dataset and by introducing new classification tools to the study.

On the other hand, various methods and techniques are described in the literature to improve the computational efficiency of DPI, while maintaining the classification results. Some are hardware-based [Rao, 2010] while others referred to software and algorithm-based enhancements [La Mantia, 2010, Lin, 2008]. Among them, the use of sampling is considered one of the software-based optimizations, as it is supposed to reduce the size of the inputs to the classification procedures.

In general terms, sampling the network traffic is the process of taking partial observations from the monitored traffic, and drawing conclusions about the system behavior from these sampled observations. It is mainly used for network management and monitoring [Carela-Español, 2011], although it can be used as well in classification tasks. In fact, some works [Aceto, 2010, Fernandes, 2009, Ficara, 2010, Guo, 2008, Canini, 2009] integrate sampling within the traffic classification process itself. Nevertheless, choosing a sampler for DPI has proven not to be as straightforward as it seems, as the target is to find the signatures, which can be harder or even impossible depending on the sampled data [Khalife, 2011b].

The overall concept of input reduction is not new to the traffic classification field, as it has been addressed for many classification methods. In this context, it is important to distinguish between DPI and non-DPI classification methods due to the existing differences at both the algorithm and the input type levels.

For instance, in a ML based classification approach [Nguyen, 2008], special feature selection mechanisms are used to minimize the set of input traffic parameters. On the other hand, some non-DPI based works [Hullár, 2011] have similar targets in reducing the classification input size by considering only the initial portions of the flows, which is often referred to as early classification methods.

A detailed taxonomy of sampling techniques for traffic classification according to the used method is provided in [Jurga, 2007]. Another way of categorizing sampling techniques, used throughout this chapter, is related to the target considered by the sampling method. From this point of view, and regardless of the sampling policy, techniques can be categorized into per-packet sampling [Aceto, 2010, Ficara, 2010], or per-flow sampling [Carela-Español, 2011, Jurga, 2007]. It is relevant that the literature shows very few works combining both techniques (e.g. [Fernandes, 2009]).

According to this criteria, the following relevant works can be identified in each category:

- Per-packet sampling: per-packet DPI sampling techniques have been tested with various DPI tools. For instance, using L7-filter [Aceto, 2010], it was shown that 72% of the matchings take place at the first packet of a flow, and that almost more than 90% of matching strings falls within the first 32 bytes of the payload. Another example is shown in [Ficara, 2010], where a novel approach was proposed to bring the sampling idea

to the regular expression field.

- Per-flow sampling: Per-flow DPI sampling [Chen, 2009b] ranged from simple techniques, such as Equidistant Invariable Mode (EIM), which simply involves a fixed sampling rate of full payload packets during the flow lifetime, to more complex ones, such as stochastic sampling.
  In addition, in [Chen, 2009b] six sampling strategies were suggested and tested with DPI identification systems. However, the obtained results were difficult to generalize since they were affected by many factors.
  Other per-flow sampling strategies were shown in separate works. Using sampled NetFlow, it was shown [Carela-Español, 2011] that packet sampling has a severe impact on the classification performance: flow accuracy drops to 85% for a packet sampling rate of 1/100.
  Related sampling (RelSamp) [Lee, 2011], proposes that flows that are parts of the same application session should be given higher probability. Mask-match sampling method [Cong, 2010] reached a 94% of flow accuracy for a packet sampling rate of 1/10. However, this method mainly focuses on long flows, and the validity of the samples is related to the randomness of the ID field of the IP packets headers.
  Moreover, per-flow sampling techniques were also used to localize application signatures. For instance, Canini [Canini, 2009], using L7-filter, sampled the first 10 packets per flow. However, many FPs were encountered due to the probabilistic nature of the used Bloom filters.

- Combination of sampling schemes: As mentioned, the combination of per-flow and per-packet sampling was less addressed in the literature. Using L7-filter, results in [Fernandes, 2009] showed that most flows can be classified only with the first 7 packets or a fraction of their payload without a significant impact on accuracy. Nevertheless, using the same tool in [Cascarano, 2011], the authors were unable to generalize these results, which were extremely dependent on the traffic trace considered. Apparently, none of these methods was able to generalize a sampling methodology for DPI optimization purposes.

Overall, there is a lack of recommended techniques for a DPI oriented sampler and, as previously shown, the current studies are limited and do not present comparative results. Moreover, most existing works do not provide protocol oriented results and are mostly based on L7-filter, which is currently outdated as a DPI classifier. On the other hand, those sampling schemes were proposed regardless of the underlying traffic characteristics, or they analyzed limited size datasets and are, most often, strongly dependent on the DPI tool. In this context, the proposed assessment is of interest, as an up-to-date DPI tool used over a big size real dataset where a comparison of the performance of the sampling methods is also provided.

4.3 Assessment of DPI-oriented Sampling Scheme

Next, each of the sampling policies presented in Section 4.1.2 is described and experimentally evaluated. As previously mentioned, they are presented in the same chronological order as they were tested, as the results from simpler policies are used to guide the experimentation with more complex ones. Namely, per-packet sampling is analyzed first, then, per-flow and combined sampling policies.

Regarding the experimental setup, the DPI-based tool developed for this work, `dpi_flows`, is modified to include the sampling policies and to generate the cost, in terms of computational time together with the classification results[3]. The computational cost is expressed in terms of t_{fc} and t_i, both of which are supposed to be reduced when DPI is applied jointly with sampling. Then, results for each method are compared, both in terms of classification metrics and computational time, with those obtained when no sampling is applied.

The datasets selected for this evaluation include CS-A, CS-A1 and CS-A2 (see Chapter 3). Since they present an unbalanced protocol distribution due to their real nature, experimental results have to be presented on a per-protocol basis. Though experiments were conducted in the multi-classification context, binary classification metrics are evaluated for each protocol.

For the purpose of our experiments, we had to choose an appropriate metric and methodology in order to evaluate the effect of sampling on DPI classification. Most importantly, the problem associated with non-classified objects or unknowns, appearing both at the ground truth (see Chapter 3) and classification results of the tested datasets, had to be handled by our evaluation methodology.

First, at the ground truth level, all non-classified flows are omitted from counting. As mentioned in Chapter 3, this is a common approach followed by most classification works in the literature to ensure 100% correct validation results. In addition, as we are comparing the sampled DPI classifier against the reference DPI classifier in full payload inspection mode (i.e. with no sampling), eliminating unknowns will permit to highlight on the number of flows which become unknown particularly as a result of the applied sampling, not due to any other reason[4].

On the other hand, unknown objects might appear during the classification phase. The number of unclassified objects is problematic from a research point of view since these cannot be counted when measuring any of the four basic metrics (TP, TN, FP and FN defined in Chapter 3). In most of literature works, an underlying assumption is that during classification, each object has to be either classified or misclassified, therefore, unknowns are not regarded

[3]Some details regarding this adaptation can be found in Appendix A.

[4]Reasons for which DPI is unable to classify flow objects in full payload inspection mode are discussed in Chapter 3.

Table 4.1: Confusion matrix for sampling-based DPI classification decisions

$$\begin{array}{rc} Assigned \backslash Actual & \begin{array}{cc} L_i & non-L_i \end{array} \\ \begin{array}{r} L_i \\ Non-L_i \\ Unknown \end{array} & \left(\begin{array}{cc} TP_i & FP_i \\ FN_i & TN_i \\ UNK_i & UNK_{ni} \end{array} \right) \end{array}$$

when evaluating existing classifiers result.

Nevertheless, and although completely eliminated from the ground truth results, unknown flows still appeared during the sampling-based DPI experiments that have been conducted throughout this chapter[5]. In fact, these flows became unclassified particularly due to payload sampling. Therefore, they had to be considered during the evaluation process since their ratio is very insightful for studying the effect of sampling on DPI, which is the main purpose of the study conducted in this chapter. Actually, unclassified flows are the main detectable effect of applying sampling to DPI.

Formally, the confusion matrix in Table 4.1 shows all of the possible classifications in a binary classification mode for a given protocol L_i, after sampling is applied to DPI. TP_i, FP_i, FN_i, and TN_i are, respectively, the number of TP, FP, FN, and TN associated with protocol L_i. UNK_i is the number of flows labeled with L_i that become unclassified after sampling is applied, and UNK_{ni} is the number of $non-L_i$ flows that became unclassified, after sampling is applied.

It should be noted in this table that there is no *Unknown* in the actual class column since, as mentioned previously, all unknown instances were removed for the ground truth results. Nevertheless, when sampling is applied, a flow belonging to protocol L_i might not be classified (counted as UNK_i) or might retain its class label at the ground truth (counted as TP_i). Therefore, and for the purposes of our experiments, the Detection Percentage (DP) metric has been defined and customized to include UNK_i in order to measure the effect of sampling as follows:

$$DP_i = \frac{TP_i}{TP_i + UNK_i} \tag{4.5}$$

where DP_i is the percentage of flows[6] that can be still detected as class L_i even after sampling is applied, TP_i and UNK_i are, respectively, the number of TP

[5]Sampling results are shown in sections 4.3.1, 4.3.2 and 4.3.4.

[6]If UNK_i is regarded as FN_i, then, DP_i is equivalent to the class recall, for $FP_i = 0$.

and the number of unknown flows, associated with class L_i after sampling is applied.

The defined DP values are then measured in order to evaluate the effect of the different sampling policies tested throughout this chapter, as detailed next.

4.3.1 Per-packet based Sampling

Per-packet, also referred to as *payload based* sampling, attempts to reduce the computational cost by partially inspecting the payloads of each of the packets in the flow. Although in general terms it is possible to consider any portion of the payloads, it seems reasonable to limit the sampling to analyze only the first bytes of each payload when the nature of the communication protocols is considered, as the commands and protocol headers will appear at the beginning. In this later case, the sampling scheme is also referred to as *packet truncation*, which is the method that will be analyzed in this subsection.

Thus, in the packet truncation sampling policy, the classifier only parses a specified number of initial bytes, the *payload truncation length*, denoted as S, within each packet payload. This policy can be simply defined as the *"first S bytes per packet"*.

The new overall time to classify a flow, t'_{fc}, can be estimated, in this case, as:

$$E[t'_{fc}] = E[N_p] \cdot E[t_h] + E[N_p] \cdot E[t'_i(S)] \tag{4.6}$$

being $t'_i(S)$ the inspection time of the first S bytes instead of the full packet payload. The cost of the pattern matching for S bytes is expected to be lower than that for the whole payload due to its dependence on the length of the analyzed pattern[7]. Therefore, $t'_i(S) \leq t_i$.

On the other hand, no reduction is to appear from the other terms in Equation 4.6, as the number of inspected packets will remain the same as before applying the sampling policy, and this partial inspection does not affect t_h since each packet should be handled by the classifier to determine the flow it belongs to.

4.3.1.1 Experimental Results

The packet truncation method has been experimentally tested using an adapted version of the DPI tool (`dpi_flows`) over the CS-A dataset (see Chapter 3). Specifically, the tool has been further customized to be able to parse only a specified number of bytes (called the *truncation length*), within each packet's payload by adding packet sampling modules.

[7]It is possible for some payloads to be shorter than S bytes. Obviously, in this cases the inspection time will remain the same with or without truncation. The impact of this effect will depend on the number of packets shorter than S, which will be experimentally addressed.

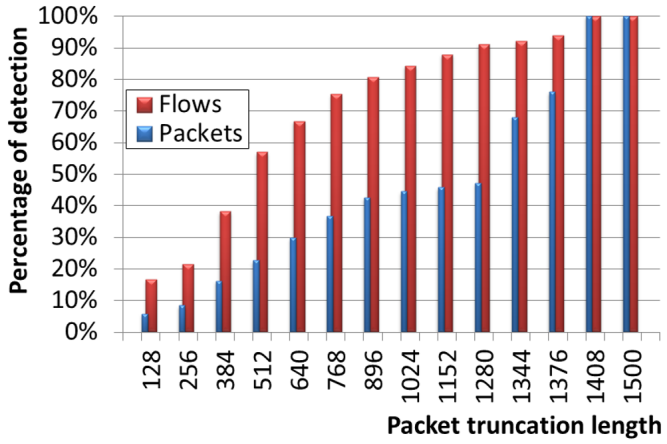


Figure 4.3: Average detection percentage results, DP, as a function of the truncation length of the payloads (per-packet sampling)

In the following, flow and packet DP values are shown as a function of the payload truncation length, S. However, as stated earlier, most of the analysis is focused on flow-based results. On the other hand, the results are assessed at three levels: per protocol, per protocol group [8] and for all the protocols.

The results obtained for all protocols in the CS-A dataset are depicted in Figure 4.3. As shown in this figure, the truncation length must be at least 1280 bytes to reach 50% of packet classification. This result is not very encouraging for DPI optimization. Through payload truncation, a 15% reduction in the payload input size leads to a 50% drop in the DPI packet classification. Although the situation is better at flow level, it is still a very poor result, as only a 80% of correct classification is achieved using $S = 1280$ bytes.

On the other hand, this is a surprising result, as it is one of the most used and analyzed sampling methods. Probably, this is related to the use of NetFlow as the source for the input data and the availability of some public datasets with this kind of truncated data. Furthermore, as previously mentioned, is is expectable to match the signatures for most of the protocols at the beginning of the payloads. Nevertheless, the experimental results evidences that it is necessary to analyze bigger chunks of the payloads to reach a reasonable point of operation.

A common behavior observed for all protocols (Figure 4.3) is that the number of detected packets and flows increases with the truncation length. However, DPI shows different behaviors per protocol group, as shown in Figure 4.4, and per individual protocol, as shown in Figure 4.5.

For example, for Internet Messaging (IM) –Figure 4.4.b– and DNS –Figure 4.4.c– DP is less affected by per-packet sampling when compared to the web

[8] Protocol groups have been defined according to [Mochalski, 2009].

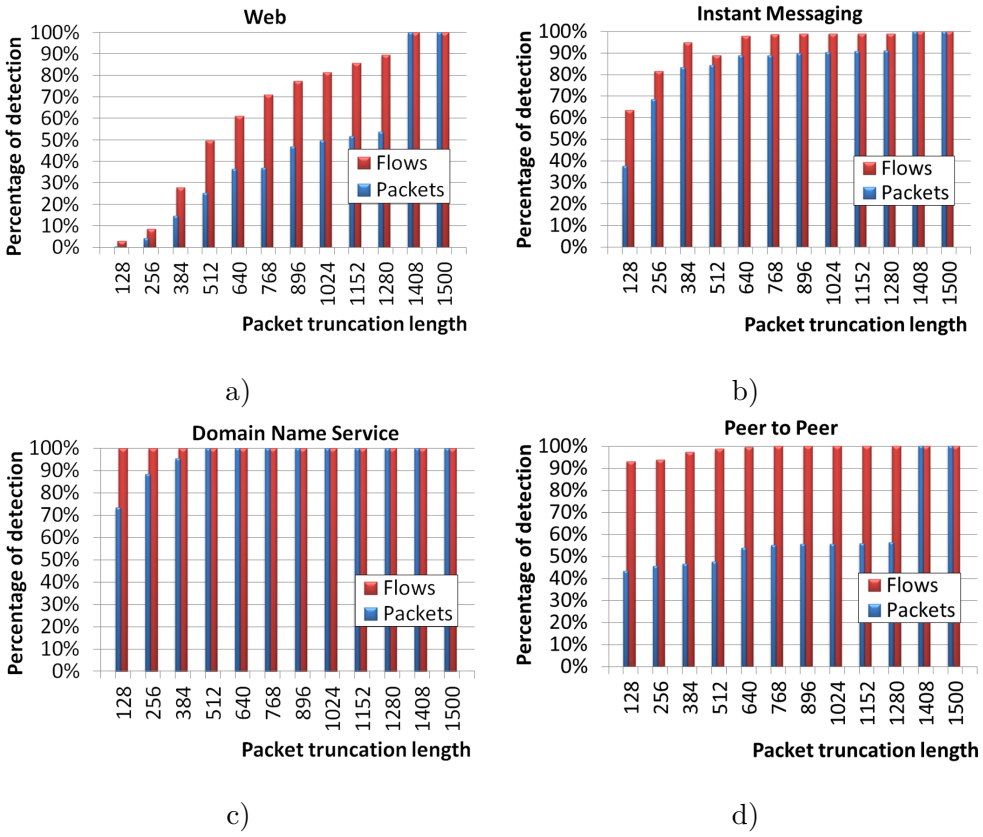


Figure 4.4: Per-packet sampling detection percentage results for various protocol groups as a function of the truncation length for packets and flows: a) Web, b) Instant messaging, c) DNS and d) P2P

protocol group, shown in Figure 4.4.a. Another example is that of P2P protocols –Figure 4.4.d– which exhibit a mixed behavior: P2P results are similar to that of the web group, at packet level, and to IM and DNS groups, at flow level.

At this stage, studying eventual computational gain with the per-packet sampling is considered of less significance since this scheme is unable to maintain the DP. Consequently, DPI is assessed next with per-flow sampling, which is supposed to provide a coarser grained view on the locations of the classification bytes.

4.3.2 Per-flow Based Sampling

The target of per-flow sampling is to inspect only a subset of the packets in each flow. In its normal operation mode, DPI inspects all the packets in a flow. Nevertheless, for flow classification, and assuming that all the packets in

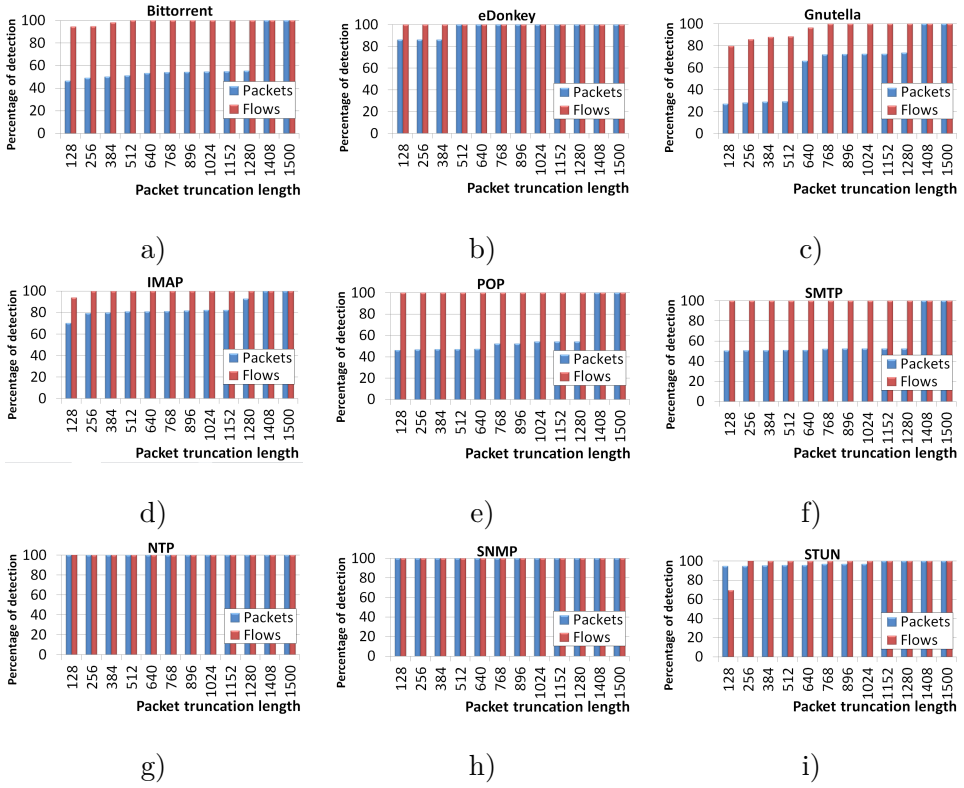


Figure 4.5: Per-packet sampling detection percentage results for various protocols as a function of the packet truncation length: a) BitTorrent, b) eDonkey, c) Gnutella, d) Mail_IMAP, e) Mail_POP, f) Mail_SMTP, g) NTP, h) SNMP and i) STUN

a flow should correspond to the same protocol, it is not necessary to analyze packets beyond the first classified one. Thus, an obvious question is whether the previous operation mode should be modified to reduce the number of analyzed packets and, therefore, the computational cost, by stoping the analysis once a packet is classified.

Therefore, a feasible sampling policy is to inspect packets only till one of them is classified. The ordinal of the packet at which the first detection inside the flow is achieved is referred to in the following as the *flow detection number*.

A first potential problem in this scenario is related to the possibility of a misclassification of this first classified packet, which will produce an error in the flow classification. On the other hand, this sampling policy is useless for those flows that remain unclassified after being closed. Both questions may affect the performance of the system and will be addressed through experimentation.

In the proposed *flow truncation* sampling policy [Khalife, 2011a], the classifier only inspects the payloads of a specified number of initial packets, the *flow*

truncation length, N_{min}, in each flow. This policy can be summarized as *"first N_{min} packets per flow"*. Thus, the targets of the experiments conducted in this subsection are to determine the optimal value for N_{min} and the associated computational cost.

Per-flow sampling is supposed to decrease the overall classification time for the whole traffic. When per-flow sampling is applied, assuming $E[N_p] > N_{min}$[9], the average time required for analyzing a flow, t''_{fc}, can be estimated as:

$$E[t''_{fc}] = E[N_p] \cdot E[t_h] + N_{min} \cdot E[t_i] \tag{4.7}$$

In this case, there is a potential reduction in the size of the analyzed payloads and, consequently, in the total inspection time per flow, due to the inspection of a lower number of packets per flow, not to a reduction of the inspection time per packet.

It is important to note again that with this sampling mode, beside of inspecting only the first N_{min} packets of the flow for the purpose of classification, the classifier still needs to handle all the remaining packets. For these packets, t_h cannot be avoided since it is evidently impossible to know if a given packet belongs to an existing flow without parsing its header at least. The decision to inspect the packet or not depends on its ordinal number being lower or higher than N_{min}.

Therefore, the time to classify a flow, t''_{fc} is expected to be significantly lower than that required for inspecting the full flows when no sampling is applied, t_{fc} –Equation 4.4–. Experimental results associated with per-flow sampling are detailed next.

4.3.2.1 Experimental Results

As with the packet truncation policy, the flow truncation method has been experimentally tested using an specifically modified version of the DPI tool (`dpi_flows`) over the CS-A and CS-A1 datasets (see Chapter 3), to evaluate DP together with the computational costs.

In this regard, the tool has been customized to operate in two different modes. In the first one, it shows the flow detection number and measures the accumulated handling and inspection times for each flow, stoping the inspection after it is classified. In the second mode of operation, the tool solely inspects, within each flow, packets whose ordinal number inside the flow is lower than a predefined threshold (N_{min}), also providing the accumulated times. In both cases, all the packets in a flow have to be handled in order for them to be assigned to its flow.

Similarly to the previous subsection, flow DP results are extracted as a function of the number of sampled packets (the *flow detection number*) from

[9]In a similar way as in the case of per-packet sampling, it is possible for some flows to be shorter than N_{min} packets. The impact of those flows is expected to be low for low enough values of N_{min}. Anyway, this will be experimentally evaluated.

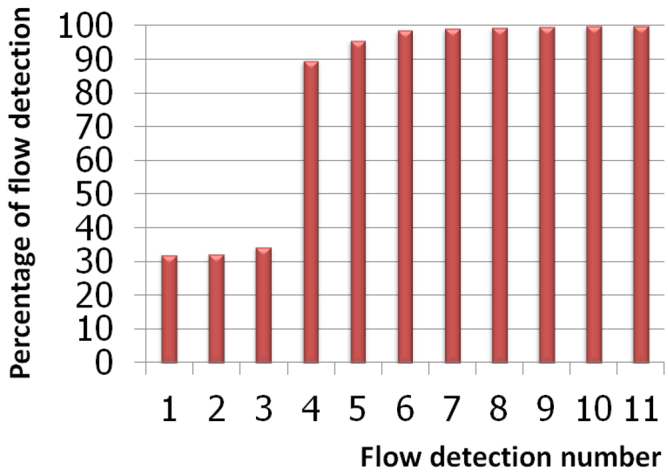


Figure 4.6: Results for flow detection percentage as a function of the flow detection number

the beginning of the flow. Two types of experiments are addressed, according to both modes of operation.

In the first one, the flow is analyzed till it is classified or it ends. In this case, the results provide an insight on the number of packets needed for classifying a flow. Obviously, there is not an upper limit on the number of analyzed packets in a given flow, which obviously leads to a suboptimal behavior for unclassified flows.

To avoid this situation, in the second mode, a threshold for the number of inspected packets is set, which corresponds to the proposed sampling policy. The reasoning behind this is based on the hypothesis that the probability of a flow to be classified decreases after some initial packets. Thus, the experiments are targeted at verifying whether most of the flows can be classified after a reduced number of packets are observed.

The firsts experiments were made by varying the threshold, N_{min}, for the maximum number of inspected packets. The DP results for all the protocols as a function of the *flow detection number* are depicted in Figure 4.6. As shown, most of the flows from all protocols are classified within the first ten packets with 99.9% of flow DP. For detection number greater than 10, the resulting DP ratio is negligible and is not shown in figures.

It can be also noticed from Figure 4.6 the very slight increase in DP for N_{min} values greater than 10. Therefore, there is a big number of flows for which the detection is achieved with just a few packets. Proportionally, the number of flows with high *flow detection number* is almost negligible.

Next, the results are analyzed on a per-protocol group basis to check whether this behavior is consistent across all of them. The most significant results are shown in Figure 4.7. As shown, some protocol groups requires even

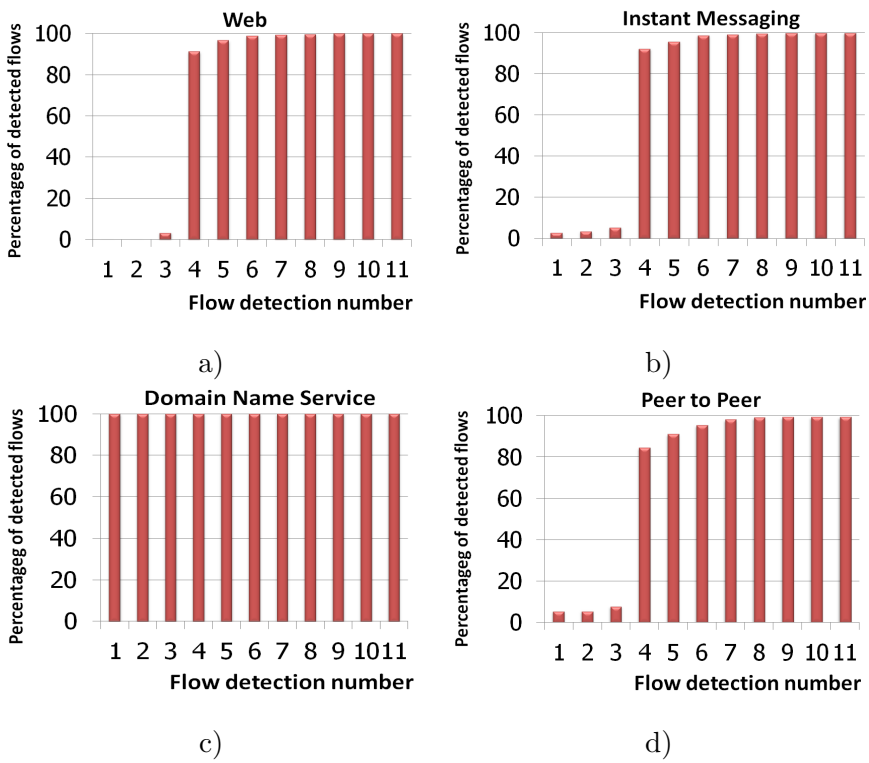


Figure 4.7: Flow detection percentage results for various protocol groups as a function of the flow detection number: a) Web, b) Instant messaging, c) DNS and d) P2P

less than 10 packets to be detected. This is specially relevant in the case of DNS group (Figure 4.7.c) which is detected from the first packet. Therefore, overall, the proposed sampling method seems to be quite effective without degrading the classification performance of the system.

Anyway, to have a deeper insight into the behavior of the flow truncation method, some additional experiments were made in the first of the operational modes using the CS-A dataset. This way, the results provide a distribution of the *flow detection number* for all the flows. Figure 4.8 shows the obtained results grouped according to P2P or non-P2P protocols. From this histogram, it is clear that most of the flows are detected within the first few packets. Nevertheless, a more detailed analysis on a per-protocol basis reveals some differences in this behavior.

As shown in Figure 4.9, most protocols average the *flow detection number* clearly below 10. The associated results per protocol (Figure 4.10) are similar to the globally obtained results on a per individual protocol basis.

According to Figure 4.9, and in order to classify most protocols, the selected value for N_{min} is 10. Although for protocols whose average flow detection

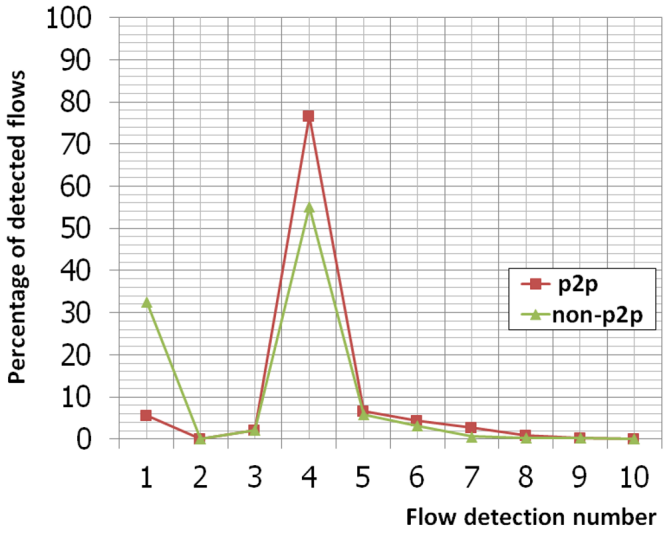


Figure 4.8: Histogram showing the number of detected P2P and non-P2P flows as a function of the flow detection number for flow truncation

number is lower than N_{min} the classifier would inspect more packets than necessary, choosing this value is supposed to guarantee that all protocols can be detected.

Thus, as a result from the experiments, it seems reasonable to recommend a value of at least 10 for N_{min}.

On the other hand, few but exceptional cases of lately or not classified flows as well as classification errors were reported during the experiments. Due to their reduced volume, these cases had no significant impact on the presented results, but it is worth to analyze them in greater detail.

One of the observed exceptions, together with their preliminary interpretations, are deviator flows. In fact, there exist flows for which the *flow detection number* seems not to belong to the associated distribution, that is, outlier flows. After a preliminary exploration of these outliers, a possible relationship emerged. When the first packet holding the application signature is lost or not captured, the flow detection number becomes much deviated from the calculated average. Obviously, this situation is more likely to occur at the beginning of the capture. In fact, it has been discovered that most deviators are flows that were under course at the start of the capture. As an example, Figure 4.11 shows that most of the deviator HTTP flows (up to a 98%) present a low `FlowID` (below 100,000). As explained in Chapter 3, the `FlowID` parameter is the sequence number of the flow within the capture, which means that they are at the beginning of the capture and that there is a high probability of missing the firsts packets of the flow. Again, this result highlights on the importance of the first flow packets to DPI classification.

In summary, the results for per-flow sampling shows that inspecting the

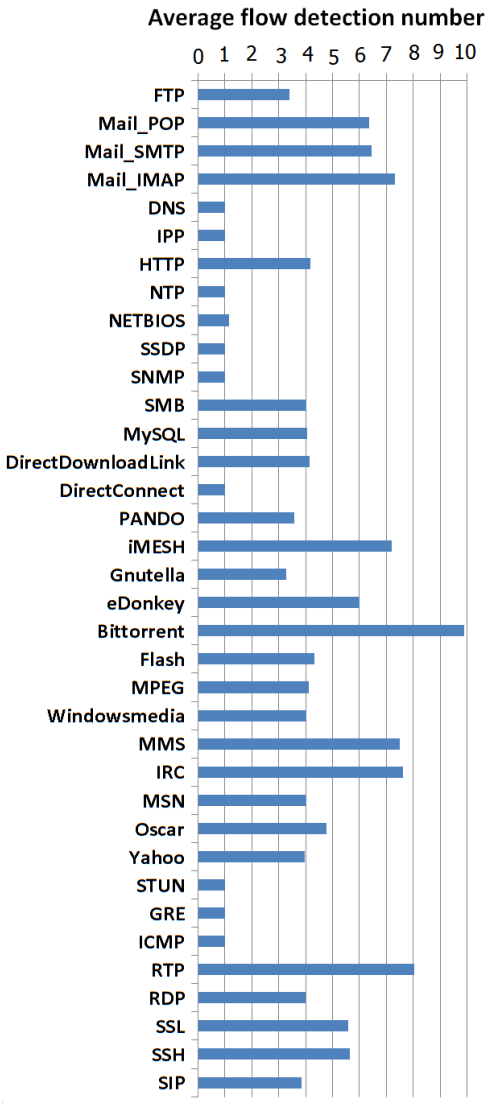


Figure 4.9: Average flow detection number for each individual protocol (per-flow sampling)

first 4 to 10 packets of a flow can maintain the flow DP at high levels, ranging from 90% to 99%.

4.3.3 Comparison of Packet and Flow Truncation Sampling Policies

At this stage, both the per-packet and per-flow sampling are presented. Before assessing additional sampling schemes, a comparison of the results and an analysis of the computational costs is relevant to highlight main findings related

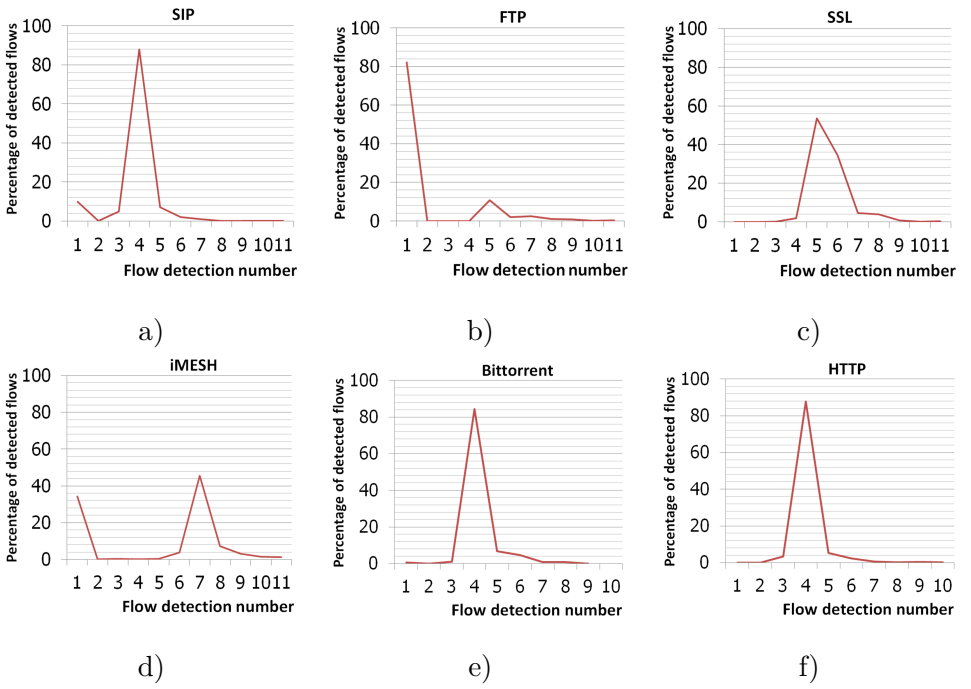


Figure 4.10: Percentage of detected flows for selected protocols as a function of the *flow detection number*: a) SIP, b) FTP, c) SSL, d) iMESH e) BitTorrent, and f) HTTP

to DPI optimization with sampling.

As previously stated, the computational cost is evaluated through the time it takes to classify a flow. As per Equation 4.4, the average flow classification time depends on the total inspection time which in turn depends on both the average number of packets per flow and the average packet size.

Obviously, the higher the number of flows with a number of packets $N_p > N_{min}$, the higher is the expected time gain, and consequently, the effectiveness of the per-flow optimization. Similarly, the higher the percentage of packets with size over 1408 bytes, the more efficient will be packet truncation. Specifically, For the CS-A dataset, the packet payload size can be estimated by 1350 bytes for CS-A and the average packet number per flow by 30,000 bytes which is equivalent to around 22 packets (see Chapter 3).

From this data, and in order to achieve a DP level of 99%, a preliminary estimate on the volume of data that should be analyzed per flow can be obtained for both cases: it is necessary to inspect up to $13,500 = 1350 * 10$ bytes in the per-flow compared to $30,000 = 1350 * 22$ bytes in the per-packet case. As per Equation 4.4, the packet inspection time will be proportional to the size of the payload.

Consequently, the per-flow sampling scheme should theoretically outper-

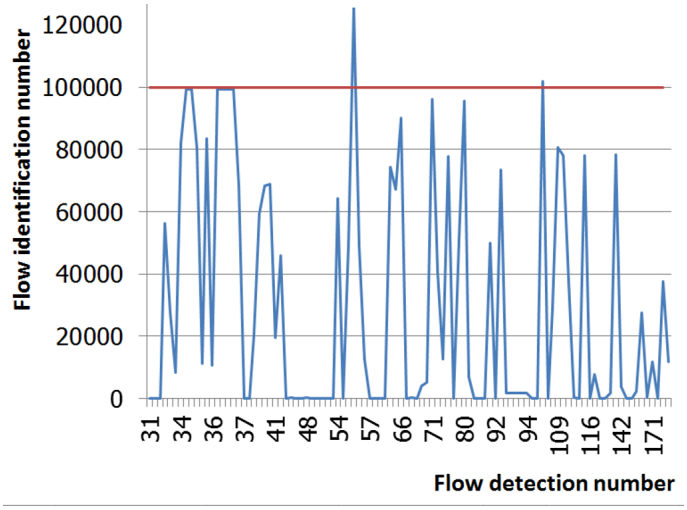
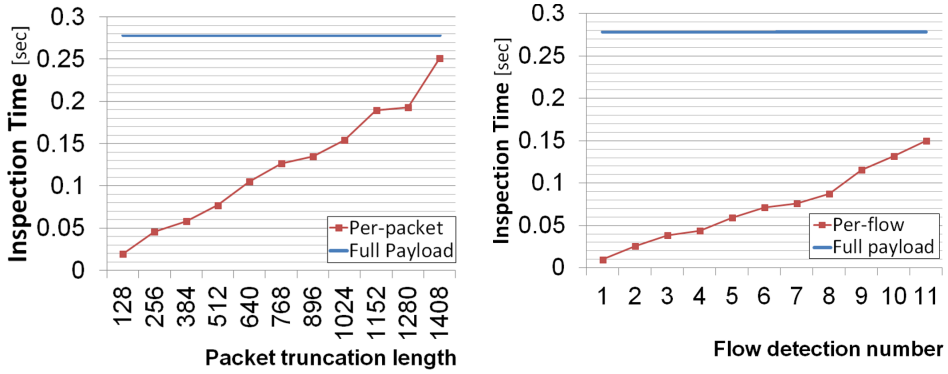


Figure 4.11: Flow identification number (Flow_ID) v.s. flow detection number (HTTP deviators for $N_{min} > 30$)



a) b)

Figure 4.12: Total inspection time t_i for dataset CS-A1 as a function of: a) the packet truncation length (per-packet sampling), b) the flow detection number(per-flow sampling), v.s. full payload

form the per-packet scheme. To prove this assertion, the computational cost had to be experimentally measured. From an operational point of view, the processing time consumed by the inspection and classification modules were measured in both cases, per-flow and per-packet sampling, by running the classifier over the CS-A1 dataset.

Figure 4.12 shows the total inspection time, t_i, for dataset CS-A1 both when applying the per-packet and per-flow sampling compared to the full payload case.

Table 4.2: Comparison between best case scenario results for per-packet and per-flow sampling methods

DPI Experiment	**Payload**	**DP**	**Total $t_{fc}(\mu s)$**	**Total t_i (μs)**	**Total t_{fc} gain**	**Total t_i gain**
No sampling	FULL	100%	2090666	278222	-	-
Per-packet sampling	1408B per packet	99%	2058212	245769	1.5 %	11.6 %
Per-flow sampling	10 Packets per flow	99%	1944064	131621	7.01 %	52.7 %

As shown in Figure 4.12a), the inspection time increases with the inspected payload size. Similarly to per-packet sampling, the inspection time associated with per-flow sampling shown in Figure 4.12b) increases with the flow detection number, that is, when increasing the inspected payload size. In both cases, the total inspection time t_i is inferior to the time consumed when inspecting the full traffic payload, that is, when no sampling is applied.

Most importantly, it can be clearly seen that more inspection time can be saved with per-flow sampling, especially for high accuracy values. However, saving in inspection time should not be considered regardless of DP values. As seen previously, to achieve a DP value of 99%, the first 10 packets with full payload are required, per flow, while the first 1408 bytes of payload are required per packet in each flow. Considering the points of Figure 4.12a), where *packet truncation length* equals 1408, and of Figure 4.12b), where *flow detection number* equals 10, these two points represent respectively the best case scenarios ($DP \geq 99\%$) for the per-packet and per-flow sampling schemes where the inspection time is optimized while DP values are maintained.

Obviously, less inspection time is required in the best case scenario for per-flow sampling, which is more likely the case with the classification time. For this reason, the total classification time for all flows in dataset CS-A1 using flow sampling, t''_{fc}, –Equation 4.7–, and using the packet sampling t'_{fc}, –Equation 4.6–, and the original one, t_{fc}, –Equation 4.4– were measured for the best case scenarios of each sampling scheme. The obtained results are shown in Table 4.2[10].

Consequently, at a DP level of 99%, the DPI classification time gain obtained with the proposed per-flow sampler reached 7.01 % while it can barely

[10]The handling time, t_h, is included in the evaluated classification time.

reach 1.5 % with the per-packet sampler.

Important implications that are relevant for the choice of the optimal DPI oriented sampler can be inferred throughout this comparison:

(i) The classification bytes have proven to be regularly found within the beginning of each flow, specifically, within the first 10 packets.

(ii) Per-flow sampling proved to be more convenient for DPI classification than per-packet sampling. It outperformed per-packet sampling in terms of providing higher DP, at the cost of less input inspection and reduced inspection and classification times.

As a result, the optimal DPI sampler should thus include a per-flow based sampling module focusing on the first flow packets. At this level, the combination of both sampling techniques will be explored to further minimize the sample size, which is discussed next.

4.3.4 Combined Sampling Schemes

Combining per-flow and per-packet sampling aims to reach an optimized DPI sampling scheme where the inspected payload size is further minimized. In this case, two different inspection approaches are considered when combining sampling schemes, where a subset of the packets are partially inspected per flow. Namely, these approaches are non-contiguous and contiguous inspection (see Section 4.1.2), referred to as, respectively, Method-I and Method-II.

In this subsection, both of these approaches will be detailed in the order they were proposed and assessed. At the end of this section, a comparison of all the analyzed approaches will be also provided.

The sampling policy to be used in both approaches is described as follows:

(I) Non-contiguous payload sampling (Method-I): In this mode, the payloads of some selected packets in a flow are partially sampled and inspected. Specifically, in the analyzed sampling policy, the p first packets are sampled per flow, from which the b initial bytes per packet are considered.

Therefore, this method is the straightforward combination of the per-flow and per-packet policies previously assessed[11].

As per the analogy presented in Section 4.1, the sampled payload chunks will be non-contiguous since they belong to disparate packets. The inspected payload size, I, will be:

$$I = p * b \tag{4.8}$$

[11] Per-packet (Section 4.3.1) is a particular case of non-contiguous payload sampling for p equals to the number of all packets in the flow.

As such, the non-contiguous payload sampling scheme in Method I can be defined as: *"first b bytes from each of the p first packets per flow"*

(II) Contiguous payload sampling (Method-II): In this case, the information considered for the inspection are the c initial bytes of the flow. Therefore, the basic unit for the input data is the flow itself, as opposite to the previous methods in which the packets are the basic elements to be sampled. Thus, the cumulative size of the continuously sampled payload is counted, starting from the beginning of a flow, i.e., the first byte of the first packet payload in the flow, till it reaches the established value, c, independently of the number of processed packets[12].

As per the analogy presented in 4.1, the sampled payload chunks are contiguous, as single packets will not be subject to any payload sampling. The only exception to this is the last sampled packet, whose payload might be partially sampled and inspected.

In this mode, the inspected payload size, I', will obviously have the same value of c. If we differentiate between both directions in the flow, it can be expressed as:

$$I = c = c_u + c_d \tag{4.9}$$

being c_u the number of contiguous bytes sampled in the up direction, and c_d the number of contiguous bytes sampled in the down direction[13]. This differentiation is supposed to highlight on the flow direction that is contributing the most in the classification process, which is particularly relevant for those cases where the classifier is unable to capture traffic in both directions.

As such, the contiguous payload sampling scheme can be summarized as: *"c initial bytes per flow"*.

Combined per-flow and per-packet sampling is supposed to decrease the overall classification time for the whole traffic. When combined sampling is applied (whether in contiguous or non-contiguous modes), the average time required for analyzing a flow, t'''_{fc}, can be estimated a:

$$E[t'''_{fc}] = E[N_p] \cdot E[t_h] + E[t'''_i] \tag{4.10}$$

where t'''_i is the inspection time of I bytes per flow instead of the full payload in the flow. It is supposed that $(t'''_i < N_p \cdot t_i)$.

[12] Per-flow (Section 4.3.2) is a particular case of contiguous payload sampling for c equal to the cumulative size of the first N_{min} packets in the flow.

[13] In this context, the up direction in UDP flows is assumed in the direction from the host originating the first packet to the receiving host.

Consequently, and due to inspecting less payload data, the time for inspecting the total payload samples, t'''_{fc}, should be theoretically lower that the time required for inspecting the full payload, t_{fc}) –Equation 4.4–, that is, when no sampling is applied.

In Section 4.3.3, experimental results have shown that the per-flow sampling scheme can outperform per-packet sampling. Since non-contiguous and contiguous sampling methods can be regarded, respectively, as particular cases of per-packet and per-flow sampling schemes, it can be inferred that contiguous sampling method should outperform non-contiguous sampling. However, this assertion has to be validated through experimentation. For this reason, both methods will be experimentally evaluated starting with Method-I.

4.3.4.1 Non-Contiguous Sampling Results (Method I)

In order to apply and evaluate non-contiguous sampling, the experimental setup (see Chapter 3) is customized by conveniently modifying the `dpi_flows` tool including specific modules involving conditions on the inspected payload size and its location within a flow. In this assessment part, the CS-A2 dataset is used.

Flow DP values are shown as a function of the inspected payload length, I, in Figures 4.13 through 4.16. DP results are shown for different values of the sampling parameters (p, b). In order to show both the inspected payload size in bytes and the corresponding sampling parameters, the $p \times b = I$ notation is used for the horizontal axis label, in two dimensional graphs. Moreover, for illustrative purposes, most relevant results are shown in a three dimensional graph (Figure 4.14).

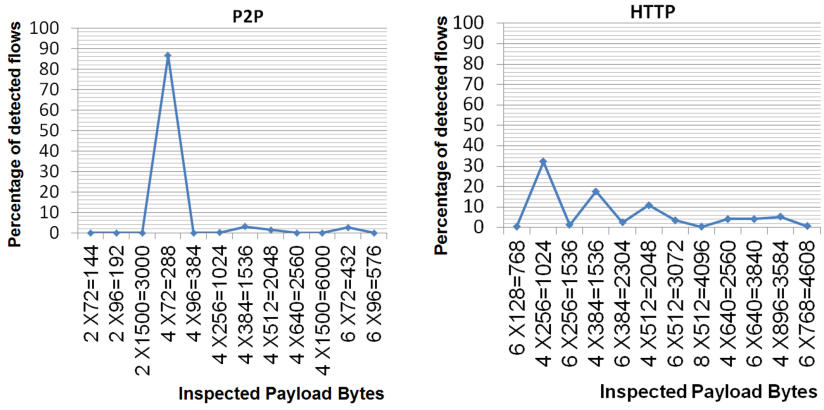
The graphs in Figures 4.13 and 4.16 show the percentage of detected flows for each *(p,b)* pair on the horizontal axis. Figures 4.13 and 4.16 are shown as histograms which helps in better illustrating critical values of (p,b) at which most of the flows are being classified for each individual protocol.

The most relevant observation from Figures 4.13 and 4.16 is that the location of the *classification bytes* is strongly protocol dependent.

In Figure 4.14, DP is averaged for all protocols as a function of the inspected payload bytes. As depicted in Figure 4.14.b, 91.50% of DP can be reached when the inspected payload size is $I = 8 \times 1152 = 9216$ bytes. Nevertheless, averaging the DP for all protocols will be dominated by the ones having the highest contributions in the dataset. In the case of CS-A2 (see Chapter 3), DNS, HTTP and BitTorrent are the dominant ones.

With the unbalanced protocol distribution, biased results are most likely to be obtained. For this reason, and in order to generalize the assumptions about the location of the *classification bytes*, DP results are shown on a per protocol basis in Figures 4.15 and 4.16.

Figure 4.16 shows DP values for various protocols as a function of the inspected payload length.



a) b)

Figure 4.13: Detection percentage as a function of the inspected payload length using non-contiguous sampling for: a) P2P and b) HTTP

Figure 4.15 shows averaged DP values for each individual protocol. Specifically, Figure 4.15 shows the required inspected payload length for DP values above 90%. As shown, and according to `dpi_flows` annotation, SSL is the protocol with the highest requirements in the CS-A2 dataset, with $I = 11,264$ bytes ($\approx$ 11 KB).

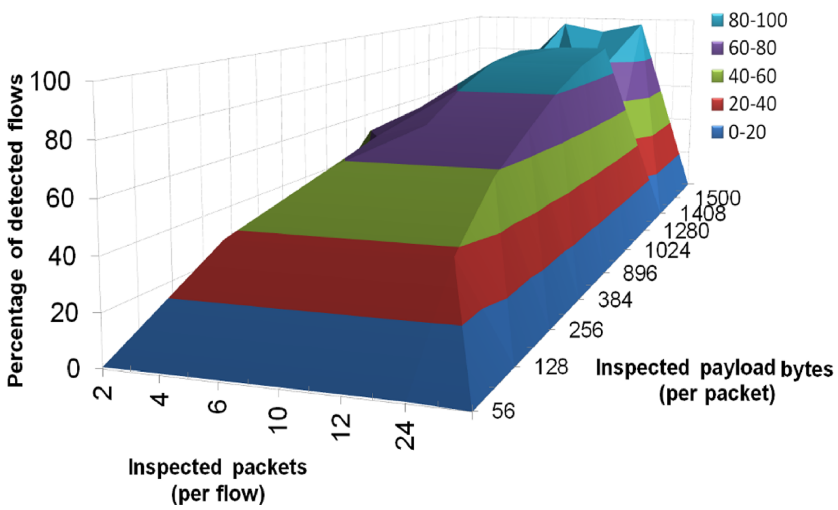
As a result, sampling in non-contiguous mode showed that, according to the used testbed, the *classification bytes* are located within *the first 11 KB of the flows* for most protocols. Therefore, Method-I results validates again the matching of the classification bytes at regular locations, specifically, within the 11 KB at the beginning of each flow.

One of the weaknesses of this sampling method is that the first packets are still subject to partial inspection. Consequently, by further minimizing the sampled payload size without truncating the first packets of a flow, the classification results should eventually be enhanced. Therefore, DPI is assessed next with contiguous sampling mode.

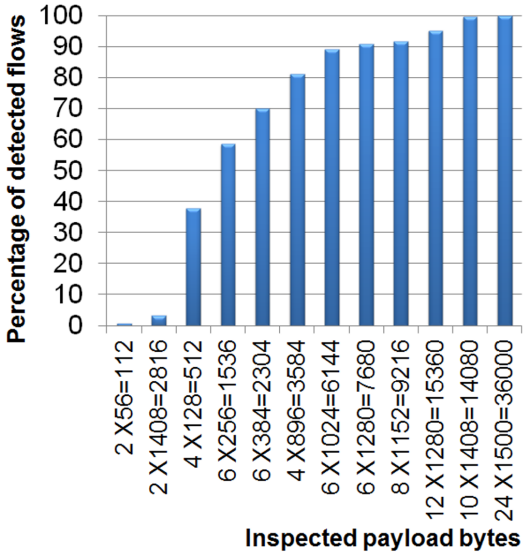
4.3.4.2 Contiguous Sampling Results (Method II)

The main purpose of the analysis shown in this subsection is to check whether it is possible to obtain similar or better DP results, as compared to Method-I, by inspecting less payload, that is, by further improving the sampling policy.

For this reason, the testbed customization used for this section consists mainly on applying the contiguous sampling policy to the `dpi_flows` tool. The obtained results for flow DP are shown as a function of the inspected payload length in Figures 4.17 through 4.19. Results are displayed for different values of the sampling parameters both in the upload and download directions,



a)



b)

Figure 4.14: Detection percentage as a function of the inspected payload length using non-contiguous sampling for all protocols: a) three dimensional, b) two dimensional

(C_u, C_d). In order to show both the inspected payload size in bytes and the corresponding sampling parameters, the $(C_u + C_d) = I$ notation is used for the horizontal axis label, in two dimensional graphs. Additionally, some of the results are shown in three dimensional graph (Figure 4.19).

Figures 4.17 and 4.18 are shown as histograms, which helps in better illus-

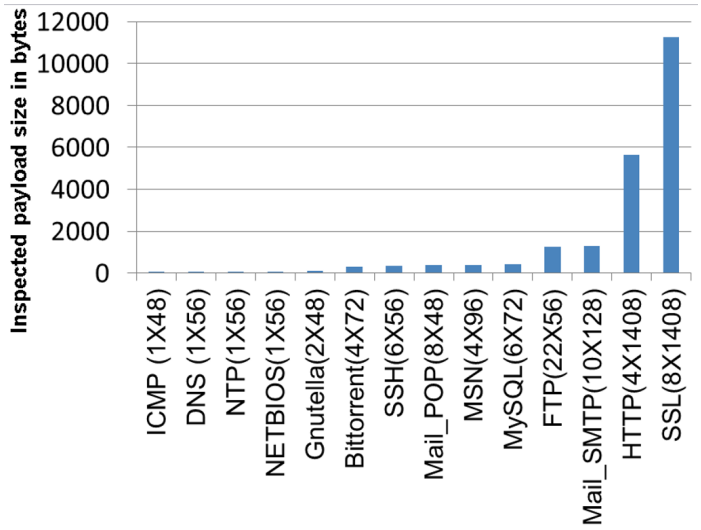


Figure 4.15: Non-contiguous sampling: Payload size to be inspected per protocol for DP above 90%

trating critical values of (C_u, C_d) at which most of the flows are being classified for each individual protocol. Similarly, Figure 4.19 shows the DP as a function of the inspected payload length, averaged for all protocols[14]. As depicted in Figure 4.19, a 90% of DP can be reached when the first *I=1024* bytes in a flow are sampled. Figure 4.20 shows the required inspected payload length for obtaining DP values above 90% for most relevant protocols.

Again, as in Method-I, SSL is the protocol with the highest requirements, with an average of 2,848 bytes ($\approx$ 3 KB) per flow. The first observation from the experiments is similar to that noted for Method-I: the location of the *classification bytes* is protocol dependent, as shown in Figures 4.17 and 4.18.

Using Method-II, the obtained experimental results showed that the *classification bytes* are located within *the first 3 KB of a flow* for most protocols. Therefore, this method validates again the hypothesis that the *classification bytes* are found at regular locations within the flows. Furthermore, in this case, the amount of data to be analyzed is smaller (1/4 ratio) than with Method-I.

Up to this level, a comparison of the obtained results, as shown next, is required to show which method locates the classification bytes accurately and should thereby, lead to the optimal DPI sampler.

4.4 Comparison of Combined Sampling Policies

In a preliminary evaluation, it seems clear that contiguous sampling is preferable, compared to non-contiguous sampling, when the volume of data to be

[14]Except for DNS, whose results are separately shown in Figure 4.18.

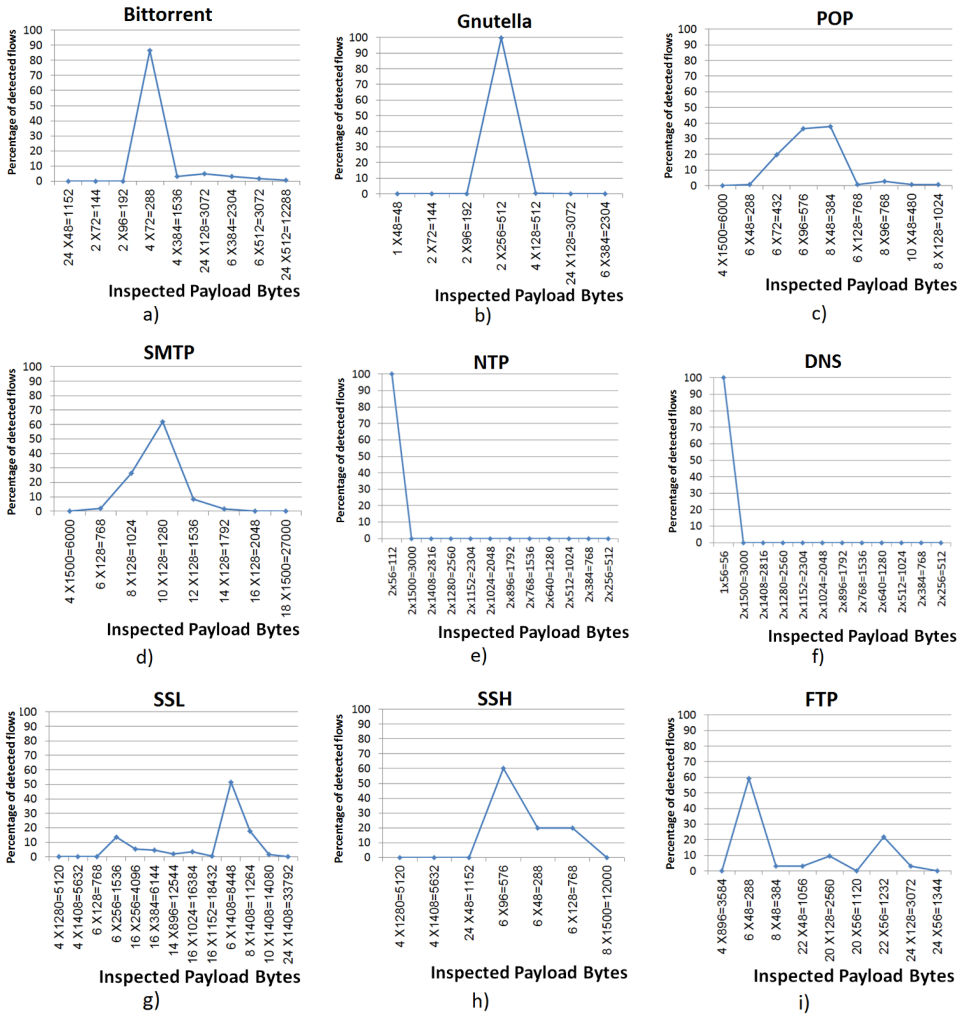


Figure 4.16: Non-contiguous sampling: Results for various protocols as a function of the of the inspected payload length: a) BitTorrent,b) Gnutella,c) Mail_POP,d) Mail_SMTP, e) NTP, f) DNS, g) SSL, h) SSH and i) FTP

analyzed is considered. Nevertheless, a more detailed comparison including the computational costs is required.

For this purpose, a measure of the associated computational cost has been conducted to assess contiguous and non-contiguous sampling schemes.

Figure 4.21 shows the total inspection time t_{fc} for dataset CS-A1 as a function of the inspected payload size for both sampling methods (Method-I and Method-II) v.s. the full payload inspection case.

In Figure 4.21, the classification time required the full payload case is invariant in function of the inspected payload size used by the proposed sampler.

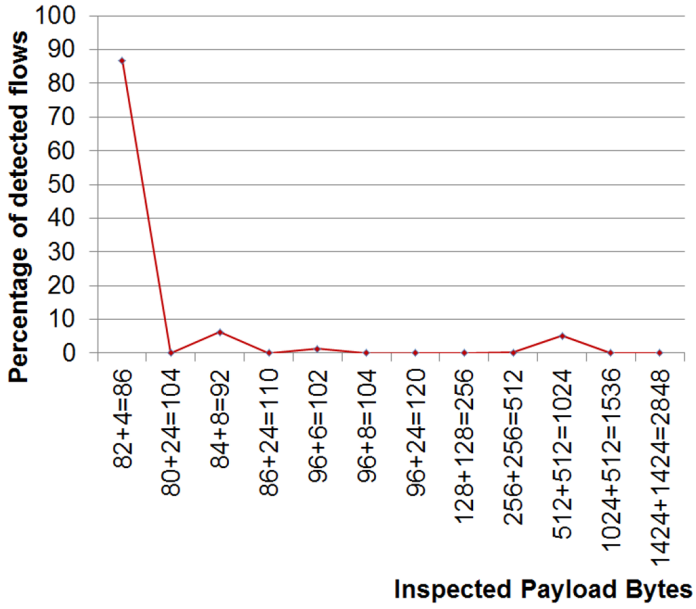


Figure 4.17: Histogram of DPI classification detection percentage for P2P protocols as a function of the inspected payload size using contiguous sampling

As such, it is shown as a straight line to help in comparing the classification time with Methods I and II.

As shown in both graphs of Figure 4.21, the inspection time increases with the inspected payload size and it is always inferior to the time consumed when inspecting the full traffic payload. However, the overall total classification time consumed with Method-II (Figure 4.21b)) is lower than that with Method-I (Figure 4.21a)).

Computational results should be also regarded from a classification point of view. For this reason, DP results obtained through Figures 4.15 and 4.20 are summarized in Table 4.3 for all protocols in the CS-A dataset together with computational results.

In summary, to achieve DP values above 90%, less volume of inspected payloads is required when using contiguous sampling ($I = 1792$ bytes) than when using non-contiguous sampling ($I = 9216$ bytes). Considering the points of Figure 4.21a), where *inspected payload size* equals $I = 9216 = 8 \times 1152$, and of Figure 4.21b), where *flow detection number* equals $I = 896 + 896 = 1792$ these two points (highlighted in red) represent respectively the best case scenarios ($DP > 90\%$) for non-contiguous and contiguous sampling methods.

From this standpoint, contiguous sampling obviously outperforms non-contiguous sampling in terms of the required payload inspection time which should be the more likely the case with the classification time. For this reason,

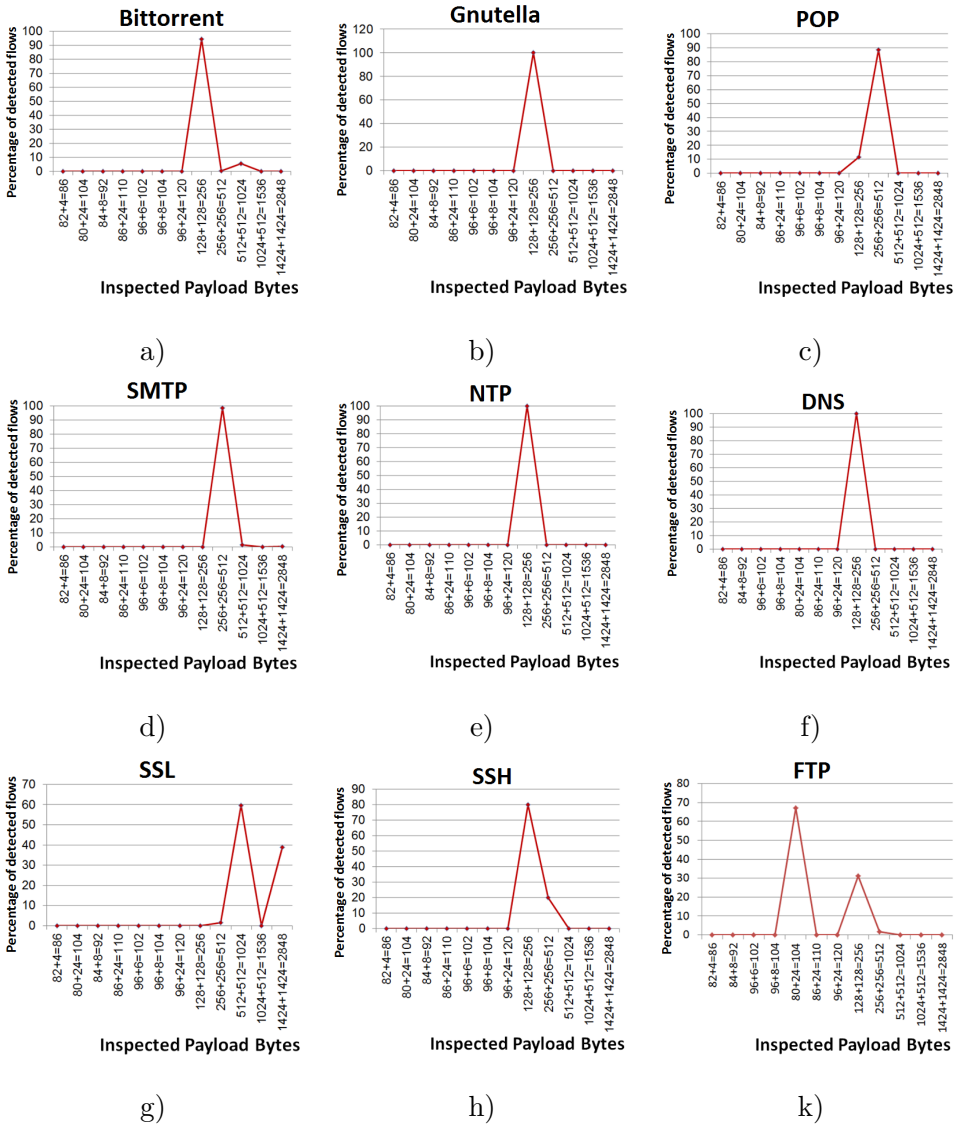
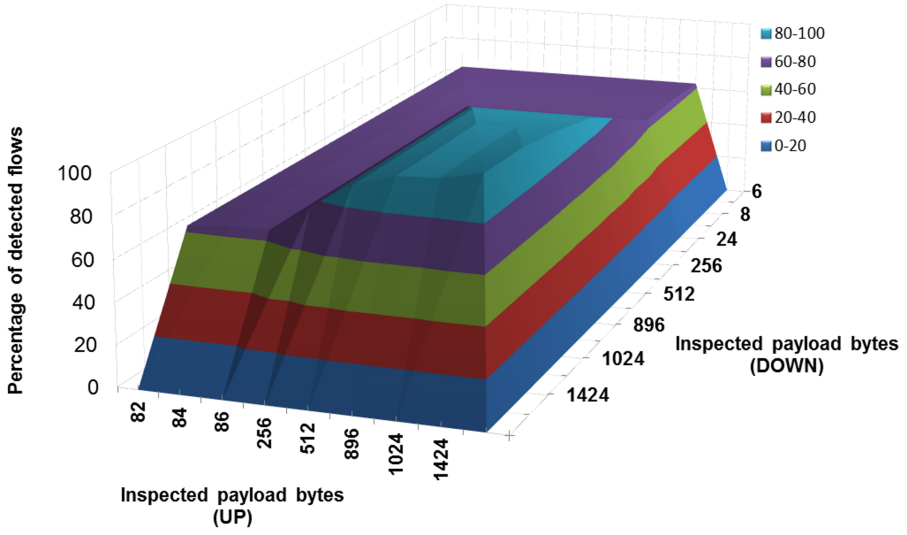


Figure 4.18: Results for various protocols as a function of the inspected payload length for contiguous sampling mode: a) BitTorrent, b) Gnutella, c) Mail_POP, d) Mail_SMTP, e) NTP, f) DNS, g) SSL, h) SSH and i) FTP

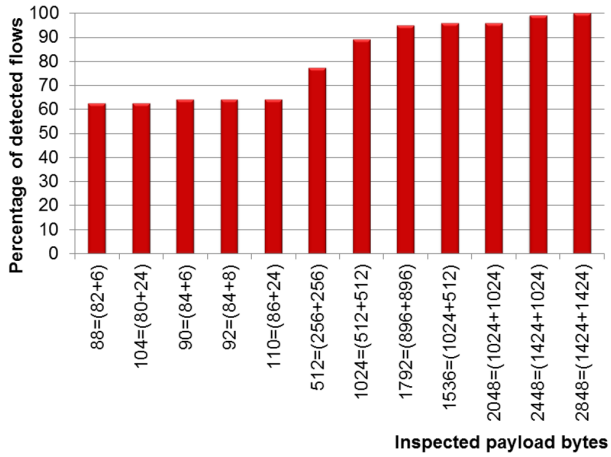
the total classification time for the whole dataset CS-A1 using both methods I and II, t'''_{fc}, –Equation 4.10–, and the original one, t_{fc}, –Equation 4.4– were measured for the best case scenarios in each sampling scheme. The obtained results are summarized in Table 4.3[15].

Based on Table 4.3, it can be clearly seen that the best sampling scheme for DPI, compared to other sampling methods, can be achieved with com-

[15]The handling time, t_h, is accounted as part of the evaluated classification time.



a)



b)

Figure 4.19: DPI classification detection percentage as a function of the inspected payload size using contiguous sampling, for all protocols (except for DNS): a) three dimensional and b) two dimensional

bined sampling in contiguous inspection mode (Method-II) where more 95% of flow DP can be maintained with a classification time gain of 12.47% for and inspection time gain of 93.73%.

Based on Table 4.3, Method-II will be the basis for the recommended optimal sampling policy for DPI classification tools, as proposed next.

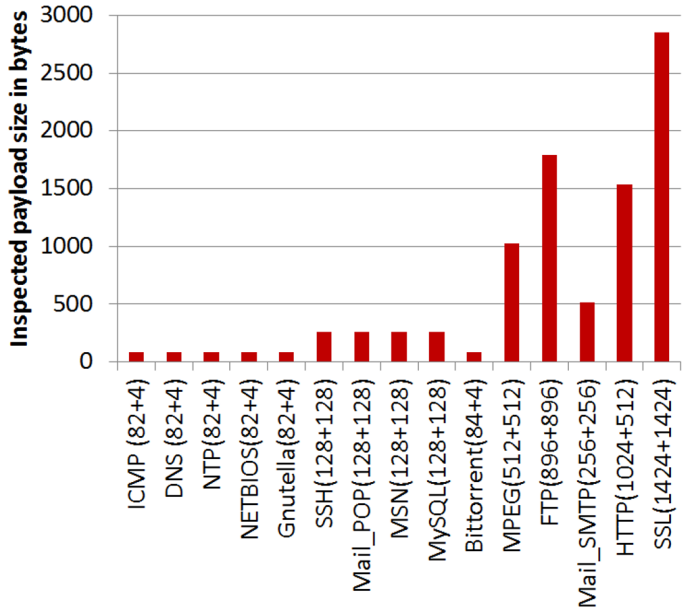


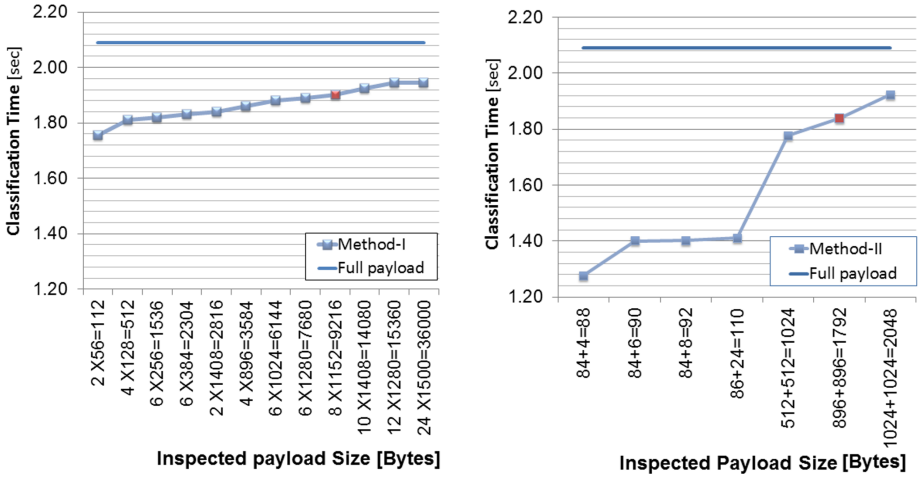
Figure 4.20: Contiguous sampling: Payload size to be inspected per protocol for DP above 90%

Table 4.3: Comparison between best case scenario results for sampling methods: Method-I and Method-II

Sampling Method	**Payload**		**Flow** DP	t_{fc} **gain**	t_i **gain**
(I) Non-contiguous sampling	$I = 9216$ Bytes	First $p = 8$ packets per flow, first $b = 1152$ bytes per packet	91.50%	9.01%	67.70%
(II) Contiguous sampling	$I = 1792$ Bytes	First $C_u = 896$ bytes UP, first $C_d = 896$ bytes DOWN	95.00%	12.47%	93.72%

4.5 Optimized DPI Sampling

At this point, all of the proposed sampling schemes have been individually evaluated, based on which, basic recommendations on the optimal DPI sampler should be presented.



a) b)

Figure 4.21: Total classification time, t_{fc}, for dataset CS-A1 as a function of the inspected payload size for: a) Method-I and b) Method-II, v.s. full payload

4.5.1 Proposed DPI Sampling

The proposed sampling scheme is targeted at only inspecting the *classification bytes*, which, according to the previous results, can be achieved through contiguous sampling of the initial bytes in each flow, by analyzing only a few initial packets. The remaining packets in each flow will be analyzed for packet handling purposes, that is, basically for attributing them to the flow they belong to.

One of the conclusions from the previous assessment is that the DP is dependent on the inspected payload length and the protocol, that is, each protocol requires a different value to reach a given DP. Therefore, the location of the *classification bytes* will be different for each protocol. This is not a surprising result given the different protocol behaviors in exchanging their initial signalling messages.

Consequently, it is possible to set up an upper limit for the number of inspected bytes on a per protocol basis as to include the *classification bytes* for this protocol. Thus, by adjusting this upper limit, the sampling policy can be tuned to reach a required value of DP and to classify one protocol or the other, by covering the associated classification bytes locations.

Formally, let A_0 be a required flow DP value and $\{B_p\}$ the upper limits for the position of the *classification bytes* for each protocol p. According to our experiments, there should exist a value for the inspected payload size, C_0, such that A_0 is achieved when using contiguous sampling with minimum inspection. Therefore, given the pair (A_0, C_0), the sampling policy can be stated as:

Given A_0, a DP level to be maintained, the DPI sampler should contin-

uously pick all payload bytes, starting from the beginning of a flow, up to inspecting C_0 bytes.

At this point, the question is how to select the value for C_0, given A_0. For different protocols in a given set, C_0 should be assigned to the highest positions of the *classification bytes*, to be able to cover all remaining protocols with lower positions. This condition is critical to be able to generalize one DPI sampling scheme for all protocols.

For example, for the datasets we used, if the classification target is to only detect P2P protocols, C_0 can be chosen equal to 92 bytes to attain $A_0 = 92\%$ (see Figure 4.20). When all the protocols are targeted for classification, which is the most common case, the protocol with the highest value for the upper limit (SSL according to our results) should imply a value of at least $C_0 = 1792$ bytes to attain $A_0 = 95\%$.

4.5.2 Advantages of the Proposed DPI Sampler

The proposed DPI sampling methodology can be easily be evaluated in terms of DP and computational gain based on the results previously shown in Table 4.3.

Using the proposed sampler, 95% of classification DP could be obtained for all protocols with 12.47% of classification time gain.

Another relevant issue is to compare our DPI sampler with other sampling schemes. For example, one of the most referred works, EIM [Chen, 2009b], which does not necessarily include the first packets of a flow, obtains a DP of only 15% when using a sampling rate of 8 out of 13 packets, with a gain of 31% in the inspection time. Clearly, the proposed approach outperforms EIM, that provides a 93.72% of reduction in the inspection time with 95% of flow DP.

Moreover, with the proposed sampling scheme, the following advantages could be obtained for DPI classification:

- Maintaining the classification results: With the proposed DPI sampler, DP level can be maintained for all protocols up to a predefined limit of choice.
- Enhancing the user privacy level: By minimizing payload disclosure, the sampler can mitigate privacy breaching and legal concerns which are usually associated with the native DPI classification based on full inspection.
- Protocol consideration: Most sampling policies were general with no specific attention on the detection of a particular protocol (or class of protocols). For example, with EIM, packets are sampled regardless of the classification bytes locations. However, by sampling the signalling phase of each application flow, the proposed DPI sampler is more accommodated, in this sense, to the characteristics of the traffic being classified.

- Minimizing the inspection time: The DPI signature matching process will be faster due to inspecting less payload data at predefined locations, instead of applying full flow inspection. Most sampling policies provide sampling rates, which implies that the number of sampled packets will increase as long as the flow is under course. In the proposed sampler, this is fixed to C_0 bytes per flow, which yields higher computational gains especially for large flows. This is of particular importance for cases where packets holding the signatures are delayed or lost. This might be the case for flows that are unknown when applying the DPI classifier on the full flow payload (i.e. before sampling). As such, when the continuous sampling threshold is reached, the flow will be early labeled as unknown instead of further inspecting unnecessary packets.

- Defining minimal dataset payload capture: In order to create proper ground truth results, the captured payload size for the traffic dataset must be large enough to yield to the highest possible accurate classification. Most works in the literature used the first 16 to 80 bytes of the payloads per packet in the tested datasets, since full packet payload capture is constrained by data confidentiality. The tests made in this chapter using a `nDPI` derived tool proved that these values are insufficient for obtaining valid ground truth results.

4.5.3 Discussion

In order to define an optimal DPI sampler, the work shown in this chapter has followed a *proof by experiment* approach due to the lack of documentation and due to the existing heterogeneity on many levels including: DPI tool, application signatures, etc.

The most crucial part of our proposal is the ability to generalize one common value for the number of inspected bytes per flow, C_0, since the relevance and the applicability of this parameter have potential dependencies on the other related choices, including:

- *Traffic dataset choice*: C_0 depends on the tested dataset as it is specified based on the protocol requiring the highest amount of sampled payload size.

- *DPI tool choice*: The required payload size obtained with `nDPI` is too large ($C_0 = 1792$ bytes) compared to other DPI tools such as `L7-filter` [Aceto, 2010], which reportedly requires only the first 32 bytes. Apparently, each DPI tool might incorporate different matching algorithms and application signatures. In fact, `L7-filter` mainly relies on detecting protocol patterns while `nDPI` incorporates other helper techniques such as statistical analysis and finite state machines to enhance the DP. According to the available evidences, this is affecting some results, especially

those related to the packet truncation method. Nevertheless, `L7-filter` recognizes fewer protocols and is nowadays obsolete. This is particularly relevant for P2P protocols, which are the hardest ones to detect and were not included in `L7-filter`.

As a result, generalizing one consolidated DPI sampling scheme (i.e. one value of C_0) requires in turn a consolidated DPI validation tool. Nevertheless, the presented recommendations on DPI sampling can be generalized regardless of the DPI tool of choice and the tested dataset.

This chapter gives end to the payload assessment and optimization part. One of the major outcomes of this part is the underlined discriminative power of the first exchanged packets during the establishment of a flow belonging to a certain application, a characteristic that seems to be intrinsic for most application protocols.

In the next chapters, though a relatively different discipline in traffic classification is addressed (blind classification), we attempt to make use of the experience gained throughout this chapter with DPI assessment.

Specifically, the next chapter shows a proposed blind classifier that explores the first exchanged application layer messages, instead of the payloads of the IP packets, at the beginning of each session. Exploring the traffic properties in the same zone where application signatures are located inside a flow is one of the main motivations of the blind classification proposal presented in the next chapter.

Chapter 5

Network Traffic Application Identification Based on Message Size Analysis

In this chapter we present a new blind, efficient and accurate method to identify the application protocols through the analysis of the sizes of the *messages* exchanged between application-level processes. The suggested method is blind and falls in the category of *quintuple-centric* classification, where the classified targets are *micro-flows* (often simply called *flows*), identified by the quintuple (ip_1, $port_1$, ip_2, $port_2$, *transport-protocol*).

As we stated in Chapter 2, the selection of a set of traffic parameters is a strategic choice for traffic classifiers, especially blind ones, which rely on non-payload traffic features usually extracted at the network and transport layers of the Open System Interconnection (OSI) model [Khalife, 2014, Moore, 2005a]. Ideally, the selected features should be discriminative, immune to network dynamics and obfuscation techniques, while still protecting the user privacy. In [Khalife, 2014] we presented a survey of traffic classification methods and showed that most of them use non-payload features, usually extracted at the network and transport layers of the OSI model [Khalife, 2014, Moore, 2005a].

A blind classifier should better use input features that are resilient to the diversity in the underlying network technology, as well as jitter, congestion and other random phenomena. In this sense, individual packet sizes depend on the network technology's Maximum Transmission Unit (MTU), and, on the other hand, packet interarrival times are sensitive to jitter. Another aspect of input resilience concerns traffic obfuscation. For one, the issue of port number obfuscation is well known and admitted by the research community; so that classifiers based on port numbers are considered obsolete. Concerning packet sizes, it was reported in [Iacovazzi, 2010] that some applications use padding to

tamper the packets and evade packet-size based classifiers, and in [Wu, 2012] that packet sizes and many other traffic features exhibit similar distributions through different application protocols (e.g. packet size distributions of BitTorrent and HTTP). In [Yang, 2012], where a Bittorent traffic identifier was presented, the authors reported that the first three messages of BitTorrent handshaking have distinctive size properties, but unfortunately they are not transmitted in single packets; they are rather divided into several packets for the purpose of obfuscating packet-size based classifiers. As a conclusion, the authors recommended the use of messages instead of packets to detect the size features.

Application-level messages are not totally immune to the above mentioned variations and obfuscations. However, among the commonly used features, they present the highest resilience, together with the highly desirable property of being derived directly from the communication of application entities, and as such, they permit to look straight to the target, the applications they are supposed to identify.

We present a blind, quintuple centric traffic classifier based on the sizes of application-level messages, as observed at the transport layer. The aim is to classify flows (quintuples) through the analysis of the "message-sequence patterns". Although the keyword message designates data units exchanged at the application layer, we show that the sizes of these messages can be extracted from layer 4 data headers, without actually inspecting the payloads of the messages. We then show by experience that by applying a supervised Bayesian analysis to the sequence pattern (sizes, directions and positions of the exchanged messages) we can identify the involved application with good accuracy.

Although at an early stage, it is useful to provide a preliminary discussion of the main novelty of this work, that is, message-size analysis, and the motivations that lead us to build and evaluate a blind classification method based on this feature. To the best of our knowledge, the message-size feature has not been sufficiently investigated and its potential remains to be revealed, although it has occasionally appeared in the literature, as we will see in the related work section. In fact, most if not all of the previous similar work has focused on the analysis of packet sizes, occasionally combined with heuristics exploiting timing information.

Our main critic to previous work on packet size analysis is that we did not find enough comprehensive effort to interpret the feature in the particular target context of network application identification. We have come to the conclusion that most of the mentioned work aimed to experiment standard statistical methods to the "obvious" available feature (packet-size, eventually timing information), privileging the technical aspects belonging to the science of data analysis, while almost neglecting the necessity to weigh the ideas and the results in the specific context of the layered protocol architecture. This point contrasts with our methodology, as will be better clarified when we pro-

vide an insight into the message-size feature with a particular regard to the way network application protocols are usually designed. For instance, we can remind two main characteristics of the universal layered protocol architecture:

1) Application-level processes exchange *messages*, not *packets*; the latter are subject to segmentations and retransmissions freely decided by the transport layer in the context of reliable connected transport.

2) Application protocols are designed around *methods*, which are the basic semantic units for communication at the application level. Accordingly, these methods are the *generators* of the observed messages, and a comprehensive study of the characteristics of these methods is essential to pave the way for a classifier that efficiently exploits the generated observations.

Arguably, application-level messages are not totally resilient to random network events and to obfuscation techniques. However, compared to other commonly used features, such as packet sizes and inter-packet times, they present the highest resilience and the highly desirable property of being derived directly from the communication of application entities. As such, they permit to look straight to the target, the applications they are supposed to identify.

Finally, it is important to stress that the novelty of this work is not restricted to the use of message sizes as the main discriminative feature for classification. Indeed, the classification method that we will suggest is not a reform of previous classification techniques by simply substituting packet-size observations with message-size observations. The road map we followed starts with a comprehensive insight into the application-level messages and to the way they are generated in the light of application-level protocols and the way these latter are designed. The proposed classifier is built on top of this insight, through the definition of appropriate topological and statistical models, and the application of Bayesian and machine-learning techniques that are most convenient to these models.

The road map of the approach is reflected by the structure of this chapter. After a survey of related previous work from the literature, with particular attention on similarities and contrasts with respect to our method wherever appropriate, we will proceed as follows:

- First, an insight into application-level protocols and process-level messages is made.
- Based on it, a definition and normalization of message size vectors and an appropriate metric distance to assess their degree of similarity are proposed.
- Then, to build those parameter vectors, a method to extract the sizes of the messages from TCP/IP headers is described.

- Next, we address the definition of statistical models for the message size vectors.
- The proposed classifier architecture is described next. It is based on Markov models, which are used to represent the generation of the messages by the different applications. Obviously, both models, i.e. statistical model for the vectors and the Markov model, should be properly trained in order to build a profile for each class. Those profiles are then used to classify flows by a naïve bayesian classifier.
- Finally, the experimental results are shown.
- As a final remark, some conclusions and future work are outlined.

5.1 Related Works

It is widely admitted that payload based techniques, namely DPI [Klaus, 2009], have the highest classification accuracy thanks to their ability to inspect the packets' payloads and match discriminative application signatures. In [Lu, 2014] a hybrid method combining port numbers and packet inspection was suggested. However, classification based on port numbers is obsolete since malicious applications can use arbitrary port numbers, including standard ones for sake of obfuscation. Moreover, packet inspection breaches the users' privacy and fails to process encrypted payloads. Additionally, the need to analyze the whole payloads of every packet in the network represents a big challenge from the computational point of view.

Blind classifiers rely on the analysis of patterns of traffic observed at the transport layer without inspecting the packets' payloads. This grants them the ability to deal with obfuscation [Zink, 2012], encryption and tunneling [Mujtaba, 2009], as well as user privacy rules, at the expense of some sacrifice in accuracy. Blind classifiers usually require less computational power than DPI classifiers, mainly because the latter have to process a larger amount of data. The experience has shown that blind classifiers are well suited to detect non-standard applications (e.g. P2P, BitTorrent [Zink, 2012]), which are intrinsically hard to detect due to their decentralization and dynamicity and especially their use of obfuscation and private, non-standard techniques.

Various blind traffic classification techniques are analyzed and experimented in the literature, mainly falling in two categories: host based techniques, that classify host activities by analyzing interaction schemes [Karagiannis, 2005]; and quintuple-centric techniques that classify flows based on key features observed at layer 4.

Examples of the latter category include [Zander, 2005, Tabatabaei, 2012, Gu, 2010, Zhenxiang, 2011, Erman, 2007a, Auld, 2007, Yildirim, 2010, Crotti, 2007, Wang, 2010, Li, 2008, Dainotti, 2008, Huang, 2009] and [Wang, 2007].

These features typically include the flow size and duration, the packet sizes, the packet interarrival times [Crotti, 2007, Wang, 2010], and so on. Our work falls in the latter category, with the distinction of using application-level messages instead of packets, while ignoring interarrival times.

Different Internet applications present different distributions in their packet sizes [Wu, 2012], and many classifiers use this property. Some use simple statistical techniques [Yildirim, 2010, Crotti, 2007, Wang, 2010] such as PDFs while others use advanced ones such as application profiling [Wang, 2009b] or ML [Zander, 2005, Tabatabaei, 2012, Gu, 2010, Zhenxiang, 2011, Erman, 2007a, Auld, 2007, Li, 2008, Dainotti, 2008, Huang, 2009, Erman, 2007b, Moore, 2005b]. Some authors [Khalife, 2014, Zhenxiang, 2011, Erman, 2007a] suggested Bayesian supervised classifiers, especially naïve ones, as they are particularly characterized by their low complexity, fast training and computational efficiency.

Various relevant ML approaches can also be found in the literature. K-means [Erman, 2007b] and AutoClass in [Zander, 2005] were reported to identify some P2P applications with up to 80% of accuracy. Using ANNs in [Gu, 2010] and SVMs in [Tabatabaei, 2012] yielded up to 85% accuracy for detecting a set of P2P applications. KNN algorithm provides 90% of reported accuracy [Huang, 2009] for some known applications including BitTorrent. However, the long training time and high complexity associated with most supervised learning algorithms (ANN and SVM), the high storage and computational resources, associated with others (i.e. KNN) implies a low scalability and a lack of generalization capabilities regarding the monitored network and the temporary evolution of the traffic. On the other hand, despite the high reported accuracy, only a very limited set of protocols have been checked in these contributions. At the time this work was conducted, Rizzi et als. [Rizzi, 2013] proposed a neuro-fuzzy classifier with low structural complexity, yet comparable results to SVM. There are two major differences to note with this work. First, the former uses packet sizes and interarrival times as features, while our method uses message sizes and ignores the timing data. Second, in the context of network traffic, new applications appear frequently, and a method based on independently "profiling" the applications is highly desirable. Our method follows a profiling approach, where the training consists of profiling each application standalone.

Bayesian techniques, as in [Zhenxiang, 2011, Auld, 2007, Moore, 2005b], have particular simplicity and low computational resource requirements [Khalife, 2014]. Given an element, characterized by the observation of its features, these techniques estimate the probability that a class generates such an observation and label the element with the class that provides the highest probability. As such, these methods train very quickly, have low complexity and require reasonable computational resources. However, the main characteristic of naïve Bayes classifiers, which is the reason behind their low complexity, is the assumption that the different features are independent and have standard

Gaussian distributions under the normality assumption. To overcome some of its limitations, the naïve Bayes model has been subject to many enhancements, for example through incorporating neural networks [Auld, 2007] and payload inspection [Zhenxiang, 2011].

On another approach, the work in [Wang, 2009b] suggested identifying applications through the detection of the Longest Common Subsequence in their packet sizes. Reportedly, the idea was successful on a specific set of four P2P applications (Maze, Thunder, PPLive and Feindian). More general results are not available, and some preliminary experiments carried out at our lab did not show such packet-size signatures in a significant number of applications, even with a reduction of the alphabet by rounding or clustering methods. Although many applications have some discriminative packet sizes, the majority of application methods rather generate variable sized packets that are better described with probability distributions than with discriminative key values.

In this work, our proposed system analyzes the sizes of application layer messages using a Bayesian approach to label each flow from the probabilities provided by a set of Markov models, each one associated to a given protocol. To the best of our knowledge, we are the first to combine the discriminative power of messages at the application layer with the simplicity and performance of Markov-based classifiers and the use of Multi-Peak Gaussian distributions to characterize message sizes. Our classifier analyzes messages based on their size, their direction and their specific positions in the flow.

Few approaches analyzed traffic properties at the application layer without inspecting the payloads. Probably the most related previous works to this one are [Waizumi, 2011] and [Jaber, 2009]. In [Jaber, 2009], packets sizes are analyzed instead of message sizes. In the training phase, the authors used K-means to classify the distributions of the packet sizes and provide one global cluster model. Then for each target application, and for each packet position in the flow, a probability was assigned to each cluster. The classification process examines the packet sizes as they come and computes accordingly the probability of the flow (packet-size vector) to be generated by an application. The flow is then assigned to the application providing the highest probability in a naïve Bayesian way. In [Jaber, 2011], the same authors propose an enhancement via the use of the packet interarrival times as a feature. Although a previous feature analysis in [Erman, 2007a], using backward greedy search, reported that "features that have a time component such as duration, interarrival time, and throughput were found not to be useful by the feature selection algorithm", the proposal in [Jaber, 2011] was to subtract the observed "monitor-to-server" round-trip-time from the inter-packet times, and as a result, this feature is argued to become meaningful when observed at approximate positions in the flow. Although the reported results suggest a potential usefulness of the inter-packet time feature, the suggested technique permits to cope only with the randomness of host locations. The authors admittedly neglected the effect of

variations in network conditions and jitter, and it is not clear to which extent this assumption is valid under various network conditions and congestion. Also, the evaluation, both in [Waizumi, 2011] and [Jaber, 2009], was applied to only five standard applications (HTTP, SMTP, HTTPS, SSH and IMAP). As a main difference, packet time information is not used in our model. Other key differences of our work with [Jaber, 2009] are the use the message level sizes instead of packet sizes and the use of a per-application profiling model for clustering the message sizes.

The work presented in [Waizumi, 2011] has a common point with ours in the fact that it analyzes the message sizes for the classification. Despite of a different mathematical model, the key difference is that [Waizumi, 2011] quantifies the message size in terms of number of packets. Our experience showed that this coarse-grained quantification overlooks important and meaningful characteristics of the message. For instance, following this quantification, all messages involving one sole packet are considered similar. This does not permit to identify key message sizes, especially those involved in some application level handshaking and methods, such as SMTP "HELO", HTTP "GET", FTP "USER" and so on. These methods often generate one-packet messages; still their expected sizes are different and provide potential information for the classifier. For these reasons, our quantification of the message size is fine grained, based on the total number of bytes in the message and not only the number of packets.

Finally, we should note that a trend is emerging to monitor message sizes as the main source of information on encrypted traffic flows. This is noticeable in [Pironti, 2014] and [Iacovazzi, 2014], and it reveals again the importance of the message size feature for flow classification.

The main contributions in this work are:

- *Message size vectors*: the use of message-size parameters instead of packet-size parameters. Most previous work on identifying applications from packet lengths used the (non-empty) packets to generate the parameter vectors, based on their sizes and direction. Although we use the same notion for the direction, we followed a different approach for the sizes and the definitions of the parameter vectors, which removes from the process-level messages the "'noise' induced by layer-4 segmentations, retransmissions and acknowledgments.

- *Message size scaling*: definition of a normalized message size measure and a metric distance that is appropriate to their semantic meanings in network communications.

- *Gaussian mixture*: the use of a Multi-Peak Gaussian model (MPG) for estimating the PDFs of the vectors' components.

- *Profiling*: the model is simple and extensible with minimal effort, as it consists of profiling each application standalone. The addition of a

new application protocol requires rerunning the training procedure on a sample set to extract its parameters, without any need to review the other applications' model parameters and sampled data.

- *Testing*: the model was tested on a relevant number of applications mixing standard client-server and P2P protocols. Many previously reported works were tested on a small and specific set of application protocols. We tested our method on a dataset containing about 3 million flows and 18 application protocols, 10 using TCP and 8 using UDP.

5.2 Protocol Methods and Messages: an Insight

Since the application-level protocols are the classes of interest for network traffic classification, it is important to point out some major characteristics that are common to these protocols, especially those characteristics that have major impacts on the patterns observed in the flows subject to classification. In this section we point out some important characteristics that are most relevant to our approach, since they justify the use of the message size sequence as a main feature, and further motivate many decisions in the design of the suggested classifier. The design of any application-level protocol involves the definition of metadata units called Methods, which specify the syntax and semantics of the messages exchanged between the application entities that use this protocol. Our work was mainly motivated by this axiom, which implies that a good knowledge of the methods and their characteristics should provide a good basis for the recognition of a protocol. In the context of blind traffic classification, this knowledge involves the sequencing of the methods as well as the sizes of the messages they generate.

As we will see later, the basic building block of our suggested classifier is an application profiler: a systematic machine-learning procedure that captures the essentials of the methods of a given application protocol. All the protocols subject to classification are profiled, separately, and the classifier uses these profiles to find the one that best matches any flow subject to classification.

5.2.1 Protocol Methods

A flow belonging to some protocol X is a sequence of messages, where each message is an instance of one and only one of the methods of the X protocol. This means that the methods of X are the *generators* of the messages observed in any X flow. Examples of protocol methods are HTTP "GET" and "POST"; SMTP "HELO", "MAIL FROM" and "RCPT TO"; FTP "USER" and "PASS", etc.

Any protocol usually defines a set of methods dedicated to some handshaking procedures. In most cases, these handshaking methods generate relatively small messages, often transportable in one single packet. In this context, the

definitions of *message* and *packet* coincide, so that a packet-based classifier and a message-based classifier are likely to yield similar results. On the other hand, some methods involve the transfer of large amounts of data; this data transfer takes the form of a sequence of full-size packets flowing in one direction while empty "ACK" packets flow in the reverse direction. A packet-based classifier would consider this sequence as a sequence of unrelated packets, but from the point of view of a message-based classifier, this sequence of packets is merged into a single large message. It follows that the major difference between message-size classification and packet-size classification is in the way large messages (composed of more than one packet) are handled.

Since the methods are the generators of the messages, the question that arises here is, how similar are the messages generated from the same protocol method? The answer to this question is empirical and approved by experience. Some methods generate fixed message sizes (i.e. Bittorent 1.0 *handshake*: 68 bytes), but in the general case, most methods generate variable size messages. Most importantly, although the messages generated from the same method may vary in size, the sizes are almost similar with some reasonable amount of randomness. In general, the sizes have a distribution centered on a mean value, and the normality assumption is statistically valid.

It is theoretically possible to profile a protocol by enumerating all its methods and estimating the Probability Distribution Function (PDF) for each of them from sampled messages. However, a more practical approach is to estimate these PDFs using an unsupervised learning procedure, eliminating the need of any prior knowledge of the set of methods. The unsupervised training procedure will be shown in a subsequent section, dedicated for the training of the classifier. For instance, we should note that this approach is preferable because it is practical to apply it on complex protocols, and it also permits to handle undocumented protocols.

5.2.2 Sequence of Methods

In protocol design, the sequencing of the methods can be quite simple or quite complex, according to the complexity of the protocol itself. In the simplest case, the sequence of methods can be rigidly predefined, such as in the DNS protocol where the flow is an alteration of the request and reply methods. In the more general case, the sequence of methods is contextual, depending on the scenario occurring in the application context. For example, HTTP is a relatively complex protocol with many methods and no single predefined sequence.

In practice, there always exist some implicit or explicit rules that govern the sequence of methods, so that some sequences are more likely than others, and some sequences are even impossible. As a result, the probability of occurrence of a given method at a given position in the flow depends on the history of the flow, namely on the methods that preceded it. This raises the question of how

to model this dependency. There are many ways to address this issue, each leading to a different model and a different level of complexity.

An obvious way to handle dependency is to represent each message size vector as a single point in L-dimensional space, where L is the size of the analyzed message sequence, and to apply some clustering to assess the similarity between these vectors by their metric distance. The major drawback of this approach is the so-called *'curse of dimensionality'*: huge amounts of sample data would be required for training such a classifier, especially when the dimension L grows above 3.

Another way to handle the dependency would be the definition of a first order Markov model, where states represent methods and the transition matrix reflects the probabilities of succession between the methods. A second order Markov model would be capable of handling two-step dependencies, but at the cost of an even higher complexity (more states and bigger matrix). In fact, each additional level of dependency increases the model complexity and moreover, it requires much more training data in order to estimate the various parameters reliably enough.

The models mentioned so far are highly complex. At the other extreme is the Ostrich model: ignoring the dependency altogether and assuming that for each method, the probability of occurrence at any position in the flow is the same, independent of what happened before reaching this position. Poor accuracy is expected from the Ostrich model, because it treats all the messages of a flow in a similar way, disregarding very meaningful information.

The solution we adopted is a compromise between accuracy and model complexity. It implicitly handles the dependency by assuming the probabilities of occurrence vary with the position in the flow. Indeed, application methods occur at typical positions in network flows. Although this assumption is not deterministic, it is statistically significant, especially at the beginning of the flows, in the first exchanged messages which are often dedicated to handshaking. The resulting model, that will be detailed later, is a first-order (observable) Markov model where each state represents a position in the flow. The profiling of a protocol consists of applying the training process to each individual state, in order to capture the most occurring methods at this state as well as the shapes of the messages they generate. The result of this training is a mixture model that allows assigning a weight to each individual method; this weight reflects its probability of occurrence at the given state.

A final note here concerns some particular behavior occurring in the FTP protocol, where the handshaking occurs in one flow called "the main connection", while the data transfer (upload or download) occurs on a "secondary connection", opened specifically for the data transfer method. The secondary connection is identified by a different quintuple than the main one, and no handshaking occurs in the resulting flow; the file transfer takes place immediately. As a consequence, a quintuple centric classifier will observe some flows consisting of a single large message. As long as FTP is the only application

that produces such a scenario, this is not problematic for the classifier. However, in the presence of other such applications, the classifier needs additional information to classify the flows consisting of a single large message. In the current status, one-message flows are ignored, and the classifier was tested on flows that include at least two messages.

5.2.3 Visibility of Message Sizes from Transport Headers

It is possible to extract message size information from layer-4 headers, without inspecting the payloads. A method to extract this information will be shown in a subsequent section.

5.2.4 Message Size Scaling

The protocol methods are encoded inside the payloads, and we seek to establish a blind classification method that does not investigate these payloads, but exploits potential information from the sizes of the messages generated by the methods. In order to be able to estimate the likelihood of a message to be generated from a method, based on its size, we need a way to assess whether two messages have similar sizes and to which extent. This is the role of the metric distance that we should define.

The intuitive Euclidean distance between messages, defined as the "absolute difference" in terms of bytes, is a rather naïve choice because in topological terminology, it possesses the "translation invariant" property:

$$D_E(x, y) = D_E(x + a, y + a) \tag{5.1}$$

This property is not appropriate for our application, because it does not fit well with the way the network applications messages are generated from protocol methods. As an example, consider a data transfer message. The transferred data might be a file of 20 Kbytes or 30 Kbytes. The absolute byte-difference is 10 KB (big), but still, the two messages have the same semantic meaning, and likely belong to the same method in any network application (i.e. transfer of an icon or a small image). On the other hand, a "HELO" message in SMTP has a typical size of 30 bytes while a HTTP GET request has a typical size of 300 bytes. The difference here is less than 300 bytes, but it is more meaningful than the 10 KB difference in the above-mentioned image transfer. The defined metric distance must point out that the same byte-difference is more significant between two small messages than between two big ones. The bigger the message, the less significant is the byte-difference.

Therefore, the metric distance defined on message sizes must consider relative differences rather than absolute differences. In other words, it should not be translation invariant. For this reason, we will define later a transformation that normalizes the message sizes into the]-1, 1[, together with a metric distance that permits to assess the degree of similarity between any pair of messages relatively to their sizes.

5.2.5 Potential of Message Size Analysis for Identifying TCP Applications

For TCP-based applications, message size analysis is expected to outperform packet size analysis, because the former reflects what is really exchanged between application processes, while the latter is affected by transport layer segmentations and retransmissions, and these can be considered as a source of noise to the classifier. On the other hand, this assumption is not relevant in UDP-based applications, because no segmentation or retransmission occurs at layer-4, and the flowing packets are directly exchanged between end processes. In this case, the messages are usually the packets themselves.

5.3 Flow Classification Based on Message-Size Vectors

The classification method that we present is based on Markov models with a training procedure that mixes supervised and unsupervised schemes to produce a model for each application (model-per-class). The approach consists in considering the messages' sizes as the productions (observations) of a first order Markov model. The training of the models is achieved using randomly sampled data from each application separately. It consists in estimating the PDFs of the message sizes at given positions in the flow, with the underlying idea that these messages are generated from (a-priori unknown) methods defined in the design of the application protocol. We empirically assume that each method generates messages with normally distributed sizes. Thus, each observation is assumed to be generated from a Gaussian Mixture PDF, which is dependent on:

- the application, and
- the state (position in the flow).

The parameters of the Gaussian mixture associated to each (application, state) are interpreted as follows:

- Each peak represents a method in the design of the application protocol.
- The mean and variance associated to each peak reflect the distribution of messages generated by the underlying method.
- Associated to each peak is a weight; this weight corresponds to the probability of occurrence of the underlying method at the given state.

Finally, when classifying traffic, the class associated to each flow will be that of the model providing the highest probability. This probability is evaluated by a naïve Bayes classifier that computes the Bayesian probability that a messages

size sequence was produced by the model associated to a given application and assigns the flow to the model (application) that provides the highest probability score.

5.3.1 System architecture

The proposed system consists of three main components (Figure 5.1):

- **Features Extractor**: This module extracts, from each flow F_i, the vector, O_i, of message sizes, s_j,

$$O_i = \{s_1, s_2, \ldots, s_L\} \tag{5.2}$$

  where L is a parameter of the system specifying the number of messages to be considered in the classification process.

- **Model Set**: A set of N Markov models (one per application to be detected), characterized essentially by the probability distributions of the messages sizes expected at each state (position) in the flow. It is important to mention here that the underlying first-order Markov model is simple and presents the same topology for all the applications (providing one state for each message position), but the probability distributions for the message sizes at each state differ and are specific to each application. As we explained earlier, this emanates from the fact that the messages are usually generated from the "methods" of the application protocol. The goal of these models is to provide the ability to compute, given an observed vector, O_i, and a model, λ_n, the probability of observation of the vector being produced by the model:

$$\{P(O_i|\lambda_n)/1 \leq n \leq N\} \tag{5.3}$$

- **Classifier**: This is the decision module, which selects the application the flow belongs to as that of the model providing the maximum probability

$$class(F_i) = argmax_n\{P(O_i|\lambda_n)/1 \leq n \leq N\} \tag{5.4}$$

Therefore, the system uses a Maximum A posteriori Probability (MAP) approach in which all the classes are supposed to be equally probable a priori.

A key aspect in the proposal is the evaluation of the observation probabilities for message sizes. For this, as previously mentioned, multivariate Gaussians dependent upon the state and the class are used as PDFs. That is, we use a discrete Markov model in which the set of Gaussians associated to the observable sizes is different for each (class, state) pair. While the Markov model is discrete, the observable messages sizes are assumed continuous and distributed according to a Gaussian mixture. Obviously, the training of the

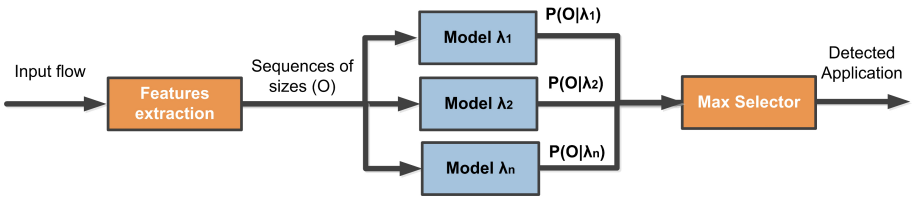


Figure 5.1: Diagram of the proposed system

system requires the obtention of each of these Gaussians for each of the models from the training samples.

The details on the operation of each of the modules are described in the next subsections while a section will be devoted for the details of the training procedure.

5.3.1.1 Features Extraction

As previously explained, the first step to classify a flow is the extraction of the features vector which will be used as the input to the models. For this, each flow is modeled as a vector of relative numbers that parameterizes the sequence of messages.

Specifically, each message is modeled by its byte length with a sign that is positive if the message is generated by the flow's initiator and negative in the other case. To compute the message sizes, the TCP header information is taken into account in order to track any sequence of packets that constitute one message.

In fact, it is known that large data transfers, such as the transfer of an image, a file, or an HTML document, are decomposed by TCP into many packets according to the negotiated Maximum Segment Size (MSS), which derives from the MTU of the underlying network. These data transfers can be tracked by inspecting the layer-4 headers.

At the same time, tracking the layer-4 headers permits to:

- Remove retransmissions from the flow, which is desirable to improve the accuracy and focus on the application-level messaging instead of the "brute" packet flow.
- Remove "pure acknowledgment" packets, which are relevant to layer 4 but have no relevance to the application level messaging.

The number of messages to be considered in the parameter vector, L, is predefined globally for the system (or a maximum is fixed). Ideally, L should be as low as possible in order to classify the flow as soon as possible and to handle short flows. This factor should be selected during the training of the system.

Therefore, the vector generation method acts as follows. Given a flow, $F_i(a \rightarrow b)$, initiated by host a and directed to b, a vector, O_i, of L signed integer values, corresponding to the message sizes and their direction with respect to the flow initiator, is built. TCP handshaking packets and pure ACKnowledgment (ACK) packets are removed from the flow and not considered in the vector. That is,

$$O_i = \{s_1, s_2, \ldots, s_L\}, s_m = \begin{cases} size(message_m), & if\ message_m(a \rightarrow b) \\ -size(message_m), & if\ message_m(b \rightarrow a) \end{cases} \tag{5.5}$$

being $message_m$ the m-th message in the flow.

A distinction is made in the treatment between UDP and TCP flows. Obviously, in UDP flows, packets correspond to messages and there is no distinction between the two concepts. However, as depicted in Figure 5.2 for TCP flows, as long as the payload data flows in one direction, the payload sizes are cumulated into the same message (the same vector component), until one of the following occurs:

- A packet carrying data is detected in the opposite direction.
- A PUSH flag is detected in the TCP header. This flag usually indicates that the sender process has finished its message. An exception is when the receiver's window is full, but this case is ignored in our approach.
- A packet that is smaller than the MSS is detected. This heuristic is based on the assumption that large data transfers use full MSS packets until the transfer is over or the window is full (the latter case is ignored as we stated earlier). Therefore, a small packet usually indicates the end of a message even if no PUSH is detected.

An important remark is that in some application protocols it is possible that two consecutive messages follow in the same direction. Chatting protocols (i.e. MSN) are obvious examples. This rule permits to detect such situations, and makes an improvement over some previous work (i.e. [Waizumi, 2011]) where all consecutive non-empty packets flowing in the same direction are considered as one message.

5.3.1.2 Sequence Evaluation

The proposed method assimilates each application as a pure left-to-right Markov model (Figure 5.3), where each message in turn is generated from a state of the chain: the message at index 1 corresponds to state 1, the message at index 2 corresponds to state 2, and so on.

A set Λ composed by N models, one per considered application, is estimated during the training of the system

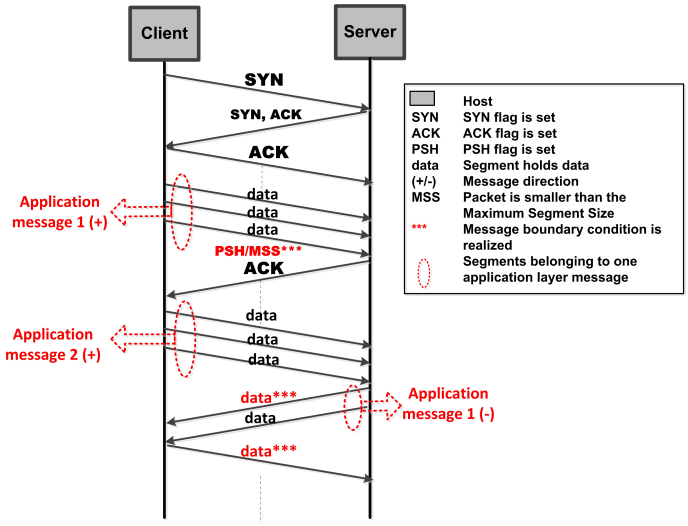


Figure 5.2: Extracting application layer messages from a TCP session

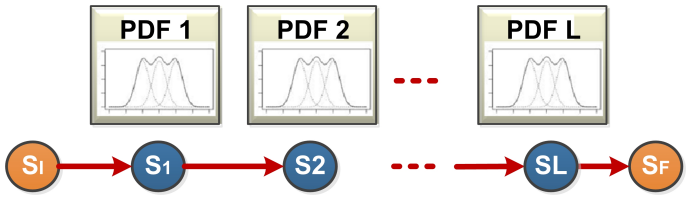


Figure 5.3: Model topology for a single application.

$$\Lambda = \{\lambda_n / 1 \leq n \leq N\} \tag{5.6}$$

As part of the training of the models, a set of PDF for the observations, G, is estimated. Each PDF is associated to a model and a state,

$$G = \{g(n,l) / 1 \leq n \leq N, 1 \leq l \leq L\} \tag{5.7}$$

Therefore, for an input vector, O_i, the system has to evaluate all the probabilities of generation of the sequence according to each of the considered models (applications), as per Equation 5.3.

As a pure left-to-right model is assumed, only the observation probabilities have to be evaluated. Thus, the probability of vector O_i to be generated from application n can be computed as the product of the observation probabilities on all indexes

$$P(O_i|\lambda_n) = \prod_{l=1}^{L} P(s_l|g(n,l)) \tag{5.8}$$

It is worth mentioning that the used modeling assumes that the sizes of the messages at each state (index) are independent from each other. That is, the

probability of having a certain message size at position l is independent on the sizes observed at previous positions $l-1, l-2, \ldots, 1$.

5.3.1.3 Classification

A flow, F_i, will be classified as belonging to the application whose associated model, λ_{app}, provides the maximum probability for generating the observations from the flow, O_i, that is,

$$class(F_i) = arg \max_n \{P(\lambda_n|O_i)/1 \leq n \leq N\} \tag{5.9}$$

This probability of the model given the sequence of observations is not directly provided by the set of Markov models, but the opposite, $P(O_i|\lambda_n)$, that is, the probability of the sequence given a model. By applying Bayes' rule,

$$P(\lambda_n|O_i) = \frac{P(\lambda_n)P(O_i|\lambda_n)}{P(O_i)} \tag{5.10}$$

Assuming that all the models are equally probable a priori, the decision rule can be rewritten as

$$class(F_i) = arg \max_n \{P(O_i|\lambda_n)/1 \leq n \leq N\} \tag{5.11}$$

Although the prior class probability, $P(\lambda_n)$, might be predefined with prior knowledge on the network traffic subject to classification, and hence can be set as a parameter to the method, the proposed approach and the results that we will show assume no such prior knowledge.

There are three reasons behind this. The first reason is that prior probabilities may not be stationary, but change over time as users come and go or engage in different kinds of network activities. The second reason is that these probabilities are highly host dependent and also environment specific. The third reason comes from the way in which the performance of the classifier will be assessed.

As will be detailed in later sections, the evaluation of the system is driven by the "worst case" detection rate. That is, we consider as a main criteria of performance the worst case detection rate, which is defined as the percentage of correctly identified flows from the application that yields the minimum such percentage.

Following this logic, the classifier should not exploit prior statistics by favoring the predominant applications. This is especially true when a classifier is evaluated on datasets that have a largely unbalanced number of elements from each class, as is usual in real traffic, because favoring the predominant application misleads to optimistic results. For these reasons the prior probabilities are ignored and assumed equal for all applications.

In order to evaluate the confidence on the classification of each individual flow, a probability for the flow belonging to each application is set as

$$P(F_i \in class(c)) = \frac{P(O_i|\lambda_c)}{\sum_{n=1}^{N} P(O_i|\lambda_n)} \tag{5.12}$$

An ideal system will provide a probability value of 1 for the correct class and 0 for the incorrect classes. Therefore, the higher the value, the higher the confidence in the classification provided and a better operation of the system.

Furthermore, this measure could be used to train the system instead of the most common case in which the training focuses just in maximizing the production probabilities for the correct classes (maximum mutual information (MMI) vs. Maximum Entropy (ME) training).

5.4 Training of the System

The method used for training the models plays a fundamental role in the proposed system. Up to now, its core is a standard Markov model based recognizer applied to features vectors composed by sizes of the messages.

As the topology and transitions of the Markov models are fixed by design, the parameters to be obtained through training are the PDF to be used, G. Therefore, it is in the choice and estimation of these PDF where the major novelty of the proposal resides. For this, as previously mentioned, we propose to use model and state dependent PDF based on multi-peak Gaussians.

The fundamentals for using multi-peak Gaussians instead of simple Gaussians are related to the proposed modeling. As previously mentioned, each model is supposed to represent a single application (protocol) behavior. But most of the protocols can be split in many methods with different behaviors regarding message sizes. Therefore, the model would be a mix of all the observed methods from a single application.

To account for this, different Gaussians are associated to each state of the model, as will be detailed in Subsection 5.4.2. Anyway, it is worth to mention that no differentiation between those methods will be done explicitly nor for training the system nor for evaluating a sequence.

Estimating one single Gaussian for every model/state would be done by using just all the samples associated to each model/state and fitting the parameters. This assignment would be almost trivial as the training samples should be labeled and each observation is directly associated to a state in the Markov model according to its position in the sequence. But, as multiple Gaussians are to be estimated for every state, a method to assign a given observation to one or many of them is required. For this, the proposed solution uses a quantization and normalization of message sizes based on a proposed metric providing a weighting of the belonging of an observation to different distributions.

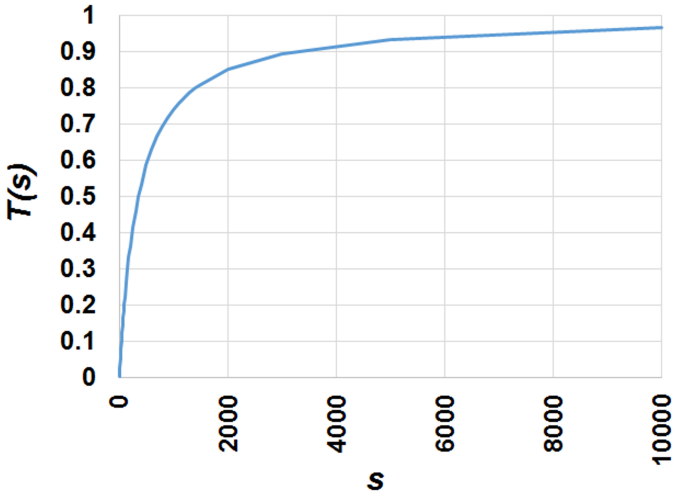


Figure 5.4: Function used to normalize the message sizes (B=350).

5.4.1 Metric and Normalization of Message Sizes

In Section 5.2.4, we explained that a metric distance must be defined in order to assess the similarity between two observed messages, and that such a metric distance should not be translation invariant, but more tolerant toward large messages than toward small ones. In other words, it must take into account the fact that the same byte-difference is more significant between two small messages than between two big ones. The bigger the message, the less significant is the absolute byte-difference.

In the proposed approach, the message sizes are "normalized" into the $]-1,1[$ space using a transformation that permits to scale the difference between two messages, in terms of bytes, relatively to the sizes of these messages. The transformation, T, is defined as:

$$\begin{aligned} T(s): \quad \mathcal{Z}^* &\rightarrow \quad]-1,1[\\ s &\rightarrow \quad s' = \frac{s}{(B+|s|)} \end{aligned} \tag{5.13}$$

where B is a positive constant that represents the middle of the space, that is, a threshold for considering a message as big or small. A typical value, as described in the literature, is around 500 bytes. In our experience, any value from 300 to 600 bytes would not dramatically change the results. Figure 5.4 graphically shows the proposed rescaling function.

After rescaling the sizes of the messages, a metric distance, $D()$, between two messages with sizes s_1 and s_2 is derived from their signs and the absolute difference between their normalized sizes, as:

$$D(s_1, s_2) = \begin{cases} 1 & if\ s_1 \cdot s_2 < 0 \\ \left| \frac{s_1}{(B+|s_1|)} - \frac{s_2}{(B+|s_2|)} \right| & otherwise \end{cases} \tag{5.14}$$

Notice that the distance between a negative value and a positive one is set to the maximum of 1. This property is motivated by the fact that, in network communication, two messages flowing in opposite directions are semantically different even if both have similar sizes.

The defined distance is a metric distance that verifies the following properties:

$$\begin{array}{lll} Range & \forall x, y \in \mathcal{Z}^*, & D(x,y) \in [0,1] \\ Identity & \forall x, y \in \mathcal{Z}^*, & D(x,y) = 0 \Leftrightarrow x = y \\ Symmetric & \forall x, y \in \mathcal{Z}^*, & D(x,y) = D(y,x) \\ Triangular\ inequality & \forall x, y, z \in Z^* \mathcal{Z}^*, & D(x,z) \leq D(x,y) + D(y,z) \\ Translation & & \text{D is not translation invariant} \end{array} \tag{5.15}$$

This metric distance permits to assess the degree of similarity between any pair of messages from the point of view of the application-level protocols that generate them.

5.4.2 Gaussian Mixture Model

Once a distance with the desired properties is set, a clustering of the observed message sizes is made in order to account for different methods of a protocol using this distance. In this section, we propose and justify a model for the Probability Distribution Functions (PDFs) of the observed message sizes. There are three observations that drove the derivation of the model; in what follows we will recall these observations and discuss the derivation of the model.

- *Observation 1 (re-statement).* Application layer messages are generated from methods defined in the design of application-level protocols. Each observed message is an occurrence of one of these methods.

- *Observation 2.* Each method generates similar message sizes with some reasonable amount of randomness. This similarity is best expressed in the sense of the metric distance defined previously, which is more tolerant toward large messages than small ones. Accordingly, we assume that the normalized message sizes generated from a method follow a normal distribution, characterized by a mean, μ, and a standard deviation, σ. Occasionally, a few methods, usually encountered in signaling schemes (i.e. BitTorrent handshaking), generate fixed size messages ($\sigma = 0$).

- *Observation 3.* Given an application and a state, the observed message may be generated from any of the application's methods, with some probability for each. For example, the HTTP-GET-request is generally more frequent than the HTTP-POST-request; an HTTP-GET-response is not likely to occur as the first message of an HTTP flow, etc.

Combined, observations 2 and 3 imply that the message-size distribution is close to a multi-peak Gaussian mixture. For a given application/state, each peak or cluster represents a method, with an underlying mean, μ, standard deviation, σ, and a weight, w, which is the probability of occurrence of the method at the given state.

5.4.3 Profiling the Applications

A supervised training approach for the proposed modeling would be based on the enumeration of the methods for each application. With enough samples, each method would be analyzed individually and its parameters (μ, σ, w) estimated. However, this task is likely impractical for several reasons being the most relevant one the need for a huge volume of traffic labeled accordingly. Therefore, an unsupervised approach based on clustering is designed to capture the essentials of the most frequently occurring methods.

The aim of the clustering is, given an application, to provide a means to estimate the number of methods and their statistical characteristics in an unsupervised way. This proposal is similar to that in [Jaber, 2012], in which the authors used K-means to find a global set of clusters from the packet sizes coming from all the applications, and then to assign a probability to each (application, state, cluster) triplet.

In our proposed model, a cluster is presumably associated to each individual method of each application protocol, and the training uses Expectation Maximization (EM) of a Gaussian mixture. The training process, which extracts the clusters and their characteristics, is applied to each class separately (a model-per-class). In a sense, this is a "profiling" approach: the training process applies the profiling routine to each protocol, and extracts its Gaussian mixture PDF at each state. A few points are important to note here.

First, this approach is easily extensible to new classes. To add a new target class to the classifier, it is sufficient to apply the profiling routine to samples of the new class, without any involvement of the already profiled classes and their sampled data. This advantage is particularly important in the context of network traffic classification, as the number of network applications is usually high, and new applications appear regularly on the Internet. This contrasts to other approaches that use global profiles or models, as SVM or neuro-fuzzy learning, such as in [Rizzi, 2013], which require rerunning the training process on the entire sample set when a new class is added to the classifier. This argument justifies the use of a maximum likelihood approach, such as EM for

estimating the multi-Gaussian PDFs, though it is arguable whether SVM's "optimal separation" would yield a significant improvement on the classifier's accuracy. But this potential increase in the accuracy could be obtained at the cost of increasing the specificity of the obtained models in relation to the training dataset, which is not desirable.

Finally, the training approach is both supervised and unsupervised, but it should not to be confused with mixed supervised/unsupervised training. It is supervised in the sense that the flows are a priori labelled. On the other hand, it is unsupervised in the sense that in the profiling phase, there is no prior mapping of individual messages to the application's methods.

5.4.4 Clustering and Estimation of Gaussian Mixtures

As previously stated, the aim of clustering is to capture the essential methods and their characteristics for each application/state instance. In our experiments, the method used to estimate the clusters is a combination of K-means and Dempster's EM. However, any method for estimating Gaussian mixtures would be appropriate, such as a greedy learning method [Verbeek, 2003].

Given an application, n, and a position in the flow, l, the first stage of the training uses iterative K-means to cluster the message-size samples. As usual, the clustering is applied till a threshold for the distortion or a previously fixed maximum number of clusters, K, is reached. The result after the clustering is a certain number of $K_{n,l} < K$ meaningful clusters.

The $K_{n,l}$ meaningful clusters from K-means are used as a starting point for EM algorithm. Each cluster is considered as a cloud around a peak in the multi-peak normal distribution, and Dempster's EM algorithm [Dempster, 1977] is used to estimate for each cluster, $C_{i,n,l}$, the centroid, $\mu_{C_{i,n,l}}$, the standard deviation, $\sigma_{C_{i,n,l}}$, and the weight, $w_{C_{i,n,l}}$. The aim is to maximize the overall likelihood of all the observations in the training set.

A certain number of clusters may have all their samples lying exactly on the centroid, with a null standard deviation. This happens when some protocol methods have a deterministic message size, but it also might be the result of random sampling, with "missing data". To deal with this phenomenon without losing generality, and in order to ensure a Bayesian analysis, a minimum standard deviation, σ_{min}, is predefined. Thus, if for a given cluster, $C_{i,n,l}$, its standard deviation, $\sigma_{C_{i,n,l}}$, is lower than σ_{min}, σ_{min} is used as its standard deviation.

Given cluster $C_{i,n,l}$ and an observation s_l, the probability density that the observation belongs to this cluster, $P(s_l \in C_{i,n,l})$ or $P_{C_{i,n,l}}(s_l)$, can be computed as,

$$P_{C_{i,n,l}}(s_l) = N_c(D(s_1, \mu_{C_{i,n,l}})) = \frac{1}{\sigma_{C_{i,n,l}}\sqrt{2\pi}} \cdot e^{-0.5\left[\frac{D\left(s_1, \mu_{C_{i,n,l}}\right)}{\sigma_{C_{i,n,l}}}\right]^2} \tag{5.16}$$

where $N_c()$ is the normal distribution function.

From this, the probability of the observation being generated from the application n at position l can be evaluated as

$$P(s_i|g(n,l)) = \sum_{i=1}^{K_{n,l}} w_{C_{i,n,l}} \cdot P_{C_{i,n,l}}(s_l) \tag{5.17}$$

It is worth mentioning that the value space is split into two disjoint subspaces before applying the clustering: one for the positive values and another one for the negative ones. This "gap at zero" means that the sign of any observation is its top-level characteristic. Thus, two multi-peak Gaussians will be estimated: one for positive observations and another for the negative observations.

5.5 Experimental Results and Assessment

5.5.1 Test bed

As described in Chapter 3, an experimental setup is built to assess the different classification proposals, including various datasets of real traffic which are parameterized and classified using nDPI for its use as training, test, and validation sets. In that chapter, a parametrization intended for traffic classification was presented and applied to all the captured datasets.

However, to be able to compute the messages length, as required by the proposed method, additional traffic parameters need to be extracted during an extended parametrization phase. In particular, the 7 additional parameters shown in Table 5.1 have been obtained for each of the packets in the datasets to be used with this method.

These parameters help in determining the payload size inside IP packets, which is used in turn to calculate the application layer message's length, as required by the proposed classification method..

From these parameters, the sizes for the $L = 6$ first messages in each flow are obtained, according to the features extraction procedure. Finally, each flow is represented by a vector containing the flow identification parameters and the sequence of messages sizes.

Regarding the MSS condition (Section 5.3.1.1), it is important to note that we do not extract the MSS from the TCP handshaking packets, but we simply set it to 576 bytes, which is the smallest MTU specified in RFC-791.

Table 5.1: Packet parameters obtained for the message analysis based method

Name	VarType	Description
Application Layer Statistics		
IP_IHL	float	IP Header Length
IP_ITL	float	IP Total Length
Timestamp	float	Packet timestamp
TCP_OFFSET	float	TCP Offset timestamp
TCP_SEQ	float	TCP Sequence Number
TCP_ACK	Boolean	TCP Acknowledgement Flag
TCP_PSH	Boolean	TCP PUSH Flag

One of the requirements of the proposed classifier is to have at least two messages to classify a flow. Therefore, the flows that have only one message are ignored. As we stated earlier, this does occur in some protocols such as FTP, that use signalling on the main TCP connection and open other connections for large data transfers.

To evaluate the method, the CS-A3 in addition to the public PS-1 datasets were used. A set of independent experiments are to be made with increasing values for L. For each of these experiments, a random set of up to 1500 samples per application protocol is chosen from the dataset as the training set. But, as not all the applications present in the dataset have enough flows with the required minimum number of messages, especially when L increases, a strategy has to be applied to the number of flows in the training set.

Thus, in order to avoid overfitting and counter for the testing, the number of samples from an application in the dataset is, at most, half the number of available samples with a maximum of 1500. On the other hand, and in order to ensure a proper training, a minimum is also set to include the application in the experiments. This minimum is set to 500 samples for TCP applications and 10 for UDP applications, which leads to the selection of 18 different applications for the CS-A dataset. The number of training and testing flows per application is thus the same and is shown in Table 5.2.

With this setup, the obtained results are detailed next.

5.5.2 Experimental Results

In this section we show the experimental results obtained during the training and the classification phases.

5.5.2.1 Training Phase Results

As previously mentioned, the outcomes of the training process are the clusters and PDFs associated with each (application, state) couple. Thus, the first step is to cluster message sizes on a per application and per state basis. As an

Table 5.2: Number of sample flows for applications in the CS-A dataset

TCPapp	#flows	#samples	UDPapp	#flows	#samples
BitTorrent	21522	1044	iMesh	386	100
iMesh	1090	500	Pando	446	200
MSN	21241	1205	MSN	800	300
SSL	224646	1480	NTP	616	300
HTTP	2954253	674	RTP	217	100
Gnutella	2876	839	Gnutella	109	98
Oscar	1787	739	Stun	1738	300
POP3	5195	1500	NetBios	516	200
FTP	1892	679	SNMP	25	10
SMTP	2040	800			

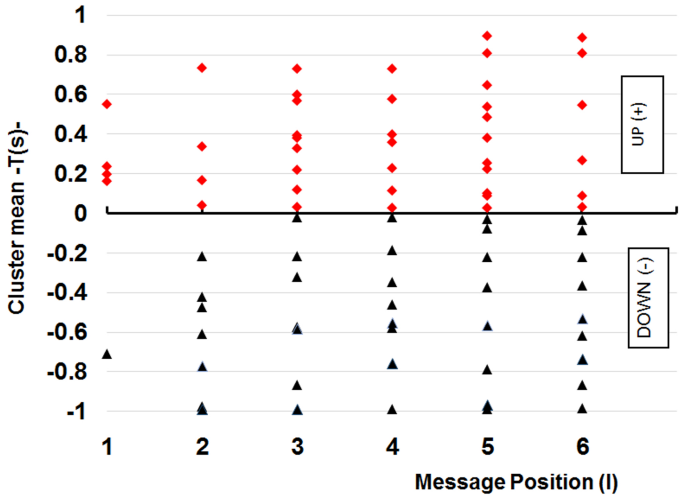
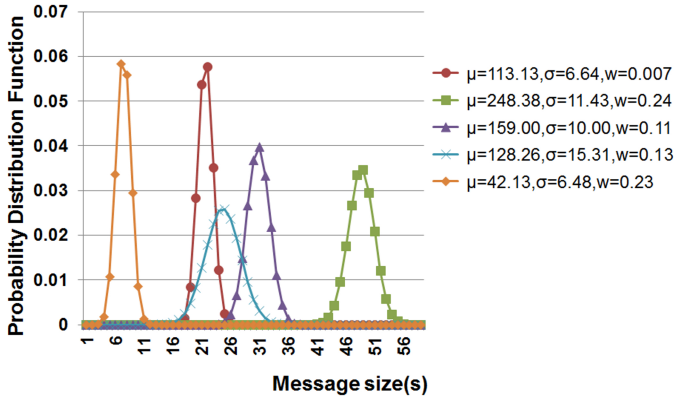


Figure 5.5: Mean message size (normalized) for the clusters obtained for BitTorrent ($L = 6$).

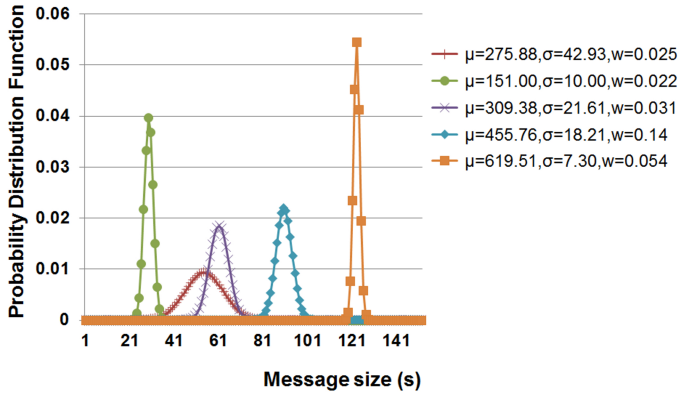
example, Figure 5.5 shows some preliminary results for the clusters obtained for BitTorrent considering the first six message positions (L=6) and a maximum of 12 clusters ($K = 12$) per state.

As shown in Figure 5.5, clusters belonging to the same applications may not be sparse when independently considering different message positions and directions (e.g clusters 5th and 6th for the up direction for $L = 3$). This reveals the number of maximum considered clusters, K, and their associated standard deviations, σ, as relevant factors for a proper clustering of the data.

Thus, some preliminary experiments were made in order to evaluate the impact of K and σ_{min}. First, many values of K were tested. Our experience showed that the method is not highly sensitive to the choice of K. The typical



a)



b)

Figure 5.6: PDFs associated with a) BitTorrent and b) HTTP for $l = 2$.

value of K, which is used in the experiments detailed in this paper is $K = 12$.

On the other hand, the exact value of σ_{min} is not critical, as our experiment showed that the results are resilient to the choice of *mindev* in a wide range of values, from 10^{-12} to 10^{-6} (the exact significance of these values derives from the defined metric distance).

As explained in Section 5.4.2, once the clusters are set, the parameters for the associated Gaussians are obtained. For illustration purposes Figure 5.6 shows the set of PDFs associated with BitTorrent and HTTP applications for the second message position ($l = 2$).

As shown in Figure 5.6, the peaks of the Gaussians (centroids) together with their deviations partially overlap at some cases. This is an indicator of the goodness of the obtained clusters and the appropriateness of a multimodal approach by using multiple Gaussians. On the other hand, an additional relevant observation is related to the final target of the system, that is, to the

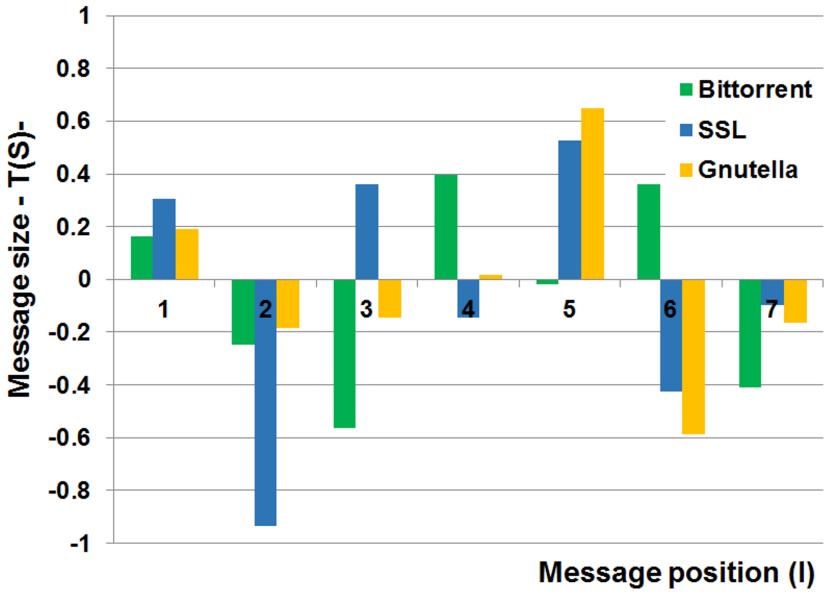


Figure 5.7: Sequences of application message sizes in both directions for sample BitTorrent, SSL and Gnutella flows.

classification capabilities. Thus, the distributions obtained for BitTorrent and HTTP are clearly different, which points to the fact that message sizes at the second position in the flows can discriminate between both applications. However, as mentioned previously, the discrimination among different applications is made based on a Markov model which accounts for the different message sizes at different initial positions in the flow, which is expected to contain more discriminative information than the individual sizes at a given position. To further illustrate this idea, Figure 5.7 shows application message sizes for sample BitTorrent, SSL and Gnutella flows at different message positions.

As shown in Figure 5.7, message sizes for those applications are similar at particular positions for some of them (e.g. for $l = 1$ for all applications, $l = 5$ for SSL and Gnutella) while being clearly different at others. Thus, considering message sizes at different positions and in both directions can discriminate between both applications, as intuitively observed in the graph trends of Figure 5.7. To prove the discriminative power of the proposed model for different applications, we have run many experiments using different datasets, as detailed next.

5.5.2.2 Classification Phase Results

In this section we show the results of the classification method on 18 applications, 10 of which use TCP and 8 use UDP.

Various metrics [Khalife, 2014] can be used to evaluate classifiers. Basic

ones include the rates of TP, TN, FP and FN. Ideally, a good classifier is the one that maximizes TP and TN rates (the overall rate of correct classifications) while minimizing FP and FN rates (the overall rate of incorrect classifications).

To reflect the correlation with FP and FN, combined metrics are commonly used in multi-classification scenarios: precision (or accuracy) and recall (or sensitivity). Precision is the ratio of traffic instances correctly classified as class A to the total number of instances classified as class A. Recall is the ratio of traffic instances correctly classified as class A to the number of actual class A instances.

At this stage, it is important to explain that according to our experience, the correct assessment of the classification method should not be based on overall evaluation metrics (e.g. overall accuracy). This is especially true for datasets that have largely unbalanced numbers of elements from each class. This is the case in the considered dataset, which contains an overwhelming percentage of HTTP flows. In these situations, assessing a classification method based on the overall percentage of correctly classified flows is misleading, because it favors any classifier that is biased towards the most frequent application. Since the aim of a classifier is to detect with good percentage all the applications, a better assessment should be based on the worst case evaluation metric, that is, to assess the application that exhibits the least such metric.

For these reasons, we choose different metrics (Figures 5.8 through 5.10) to evaluate our proposed classification model. These include precision and recall. We also consider classification results for various scenarios: overall applications, TCP and UDP applications, per application and worst case application.

As a first insight into the results, in Figure 5.8 we show the averaged recall, that is, the average of the recall rates for all the considered protocols, as a function of the number of observed messages per flow for TCP and for UDP applications. As shown, the recall increases up to around 96% for TCP applications when considering at least $L = 3$ messages. Also in Figure 5.8 we can notice a slight decrease in the accuracy for $L = 6$, which needs some analysis. On the other hand, UDP average recall rates exhibits a bigger performance with a maximum of 99.02 for $L = 4$.

As previously argued, these measures are not the best ones to assess the performance of the system, as they assume equal relevance to any application and a balanced dataset, which is not the case. Therefore, we consider the confusion matrices from which we derive some more meaningful figures. Thus, Table 5.3 shows the confusion matrix for TCP both as absolute values and relative to the number of samples in the test set for each application when using $L = 3$.

As can be observed, there exist a non-negligible number of classification errors (Table 5.3.a), mostly related to HTTP and SSL. Nevertheless, when the percentages for the number of flows in each class are considered, the results show high recall rates, with a minimum of 88.92 for MSN. The results are similar for higher values of L, except minor differences for $L = 6$, as de-

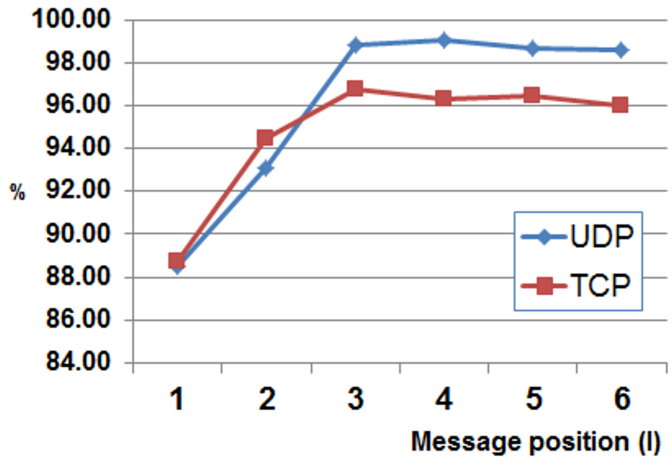


Figure 5.8: Average classification recall for TCP and UDP applications.

picted in Figure 5.9.a), in which maximum, minimum, overall and dominant (HTTP) recall rates are shown. It is noticeable a slight drop in HTTP recall for $L = 6$ which also induces a slight decrease in the overall recall for $L = 6$ when compared with that for $L = 5$. As previously mentioned, the number of training samples drops with L for some applications, as the number of available flows with at least this number of messages decreases. For $L = 6$, there are three applications with a relatively low number of flows for training (below 300), when compared with those available for the other applications, which is clearly degrading the quality of the representations for these applications and introducing classification errors.

On the other hand, Table 5.4 shows the confusion matrix for UDP applications, both as absolute (Table 5.4.a)) and relative (Table 5.4.b)) values. From these tables it is clear that the proposed system performs even better for UDP applications, with less non-null values out of the diagonal. The recall rates as a function of L are shown in Figure 5.9.b. As in the TCP case, there is a slight drop in the recall for $L = 6$, which can be again explained by a bigger decrease in the training samples from some of the applications. In this case, it is the number of samples from Gnutella who drops from more than 200 for $L = 2$ and $L = 3$ to below 100 for $L = 6$, while the decrease for the other applications is not that big in comparison.

It is relevant to mention that the recall rate reaches up to 98.39% after $L = 3$ messages are considered (recall that in the UDP case the messages are the packets themselves). This suggests that most UDP flows can be classified very quickly, being this an early classification method.

A different insight into the results can be derived from the precision values. As shown in Figure 5.10, despite the high recall rates, the values for the precision are not high for all the applications. This is clearly related to the highly

Table 5.3: Confusion matrices for TCP applications for $L = 3$: a) Absolute number of samples, b) Percentage of samples relative to the total number of application samples (row)

Classified as → **Input class ↓**	BitTorrent	iMESH	MSN	SSL	HTTP	Gnutella	Oscar	Mail_POP	FTP	Mail_SMTP
BitTorrent	15023	0	108	16	324	1	15	0	0	0
iMESH	0	1085	4	0	0	0	1	0	0	0
MSN	106	0	15760	92	1727	22	12	1	3	0
SSL	838	0	580	216551	3715	15	1433	0	0	0
HTTP	19766	1	19233	7764	152705	3172	60	0	4780	0
Gnutella	1	0	4	2	104	1911	0	0	0	0
Oscar	0	0	1	5	6	0	1654	0	0	0
Mail_POP	0	0	3	0	9	0	0	5122	63	0
FTP	0	0	0	0	14	1	0	9	1328	0
Mail_SMTP	2	0	0	0	28	0	0	3	13	1995

a)

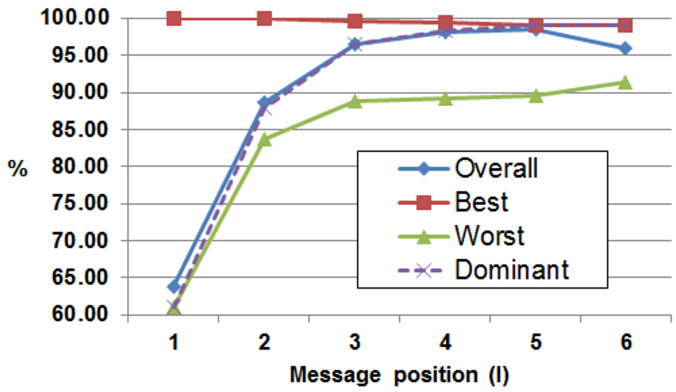
Classified as → **Input class ↓**	BitTorrent	iMESH	MSN	SSL	HTTP	Gnutella	Oscar	Mail_POP	FTP	Mail_SMTP
BitTorrent	97.00	0.00	0.70	0.10	2.09	0.01	0.10	0.00	0.00	0.00
iMESH	0.00	99.54	0.37	0.00	0.00	0.00	0.09	0.00	0.00	0.00
MSN	0.60	0.00	88.92	0.52	9.74	0.12	0.07	0.01	0.02	0.00
SSL	0.38	0.00	0.26	97.05	1.66	0.01	0.64	0.00	0.00	0.00
HTTP	1.25	0.00	1.22	0.49	96.54	0.20	0.00	0.00	0.30	0.00
Gnutella	0.05	0.00	0.20	0.10	5.14	94.51	0.00	0.00	0.00	0.00
Oscar	0.00	0.00	0.06	0.30	0.36	0.00	99.28	0.00	0.00	0.00
Mail_POP	0.00	0.00	0.06	0.00	0.17	0.00	0.00	98.56	1.21	0.00
FTP	0.00	0.00	0.00	0.00	1.04	0.07	0.00	0.67	98.22	0.00
Mail_SMTP	0.10	0.00	0.00	0.00	1.37	0.00	0.00	0.15	0.64	97.75

b)

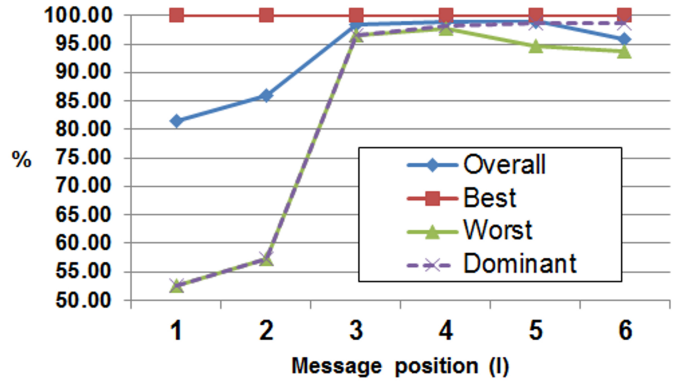
unbalanced nature of the dataset that makes the results for the precision of those less frequent protocols highly sensitive to even small error rates for the dominant ones.

As a summary, the results show that using the proposed method it is possible to correctly classify up to around 98% of the TCP flows and 99% of the UDP sessions by only analyzing the first 3 to 5 messages.

In order to check the specificity of the obtained distributions, we also



a)



b)

Figure 5.9: Classification recall as a function of L: a) TCP b) UDP.

tested the models by applying them on the m57-patents public dataset [Corpora, 2009], where we applied the trained distributions obtained from our own dataset and applied them to detect the applications available in common with the m57-patents in sufficient number. These are: SSL, HTTP and SMTP. Again, the overall classification results using our proposed model reached 96% of recall and 85% of precision. It is remarkable that no additional training nor tuning was made, which points to a good generalization of the obtained models.

5.6 Complexity Analysis

As stated earlier, a key advantage of the classifier is that each application is profiled standalone without any involvement of other applications, which is ideal for extensibility, and particularly important in the context of network traffic

Table 5.4: Confusion matrices for UDP applications for $L = 3$: a) Absolute number of samples, b) Percentage of samples relative to the total number of application samples (row)

Classified as → **Input class ↓**	iMESH	Pando	MSN	NTP	RTP	Gnutella	STUN	NETBIOS
iMESH	412	0	0	0	1	0	9	0
Pando	0	448	0	0	0	0	0	0
MSN	0	0	813	0	0	0	0	0
NTP	0	0	0	879	0	0	0	0
RTP	0	0	0	0	218	1	1	0
Gnutella	0	0	0	0	0	156	2	0
STUN	0	0	18	2	3	43	1906	1
NETBIOS	0	0	0	0	7	0	0	536

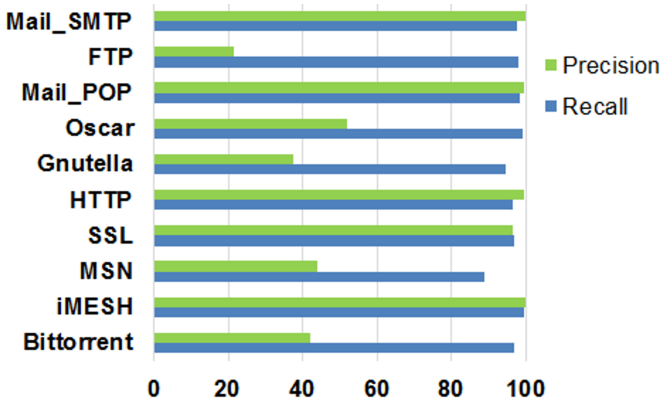
a)

Classified as → **Input class ↓**	iMESH	Pando	MSN	NTP	RTP	Gnutella	STUN	NETBIOS
iMESH	97.63	0.00	0.00	0.00	0.24	0.00	2.13	0.00
Pando	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
MSN	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
NTP	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
RTP	0.00	0.00	0.00	0.00	99.09	0.45	0.45	0.00
Gnutella	0.00	0.00	0.00	0.00	0.00	98.73	1.27	0.00
STUN	0.00	0.00	0.91	0.10	0.15	2.18	96.6	0.05
NETBIOS	0.00	0.00	0.00	0.00	1.29	0.00	0.00	98.71

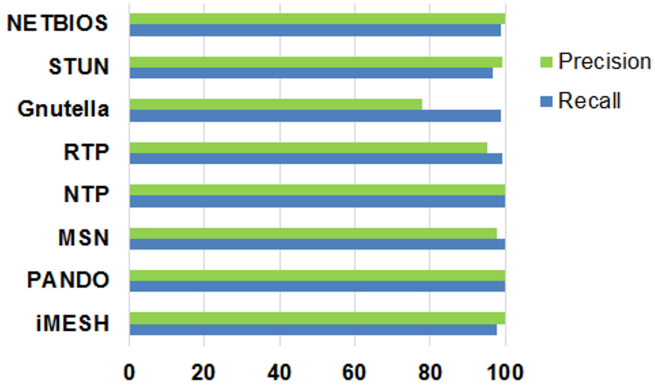
b)

classification. This is also important from the point of view of the computational complexity of the proposed method. But, previous to any cost analysis, it is important to distinguish between the two modes of operation of the system, that is, between training and classification. As previously mentioned, the training is made only once for each application, which means that the cost of training is linear with the number of applications (assuming equal number of samples in each class) and, more importantly, that it will not be the critical cost for the real usage of the system. On the contrary, in the normal usage, each time a flow is to be classified it should be analyzed by the system, this being the relevant cost for the effective deployment of the system.

Regarding the cost associated to the training, it is based on the estimation of Gaussian mixtures after a clustering of the data.



a)



b)

Figure 5.10: Precision and recall values for each application ($L = 3$): a) TCP, b) UDP.

According to [Verbeek, 2003], the associated computational complexity is $O(k^2 s)$, being k the number of clusters and s the number of samples. According to Equations 5.7 and 5.17, a multipeak Gaussian with up to $K_{n,l}$ peaks is evaluated for each application (n) and state of the model (l) during the training phase. This means that $K_{n,l}$ clusters have to be estimated per application.

Therefore, the complexity of the training phase is $O((KL)^2 Ns)$, being:

- N the number of considered applications,
- L the number of messages analyzed from each flow,
- K the maximum number of clusters, and
- s the number of sampled flows per application.

The evaluation phase consists simply in evaluating the likelihood of the message size vector versus the Gaussian mixtures and finding the maximum, so the computational complexity to classify a flow is $O(NLK)$.

Finally, the storage requirements are quite limited, as each flow can be loaded alone for evaluation. Thus, only the N models need to be stored. As each model consists on LK Gaussians, each one with 3 parameters -Equations 5.16 and 5.17-, the required storage is $O(3NLK)$.

5.7 Conclusions and Future Work

In this chapter a new blind network traffic classifier has been presented. The classifier uses the sizes of the initial messages exchanged between the hosts involved in the communication as inputs. The classification is flow-based and can be considered an early classification method, as the number of messages required for the classification can be kept low. Unlike other similar approaches, this work focuses on the messages, not the packets, that is, which is considered a differential characteristic of a protocol is the sequence of sizes of the first messages, not the sizes of the initially exchanged packets. Although both approaches will be almost the same for protocols with small message sizes, key differences may appear due to the potential segmentation of the messages into many packets.

Another differential characteristic of the proposed system is the use of multimodal distributions in an attempt to summarize all the possible methods included in a protocol in a single model. Thus, the classifier will consist of as many models as different applications it is able to detect. In this sense, previous similar works considered a reduced number of protocols (around 4-6 protocols) while in our proposal up to 18 different applications have been explored.

The experimental results are promising and an improvement over similar systems has been demonstrated. Nevertheless, more extensive experiments using bigger datasets are required in order to be able to improve the system, increasing the confidence and representativeness of the models and enabling the use of new and better heuristics. This is not an easy challenge, as the data to be used has to be properly labeled and has to be big enough. In our experiments, more than 200 GB of labeled data has proven to be insufficient for some protocols and only 18 of them could be used with some confidence.

Another relevant challenge for the real usage of the system is related to the completeness of the models and the classification of a flow not belonging to any of the trained classes. In this case, it is necessary to develop a rejection mechanism.

We also suggest as candidate for future work the tracking of host-to-host activities in order to affect accordingly the prior probabilities of the Bayesian classifier. We expect interesting improvements from this approach, though at

the cost of acceptable computational overhead.

Further handling the dependencies of the message sizes at different positions is another candidate for exploration.

Chapter 6

Host-Based Multi-label Classification using Motifs

This chapter addresses blind host-based classification, focusing on network hosts as the classification object of interest.

As mentioned in Chapter 1, classification methods are implemented as part of a system designed to detect and respond to security violations and network monitoring in computer networks. Managing today's enterprise networks is increasingly costly since a typical medium sized network is supposed to generate thousands of traffic flows and millions of packets in daily operation. Monitoring the network at too fine-grained levels (e.g. packets or flows) has obviously scaling problems and might become infeasible for large networks.

Consequently, network administrators need to extract some structure and information from their networks at larger levels of granularity, which helps them better in analyzing their security logs and in adjusting their security policy [Verma, 2009]. Adequately monitoring a network at the host level (as identified by IP addresses) can expose important parts of the logical structure of a network and the information flows which simplifies network management and protection tasks.

For these purposes, host-based traffic classification systems are considered a promising approach and can be regarded as a good tradeoff between too fine-grained and too coarse-grained (e.g. host-communities) in today's network management and security solutions.

However, the literature review (Chapter 2) shows few works based on host classification. Most of these follow a single-label classification model where a host is assumed to contribute in a single application. As will be demonstrated later in this chapter and easily argued, this mode does not reflect real communication scenarios for most computer communications examined throughout public and locally captured traffic traces. A typical behavior implies that the networking activity of a host includes many different protocols and thus the classification should be based on multi-label approaches.

In this scenario, multi-label classification discipline consists of assigning a set of application labels, instead of a single label (single-label mode), to each classified object. To the best of our knowledge, none of the existing host-based classification works were evaluated in multi-label contexts.

The aim of this chapter is thus to extend host-based classification models to handle multi-label contexts, simulating real case scenarios, where multiple applications are simultaneously in use by the same network host. The novelty of this proposal, by the time of this writing, consists of tackling host-based classification as a multi-label classification problem.

For this purpose, a classifier of choice [Allan, 2008] is first assessed as the native host-based classification system, which is then improved at different levels before being finally extended to handle multi-label classification.

The technique used in [Allan, 2008] relies on graph mining where special patterns of host communications, called *motifs*, are detected within the traffic communication graph. This choice is motivated by the promising results associated with the graphical techniques (Chapter 2) used in modeling host behavior, and the ability of graphs to capture interactions or relationships between elements. Thus, the core of the system and, therefore, the classification of the traffic is based on the interactions between hosts.

As a result, the goals of these chapter are three-fold:

- First, to set and assess a reference system. For this, the native host-based method in [Allan, 2008] relying on motifs is reproduced as literally as possible as described in the original work, though tested on different traffic and application sets.

- Second, to refine and evolve the basic reference system to improve single label classification. The native method is then criticized and some potential improvements, at different levels, are presented. In order to assess the real capabilities of the method, the proposal aims to enhance many of the elementary processes described in the reference work (e.g. parametrization, graph mining, etc.) together with the validation technique.

- Third, to propose and evaluate a multi-label graph-based and host-based traffic classification system. Thus, the improved host-based classification model developed in the previous step is extended at different levels to handle multi-label classification mode. Here, we reveal the main purpose of this chapter, which consists of studying the general applicability of host-based methods in multi-label communication scenarios.

To achieve these goals, experiments are throughout this chapter conducted in two modes: (i) Single-label, where the native and improved methods are evaluated; and (ii) Multi-label, where the host-based model is extended and evaluated based on the improved method. Before getting into the implementation details, a quick review of similar works in the literature is explored next.

6.1 Related Work

A review of the literature on traffic classification was introduced in Chapter 2, including most relevant graphical and host-based classification methods. In this section we review those works in relation to the proposal presented in this chapter.

Host-based classification methods often follow behavioral approaches (Chapter 2) using attributes that can be expressed in several ways: host interactions at the application layer [Karagiannis, 2005]; connections and flows characteristics at the network and transport layers [Cheng, 2007]; graph mining [Allan, 2009]; and active profiling ([Trestian, 2010]).

Graphical techniques are often used to model hosts in computer networks for traffic classification purposes. First, a seminal work proposed BLINC [Karagiannis, 2005] which introduced the notion of graphlets depicting host interaction at the application layer. Then, social graphs [Iliofotou, 2007] extended this concept with traditional graph measures.

However, none of these early graphical approaches included advanced graph mining techniques and motifs [Milo, 2002].

Network motifs were first explored in [Allan, 2008] and [Allan, 2009] for traffic classification purposes. A classification model based on motifs is proposed in [Allan, 2008] where authors reported the ability of accurately identifying 85% of the hosts' activities over a protocol set covering AIM, DNS, HTTP, MSDS, NETBIOS, SSH and Kazaa. Compared to traditional graph measures, motif-based classification show an improvement in class recall (greater than 80%) according to the tested traffic datasets. It is remarkable that only one application was reported as active in each host for these datasets. Thus, this is a single label scenario. To the best of our knowledge, none of the proposed host-based traffic classification approaches is designed nor evaluated in a realistic scenario in which multiple applications are present per host (multi-label).

On the other hand, multi-label [Spyromitros, 2008] classification techniques are described for other disciplines as scene classification [Boutell, 2004] and bioinformatics [Wernicke, 2006a]). Multi-label classification algorithms can be split into two main categories [Tsoumakas, 2006]: (i) Problem transformation methods [Boutell, 2004], that transform the multi-label classification problem either into one or more single-label classification or regression problems; and (ii) algorithm adaptation methods [Zhang, 2006], that extend specific learning algorithms in order to handle multi-label data directly.

As an example, Multi-label KNN (ML-KNN) [Zhang, 2006] extends KNN by using the MAP to determine the set of labels for a sample based on statistical information associated with neighbor samples found in the training set. Binary Relevance (BR)-KNN [Spyromitros, 2008] uses BR problem transformation to provide faster classification in conjunction with the KNN algorithm.

In this chapter, a simple adaptation of the default single-label KNN algorithm is proposed to handle multi-label classification. Our main focus is to

present a proof of concept by applying host-based classification problems in the multi-label context[1].

Instead of MAP or BR, our proposed model relies on KNN output in regression mode, as detailed later in this chapter. The proposed model is an adaptation algorithm that explores the closest single-labeled samples and annotates the unknown host accordingly in multi-label mode.

6.2 Fundamentals of Graphs and Motifs

Exploring the theoretical background behind motifs and graphs is essential before detailing the concept of traffic classification based on motifs.

The concept of motifs [Milo, 2002] is derived from graph mining, which is a general scientific discipline commonly applied in fields as biological networks. Basic graph concepts are relevant to motifs and graph mining techniques.

6.2.1 Basic Definitions and Concepts in Graphs

Graphs [Kolaczyk, 2009a] are data structures used to model pairwise relations between objects. Specifically, a graph consists of vertices (or nodes), which are an abstraction of the objects, linked together by edges (or arcs) depicting a relationship of interest between the interconnected vertices. Typically, graphs are represented as a set of dots or circles for the vertices linked by lines or curves representing the relationships between vertices, that is, between the objects.

A basic example is shown in Figure 6.1 depicting a simple graph made by six vertices, illustrated by circles, and seven edges, illustrated by straight lines.

In a wide array of disciplines, data can be intuitively cast into this format. As a result, many practical problems and networks including biological, social and computer networks, can be represented by graphs. They can also be used to model many types of relations and processes in chemical, physical, biological, social, linguistics, and information systems. Thus, many practical problems can be represented by graphs.

As an example, social networks consist of individuals and their interconnections which could be any type of relationship. These can be represented by graphs and, subsequently, using graph analysis some properties or characteristics can be explored, e.g., influence graphs can model whether certain people can influence the behavior of others.

Another example of an application field can be found in chemistry, where a molecule can be modeled by a graph in which atoms are represented by vertices and the attraction between atoms represented by edges.

[1]Evaluating or comparing multi-label algorithms found in the literature is out of the scope of this thesis.

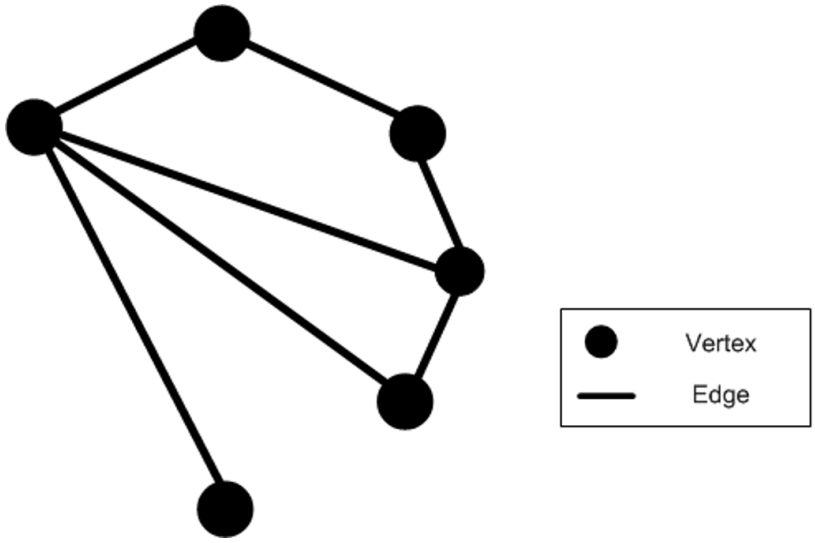


Figure 6.1: A simple graph composed by six vertices and seven edges

Another example is related to biology, where the graph theory can be useful when looking at tracking, for example, the spread of a disease, using a graph where a node represents regions and the edges represent species migration between the regions.

In computer science, graphs can be used to represent data organization, computational devices, the flow of computation, networks of communication, etc. They are particularly well suited to represent finite state automatons and to model sequences of events, as well as dependency relationships between elements. For instance, the link structure of a web site can be represented by a graph, in which the vertices represent web pages and directed edges represent links from one page to another. In network management and security, the most relevant to this thesis, graphs can be used to identify application types and to simulate, for example, the propagation of stealth worms on large networks.

In this study, graphs are used to model communication between hosts in computer networks, specifically for traffic classification purposes.

Particularly, in a graph representing a computer network, hosts or IP addresses can be represented by vertices and any observed exchanges[2] of information related to interactions of interest between two vertices [Jungnickel, 2013] can be represented by edges.

Formally [Kolaczyk, 2009b], a graph, G, is an ordered pair, $G = (V, E)$, composed by:

- V, a non-empty set of vertices (also called *nodes*), and

[2]The input used to build these graphs is mostly extracted from the transport and network layer headers.

- E, a set of edges (also called *links*) connecting vertices together. The elements in E are unordered pairs, u, v, of vertices, that is,

$$E \subseteq \{(u, v) | u, v \in V\} \tag{6.1}$$

The number of vertices in a graph, $N_v = |V|$, is the order of the graph, while the number of edges, $N_e = |E|$, is the size of the graph. As an example, the graph previously shown in Figure 6.1 has an order of $N_v = 6$, with six nodes, and a size of $N_e = 7$, with seven edges.

Without loss of generality, it is usual to label the vertices using integer numbers,

$$V = \{u_1, u_2, u_3, \ldots, N_v\} \tag{6.2}$$

and thus, the edges connecting vertices u_i and u_j are referred as $e_{i,j}$, that is,

$$E \subseteq \{e_{i,j} = (u_i, u_j) | u_i, u_j \in V\} \tag{6.3}$$

Given a graph, it is possible to consider smaller parts of it. Thus, a graph $H = (V_H, E_H)$ is a *subgraph* of another graph $G = (V, E)$ if $V_H \subseteq V$ and $E_H \subseteq E \cap (V_H \times V_H)$. Namely, a subgraph is composed by a subset of the vertices and a subset of the edges between those vertices in the original one. Analogously, an *induced subgraph* of G is a subgraph $G' = (V', E')$, where $V' \subseteq V$ is a pre-specified subset of vertices and $E' \subseteq E$ is the collection of edges to be found in G among that subset of vertices, that is E' contains all the edges $e_{i,j} = (u_i, u_j) | u_i, u_j \in V'$. Therefore, the concept of subgraph allows partitioning a given graph in many smaller ones.

Many properties can be defined to characterize graphs, apart from size and order. Some of them allow categorizing graphs in various classes, being one of the most relevant that related to the nature of the edges. Thus, a graph for which each edge in E has an ordering to its vertices, that is, edge $e_{i,j}$ is different from edge $e_{j,i}$, is named as a *directed graph* and the edge is a *directed edge* or *arc*. In this case, the edge $e_{i,j}$ goes from the *tail*, u_i, to the *head*, u_j. As opposite, a graph without an ordering in the edges is an *undirected graph* (e.g. graph in Figure 6.1).

Many types of graphs exist (labeled, colored, mixed, multi-graph, etc.). Figure 6.2 depicts a sample labeled, undirected graph graph made by eight vertices, labeled 1 through 8, with eleven edges.

In a directed graph (Figure 6.3), a *directed edge* is illustrated by an arrow (compared to straight lines in Figures 6.1 and 6.2). In a colored graph, vertices are decorated or *colored*, where different tags or, equivalently, *colors* are assigned to each vertex.

This later case is especially relevant for the present work, as different types of vertices are to be considered. Figure 6.3 is also an example of a colored,

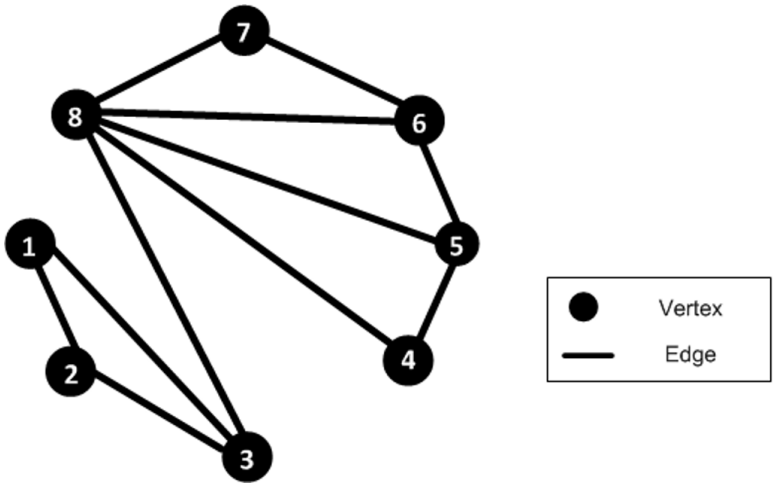


Figure 6.2: A labeled undirected graph composed by eight vertices and eleven edges

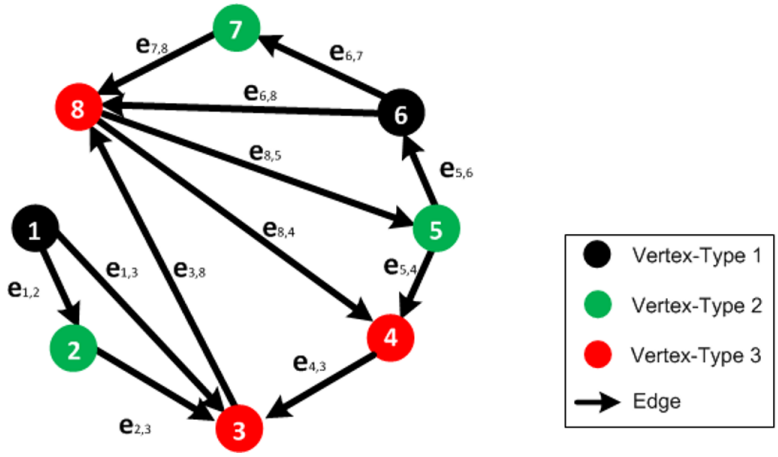


Figure 6.3: A labeled, directed and colored graph with 3 vertices colors, 8 vertices and 11 directed edges

directed graph. In this figure, vertices have one out of three types: 0 (black), 1 (red) or 2 (green).

Given two nodes in a directed graph, the existence of an edge in one direction does necessarily imply, nor it is equivalent to, the existence of an edge in the opposite direction. As an example, consider nodes 4 and 8 in the graph of Figure 6.3, though edge $e_{8,4}$ is directed from node 8 to node 4, edge $e_{4,8}$ does not show in the graph. According to the nature of the problem being modeled by graphs, the presence or absence of a directed edge - in one or the other direction has an important significance in analyzing to the type of interaction

between nodes, 4 and 8 in this example.

On the other hand, a vertex $u_i \in V$ is said to be *incident* on an edge $e_{k,l} \in E$ if u_i is an endpoint of $e_{k,l}$, which means that either k or l are equal to i. From this concept, the degree of a vertex, d_v, is defined as the number of edges incident to this vertex, v. For directed graphs, it is also interesting to distinguish between the *in-degree* of a vertex, $d_{in,v}$, and the *out-degree*, $d_{out,v}$, which count the number of edges pointing in towards and out from a vertex, respectively.

As an example, node 8 in the graph in Figure 6.3 has a $d_{in,8} = 3$, considering the number of incident edges with $d_{out,8} = 2$, considering the number of edges pointing out from vertex 8.

The connectivity of a graph can be expressed through the concept of adjacency. Two vertices $u_i, u_j \in V$ are said to be *adjacent* if joined by an edge in E, that is, if $e_{i,j} \in E$. Similarly, two edges $e_{i,j}, e_{k,l} \in E$ are adjacent if joined by a common endpoint in V, that is, one of the following conditions is met: $i = k$ or $i = l$ or $j = k$ or $j = l$.

Instead of relying on visual intuition, analyzing graphs is often formalized. For this purpose, different data structures are used to represent graphs depending on the structure, the nature of the problem and the used algorithm. In particular, it is possible and usual to represent and operate on graphs using matrices and matrix algebra.

Thus, given a graph $G = (V, E)$ its fundamental connectivity can be expressed through the adjacency matrix, A, composed by $N_v \times N_v$ elements, being the element $a_{i,j}$ equal to 1 when an edge exist from vertex i to vertex j, that is,

$$a_{ij} = \begin{cases} 1, & \text{if } e_{i,j} \in E \\ 0, & \text{if } e_{i,j} \notin E \end{cases} \tag{6.4}$$

Note that, for undirected graphs, the adjacency matrix is symmetric. Additionally, if no self-loops are allowed, values on the diagonal will be zero.

As an example, the adjacency matrix associated with the graph in Figure 6.3 , can be written as follows:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Some information and properties of G can be obtained from the adjacency matrix. Thus, for example, for an undirected graph, the row sum $A_{i+} = \sum_j a_{i,j}$ is simply equal to the degree d_i of vertex i. In the case of directed graphs, both the *in-degree* and *out-degree* of a vertex can also be calculated from the adjacency matrix as $A_{i+} = \sum_j a_{i,j}$ and $A_{+j} = \sum_i a_{i,j}$, respectively.

Another useful matrix capturing the fundamental structure in G is the *incidence matrix*, B, which is an $N_v \times N_e$ matrix with entries

$$b_{ij} = \begin{cases} 1, & \text{if } u_i \text{ is incident to edge } j \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

Some relationships can be set between A and B [Jungnickel, 2013], enabling graph analysis through matrix algebra, although this falls out of our scope.

Finally, in order to model complex problems, graphs should hold additional information. For this purpose, graphs can be extended to include auxiliary numerical values on its vertices, edges or both. For example, edges $e \in E$ are often accompanied by *edge weights* representing a magnitude of interest associated to the relationship between the adjacent vertices.

In many situations, as is the case of this work, there are no self-loops and thus, each diagonal entry in the adjacency matrix, A, is zero. In these cases, in some colored graph representations [Wernicke, 2006b], the diagonal value can be used to point to the vertex color. With this type of representation, the adjacency matrix associated with the example graph in Figure 6.3 can be written as follows:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

In a graph representing a computer network, the vertex color may be used to indicate different host role (e.g. client, server, and peer), directed edges may point to the direction of data flow between two hosts while edge weights can be used to represent traffic exchange properties, such as the size or duration of the data transfer.

In the study conducted throughout this chapter, directed graphs with colored vertices are used to model communications in computer networks for traffic classification purposes.

6.2.2 Graph Modeling using Motifs

By analyzing the distinguishing characteristics of graphs, the rules and patterns that hold for them, different graphs can be compared, and functions specific for each graph can be revealed. For this, some basic set of graph attributes such as connectivity degree, graph diameter, centrality, largest connected component, etc. can be used. This way, relationships between graphs can be defined and set.

Given a graph representation of a system it is useful to explore its characteristics and structural properties. This can be done through some evaluations ranging from the calculation of simple metrics summarizing the topological structure, both local and global, to the unsupervised extraction of complex relational patterns. A set of tools and techniques for such purposes are available [Wernicke, 2006b, Mfinder, 2013]. Thus, specific functions for each graph can be revealed.

One of the ways to analyze graphs is through partitioning. This refers to the segmentation of a set of elements, graphs in this case, into subsets. Formally, a *partition*, $\mathcal{C} = \{C_1, \ldots, C_K\}$, of a finite set, S, is a decomposition of S into K disjoint, nonempty subsets C_k such that

$$\bigcup_{k=1}^{K} C_k = S \tag{6.6}$$

Partitioning can be used to find groups of elements holding some common property, which can reveal information regarding the underlying relational patterns. In particular, in graph analysis, it is usual to apply partitioning to find "cohesive" subsets of vertices [Kashtan, 2004], that are well connected among themselves and, at the same time, are relatively well separated from the remaining vertices. Many graph partitioning algorithms exist, most of them seeking a partition, $\mathcal{C} = \{C_1, \ldots, C_k\}$, of the vertex set, V, of graph $G = (V, E)$ in such a way that the sets

$$E(C_k, C_{k'}) = \{e_{ij} | u_i \in C_k, u_j \in C_{k'}\} \tag{6.7}$$

of edges from vertices in set C_k to vertices in $C_{k'}$ is relatively small when compared with the set $E(C_k, C_k)$ of edges connecting vertices within C_k.

This way, through partitioning, a networking activity graph could be described as an ensemble of smaller pieces which could represent a fundamental interaction class if a suitable splitting function is defined. Nevertheless, we are not really interested in analyzing a graph but in finding the elementary pieces from which this graph is built. The underlying hypothesis is that, for

networking activity, the observed graph is the result of the mix of elementary interactions, each of them associated to a different application. Furthermore, the observed graph is only a sample of the underlying phenomena, as it is obtained at a given time and most probably in an incomplete way due to the own nature of networking activity. Thus, we should move from an analysis approach to a modeling one.

A model for a network graph can be described by [Kashtan, 2004]:

$$\{\mathcal{P}_\Theta(G), G \in \mathcal{G} : \theta \in \Theta\} \tag{6.8}$$

where $\mathcal{G}$ is a collection (or 'ensemble') of possible graphs, $\mathcal{P}_\Theta$ is a probability distribution for elements in $\mathcal{G}$, and θ is a vector of parameters.

Thus, the problem of modeling consists in obtaining the elementary graphs, $G \in \mathcal{G}$, a parametrization for each of these graphs, Θ, and the probability of appearance of each of the graphs in an observed network graph. The quality of the modeling mainly depends on how the probability distribution, $\mathcal{P}_\Theta(G)$ is chosen, for which many methods are proposed in the literature. A possible approach is based on random graph models, which are based on the idea of a graph being drawn "at random" from a collection $\mathcal{G}$.

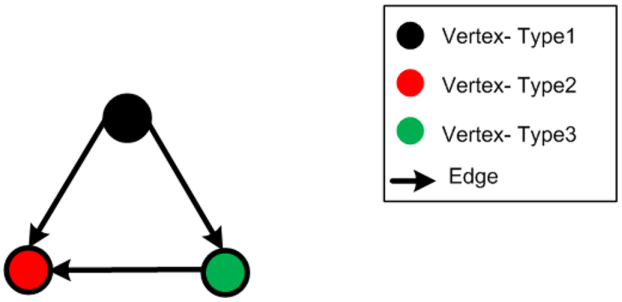
The term *random graph model* is typically used to refer to a model specifying a collection, $\mathcal{G}$, and a uniform probability $\mathcal{P}_\Theta()$ over $\mathcal{G}$.

Nevertheless, we are more concerned in identifying a representative enough collection of possible graphs, $G \in \mathcal{G}$, and associating them to the applications to be classified. Thus, we should focus on methods to select these graphs and evaluate their representativeness instead of obtaining the probability distributions.

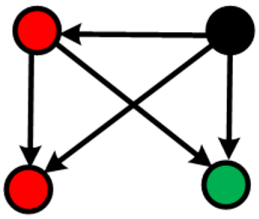
In this context, a possible way to uncover the structural design of a complex network is the use of mining techniques with graphs. With graph mining [Milo, 2002], the most relevant problem is to find a recurrent subgraph structure (called *motif*) in a given graph. The found subgraphs will constitute the ensemble of the model.

Motifs [Kashtan, 2004] are small subgraphs occurring far more frequently in a given network than in comparable random graphs. Thus, they are supposed to be the basic structural elements for broad classes of networks. An assumption behind the repetition of a topological substructure is that it is of a particular functional importance. Therefore, the underlying concept is that many of the complex interconnections that occur within a network are built up from frequently recurring patterns of basic structural elements, that are identified as the motifs.

Primarily, motif analysis has been used mostly in biological networks. For example, protein interaction networks link proteins which must work together to perform some particular biological function, which can be analyzed using motifs. Authors in [Milo, 2002] state that the appearance of network motifs at high frequencies suggests that they may have some specific functions in the



a)



b)

Figure 6.4: Motifs of order: a) $k = 3$ and b) $k = 4$

information processing, and thus in the application interactions, performed inside the network.

According to the previous definition, motifs are defined as recurrent and statistically significant[3] subgraphs (or patterns) of interconnections. Thus, motifs can be extracted by searching subgraphs occurring in complex networks at numbers that are significantly higher than those in similar randomized networks, as will be detailed next.

Motifs can be of different orders, k, according to the number of vertices, although they are supposed to contain a low number of them to be useful. Figure 6.4 depicts two examples of motif structures (with $k = 3$ and $k = 4$). As shown in this figure, vertices need not to be named in a motif representation, since they are regarded as recurrent patterns regardless of the vertex or edge names. Nevertheless, for directed and colored graphs, edge direction and vertex color are part of the motif structure, as shown in Figure 6.4.

Motifs, as any other graph, can be represented as matrices. Using the matrical representation in [Wernicke, 2006b], the adjacency matrix associated with the order-3 motif in Figure 6.4a) can be written as follows[4]:

[3] A statistically significant (or simply, significant) motif is a terminology used throughout this chapter to refer to motifs whose subgraph complies with some specific conditions.

[4] Assuming: vertex-type 1 (black color) is labeled node as 1, vertex-type 2 (red color) is labeled as node 2 and vertex-type 1 (green color) is labeled as node 3.

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 2 \end{pmatrix}$$

To detect the motifs in a given network, various algorithms and publicly available tools exist. According to the definition, it is necessary not only to obtain the set of subgraphs (potential motifs) to be considered but also to estimate their relative frequencies of appearance in relation to the possibility of a random appearance of this subgraph, that is, to assess their significance. For this, graph randomization techniques [Wernicke, 2006a] are used.

To address this problem, let's suppose that a graph, G_{obs}, derived from observations of some sort, e.g. the observed network communications, is obtained and that we are interested in some structural characteristic, $\eta()$, of this graph. In particular, we are interested in assessing whether the value $\eta(G_{obs})$ is "significant", in the sense of being somehow unusual or unexpected and thus, it is specific for the graph G_{obs}. Of course, this significance must be defined in comparison to an appropriate frame of reference, for which random graph models are often used.

For this, a collection of random graphs, $\mathcal{G}$, is defined, and the value $\eta(G_{obs})$ is compared to the set of values $\{\eta(G) : G \in \mathcal{G}\}$. If the observed value for the characteristic, $\eta(G_{obs})$, is considered to be extreme with respect to this set, this is considered as evidence that G_{obs} is unusual in having this value. More formally, a random graph model is used to create a reference distribution which, assuming a uniform likelihood of the elements in $\mathcal{G}$, is

$$\mathcal{P}_{\eta,\mathcal{G}(t)} = \frac{card\{G \in \mathcal{G} : \eta(G) \leqslant t\}}{|\mathcal{G}|} \tag{6.9}$$

Thus, if $\eta(G_{obs})$ results sufficiently unlikely under this distribution, it is considered as an evidence that G_{obs} is not a uniform draw from $\mathcal{G}$.

In this scheme, selecting the motifs in a given graph, G_{obs}, would imply the extraction of all the subgraphs and the analysis of their likelihoods through random graphs models, $\mathcal{G}_S$. Nevertheless, how to best choose $\mathcal{G}_S$ is a practical issue of some importance, since it can have a direct impact on the relevance of the results of such procedures. It is important that its elements preserve relevant structural properties of the graph under analysis, G_{obs}. Furthermore, we cannot generally hope to be able to explicitly enumerate all of the elements in $\mathcal{G}_S$, and therefore, we cannot expect to be able to calculate the previous probabilities exactly.

Therefore, the process of motif's discovery in a network can be split into three main phases, as depicted in Figure 6.5: the generation of an ensemble of similar graphs, $\mathcal{G}_S$, the enumeration and accounting of the potential motifs, and

finally, the selection of the significant ones from this set. This is implemented as [Milo, 2002]:

(1) Graph randomization, which consists of generating a set of random networks, $\mathcal{G}_S$, conserving some properties of the original graph. The set of all random graphs obtained at the end of this phase is referred to as the null-model.

(2) Motif extraction, in which the elementary subgraphs to be potentially considered as motifs as well as their numbers of occurrences are obtained from the null-model. In this regard, motif discovery algorithms can be based either on exact enumeration and counting of the existing subgraphs or on statistical sampling and estimations. Therefore, they can be roughly classified into:

   - Exact search algorithms (e.g. Grochow-Kellis [Grochow, 2007]) which are based on exhaustive searches, that is, counting each possible subgraphs. The major drawback of this approach is that the implementation of such algorithms is very computationally and time demanding.
   - Estimation-based algorithms, that takes the advantages of sampling by skipping some subgraphs during the enumeration and performs thus more efficiently. RAND-ESU [Wernicke, 2006a] is an example of an estimation-based algorithm, and is implemented in the mfinder-tool [Mfinder, 2013] and Fast Network Motif Detection (FANMOD) tool [Wernicke, 2006b].

(3) Significant motif detection, which consists in evaluating the statistical significance of each subgraph basically by comparing the number of its appearances in the null-model against that in the original input graph being analyzed. Some additional statistical measures, as will be detailed next, should be applied to select those subgraphs considered really significant and that will be listed as motifs.

At the end of the significant motif detection process, a set of motifs will be associated with the original input graph, G_{obs}, providing an ensemble, $\mathcal{M}$, for it. As previously stated, the input graph, G_{obs}, is supposed to be generated from the repetition and combination of subgraphs in this ensemble. Additionally, during the significance analysis of the motifs, a probability distribution for them is also evaluated. Thus, at the end of the process a model for the observed graph is set. Nevertheless, it is important to note that the model is only an approximate one, as not all the existing subgraphs are considered in $\mathcal{M}$.

Once an overview of the method for motif extraction is presented, some details and procedures for each of the involved steps are provided. Moreover,

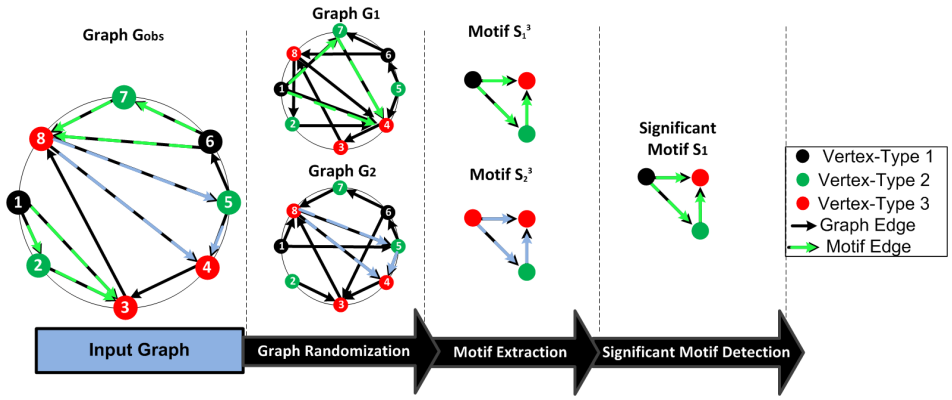


Figure 6.5: Schematic view of motif detection

an example of the motif extraction process, applied on the input graph of Figure 6.5, will be subsequently detailed considering each of the aforementioned steps.

6.2.2.1 Graph Randomization

The first step for motif detection is to obtain the null-model, which is defined by an ensemble of graphs, $\mathcal{G}_S$, obtained by randomizing the original input graph, G_{obs}.

As the purpose of this procedure is to obtain and evaluate the potential subgraphs, and as previously argued, the randomized graphs should be structurally similar to the original one [Milo, 2002]. For this purpose, the randomization process should be controlled with specific conditions. In particular, the randomized graphs should be isomorphic to the original one.

Formally, graphs $R = (V', G')$ and $E = (V, G)$ are isomorphic if there exists a bijection between the vertex sets of R and G, $f : V' \rightarrow V$ such that any two vertices u_i and u_j of R are adjacent in R if and only if $f(u_i)$ and $f(u_j)$ are adjacent in G. The mapping f is called an isomorphism between G and R.

Intuitively, isomorphic graphs are the "same", except for "renamed" vertices. When randomized graphs are isomorphic with the original graph, essential properties of the original graph (e.g. number of vertices, out-degree, etc.) will be preserved.

In the case of colored graphs, the isomorphism should also respect the colors of the vertices, that is, if the color of u_i is x, then the color of $f(u_i)$ should also be x.

One way to obtain isomorphic random graphs is through edge switching. Figure 6.6 illustrates an example of the randomization process through a series of edge switching operations in a colored directed graph.

As shown in Figure 6.6, edge switching should maintain the colors of the vertices to generate isomorphic graphs, which implies that edges should be exchanged only if the endpoint vertices have the same color. Thus, in the

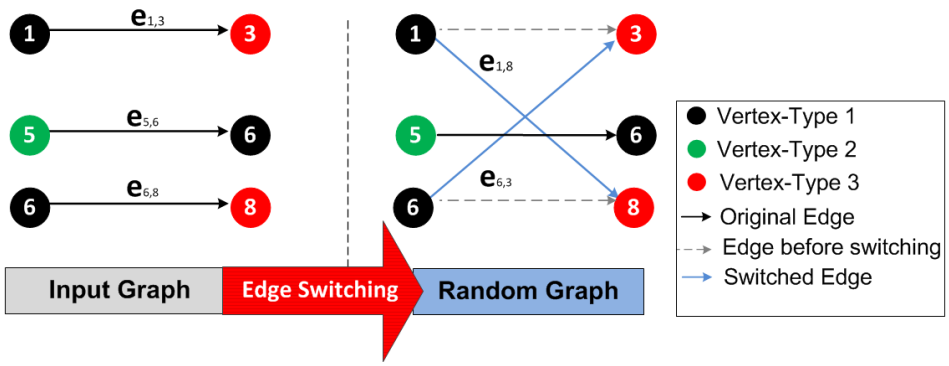


Figure 6.6: Edge-switching process with regard for a colored directed graph for generating random networks

example, edges $e_{1,3}$ and $e_{6,8}$ become respectively, $e_{1,8}$ and $e_{6,3}$ after switching, given that these edges have the same direction, and set an adjacency between vertices having the same color.

Another relevant question to generate isomorphic graphs is to maintain the direction of edges by switching edges with the same direction between endpoint vertices of the same color. Given this condition, the out-degree reflecting the number of bidirectional edges incident upon each vertex is maintained after randomization through edge switching.

By considering a set of random isomorphic graphs, the null-model is supposed to be independent of any structural function related to the original graph, G_{obs}.

Formally, there exist an ensemble, $\mathcal{G}_S$, of random graphs corresponding to the null-model associated to graph G_{obs}. Nevertheless, with the proposed randomization process, only a subset of N random graphs are considered from $\mathcal{G}_S$.

To illustrate this process, let's consider the randomization of input graph G_{obs} in Figure 6.5 as an example. In this example, the graph is randomized through edge-switching maintaining the direction of edges and vertices colors. For simplicity reasons, we show $N = 2$ isomorphic graphs.

In this example, random graph G_2 is obtained after switching edges $e_{1,2}$, $e_{1,3}$, and $e_{6,8}$ respectively, into edges $e_{1,5}$, $e_{1,8}$, and $e_{6,3}$, given that these edges have the same direction, and set an adjacency between vertices having the same color. The same applies to graph G_1 which was similarly obtained by switching edges $e_{1,3}$, $e_{1,2}$, $e_{2,3}$, $e_{7,8}$ and $e_{8,5}$ respectively, into edges $e_{1,4}$, $e_{1,7}$, $e_{2,4}$, $e_{7,4}$ and $e_{8,2}$. As an example of maintaining the out-degree for all vertices, vertex 8 has an out-degree equal to 2 in G_{obs} and in both random graphs. For the example illustrated in Figure 6.5, the main outcome of the randomization phase is the null-model as represented by set of random graphs, $\mathcal{G}_S = \{G_1, G_2\}$.

6.2.2.2 Motif Enumeration

The next step (Figure 6.5) is to mine the generated random graphs for significantly recurrent subgraphs that could be considered motifs. During this process, the number of occurrences of each subgraph in each of the mined graphs should be estimated, as it will be the basis for the selection of the significant ones in the next phase.

As previously mentioned some methods are available for the enumeration of the subgraphs in a given graph, both through exact search algorithms and sampling methods.

The complexity of the network graphs creates substantial computational challenges for the application of exact search methods. Thus, for a given choice of subgraphs order, k, it is necessary to search for each of the i possible k-vertex subgraphs and count its number of occurrences, N_i, both in the observed network G_{obs} and in each (or at least a large random subset) of the networks, G, in the null-model $\mathcal{G}_S$. But the number, L_k, of possible motifs of order k grows exponentially with k. For example, in a directed graph, there exists $L_3 = 13$ distinct types of connected, three-vertex subgraphs, $L_4 = 199$ four-vertex subgraphs, and up to $L_5 = 9364$ such five-vertex subgraphs.

In this scenario, graph sampling techniques can be used in overcoming this computational challenge. Specifically, if k-vertex subgraphs H are sampled in some fashion, then an unbiased estimate of the total number N_i of a given subgraph type i is just

$$\hat{N}_i = \sum_{H \text{ of type } i} \pi_H^{-1} \tag{6.10}$$

where π_H^{-1} is the inclusion probability for H.

Authors in [Kashtan, 2004] proposed a sampling method for the estimation of this count. Given a network graph, G, a single edge is first selected in a random way. Then, using the concept of link-tracing [Zhang, 2014], a new edge to a neighboring vertex is followed, where that edge is selected randomly from among those in G incident to the two vertices defining the first edge. This procedure is iterated in such a way that at the m-th stage we follow an edge randomly selected from among those incident to the collection of available vertices at the $(m-1)$-th stage, excluding those edges that have been already considered. When a total of k vertices have been included, the final sampled subgraph, H, is obtained by including in the set of edges any remaining unencountered edges among these k vertices. That is, H is defined to be the induced subgraph on the set of discovered vertices.

An ordered set of $k-1$ appropriate edges must be sampled, using this sampling strategy, in order to sample a given k-vertex subgraph H. Therefore, the probability of sampling H is the sum of the probabilities of all such possible ordered edge sets, that is

$$\mathcal{P}_H = \sum_{\sigma \in S} \prod_{e_j \in \sigma} \mathcal{P}(e_j | (e_1, e_2, \ldots, e_{j-1})) \tag{6.11}$$

where S is the set of all possible sets of $k-1$ ordered edges, σ is an element of that set, and e_j is the j-th edge in a specific element, σ.

The probability that subgraph H is included in a collection of n subgraphs sampled in this way can be evaluated as $\pi_H = 1 - (1 - \pi_H)^n$.

Thus, given the input network graph, G_{obs}, the null-model, $\mathcal{G}_S$, and a maximum order for the considered subgraphs, K_{max}, the output from this phase are three elements:

- an ensemble, $\mathcal{M}'_{K_{max}}$, composed by all the possible subgraphs ${S_i}^k$ of up to K_{max} vertices

$$\mathcal{M}'_{K_{max}} = \{S_i^k : 2 \leq k \leq K_{max}, 1 \leq i \leq L_k\} \tag{6.12}$$

  being L_k the number of different k-vertex subgraphs;

- a vector, $\overrightarrow{N}_{K_{max}}$, composed by the number of occurrences, ${N_i}^k$, of subgraph ${S_i}^k$ in G_{obs}

$$\overrightarrow{N}_{K_{max}} =< N_1^2, N_2^2, \ldots, N_{L_2}^2, N_1^3, \ldots, N_{L_3}^3, \ldots, N_{L_{K_{max}}}^{K_{max}} > \tag{6.13}$$

- A probability distribution, $\mathcal{P}_{\mathcal{G}_S}()$, for all the subgraphs S_i^k in $\mathcal{M}'_{K_{max}}$ derived from the null-model, $\mathcal{G}_S$, or alternatively (used in this chapter) a vector $\overrightarrow{M}_{K_{max}}$, composed by the number of occurrences, M_i^k, of subgraph S_i^k in $\mathcal{G}_S$. In this later case, only the frequencies of each subgraph can be evaluated.

To illustrate this process, let's consider the motif extraction related to graph G_{obs} of Figure 6.5 as an example.

In this example, the set of random graphs is $\mathcal{G}_S = \{G_1, G_2\}$. By applying RAND-ESU [Wernicke, 2006a], an estimation-based motif discovery algorithm (Section 6.2.2.2) for $K_{max} = 3$, some subgraphs are skipped during the enumeration and the result is two subgraphs (or motifs), namely $S_1^3 \in \mathcal{M}'_3$ and $S_2^3 \in \mathcal{M}'_{K_{max}=3}$, of order $k = 3$.

Together with each subgraph, statistical information related to the number of occurrence both in the input and random graphs are obtained. As per the example in Figure 6.5, the number of occurrences of subgraphs ${S_1}^3$ and ${S_2}^3$ in G_{obs} is represented by $\overrightarrow{N}_3 =< N_1^3, N_2^3 >=< 2, 1 >$. The number of occurrences, of subgraphs ${S_1}^3$ and ${S_2}^3$ in $\mathcal{G}_S$, the null-model, becomes $\overrightarrow{M}_3 = < M_1^3, M_2^3 > =< 1, 1 >$. For the example illustrated in Figure 6.5, the main outcome of the motif enumeration phase is the set of subgraphs or motifs $\mathcal{G}_S = \{S_1^3, S_2^3\} \subset \mathcal{M}'_{K_{max}}$ together with vectors $\overrightarrow{N}_3$ and $\overrightarrow{M}_3$ based on which most significant motifs will be elected.

6.2.2.3 Motifs Selection

The next step is to select as motifs, $\mathcal{M}$, only those subgraphs in the set of all the possible subgraphs, $\mathcal{M}'_{K_{max}}$, that are significant.

In order for a subgraph, S_i^k, to be considered statistically significant, and thus as a motif, many different parameters can be evaluated and some threshold is to be set for each of them [Priami, 2005]. The basic approach is to compare its number of occurrences, N_i^k, to the reference distribution, $\mathcal{P}_{\mathcal{G}_S}()$, or to the number of occurrences (used in this chapter) of this subgraph in the null-model, M_i^k. If the obtained value is found to be extreme, then the subgraph S_i^k is declared as a network motif.

Nevertheless, some parameters with different degrees of complexity are described in the literature to assess the statistical significance of a subgraph and the obtained set of motifs [Priami, 2005]. Some of them are:

- Number of random graphs in the null-model: The higher the number of generated graphs, N, during randomization, the higher will be the confidence on the subsequently detected motifs. Thus, it is necessary to establish a minimum number, N_g, of isomorphic graphs to be included in the null-model so as to consider the found motifs as really significant. Obviously, this number should depend on the size of the observed graph.

  In practice, in the context of networking activity graphs, a starting value of $N_g = 5000$ random graphs or above is commonly used [Mfinder, 2013, Wernicke, 2006b].

- Frequency of a subgraph: The frequency, F_i, of a subgraph, S_i, is the number, N_i, of occurrences of S_i in the original input graph, G_{obs}, in relation to its occurrences, M_i, in the null-model, G_s:

$$F_i = N_i / M_i \tag{6.14}$$

  A subgraph S_i is called recurrent (or frequent) in G_{obs} when F_i is above a predefined threshold value. Subgraphs that comply with the threshold condition are elected as *significantly* recurrent subgraphs or motifs.

  In practice, a starting frequency value of 10 is commonly used [Mfinder, 2013, Wernicke, 2006b].

  Even this is the basic measure to determine if a given subgraph is a motif or not, the confidence on this choice depends on the probability distribution for the subgraphs in the null-model. In other words, if all the subgraphs were equally probable in the null-model, the choice of the significant ones by simply looking at the observed frequencies would be appropriate. But, given that this distribution is not expected to be uniform, the confidence that an observed value is extreme for this subgraph is not the same for all of them when using a single threshold. Therefore,

it is advisable to use additional measures to check the "extremeness" of an observed frequency for a subgraph.

- Z-score: The *Z-score* is a common measure of a motif statistical significance. Formally, for a subgraph, S_i, the graph G_{obs} and a set of randomized networks G_s, the *Z-score* or $Z(S_i)$ is defined as:

$$Z(S_i) = \frac{F_i - \mu_{G_S}(S_i)}{\sigma_{G_s}(S_i)} \tag{6.15}$$

  where $\mu_{G_S}(S_i)$ and $\sigma_{G_s}(S_i)$ stand, respectively, for the mean and standard deviation of the frequency values for subgraph S_i across all the graphs in G_s. Obviously, to be able to evaluate this measure, the output of the motif enumeration phase should be the probability distribution, $\mathcal{P}_{G_S}()$, instead of just the number of occurrences of each subgraph, $\overrightarrow{M}_{K_{max}}$. The larger the value for $Z(S_i)$, the more significant (extreme) is the motif or subgraph S_i.

- The p-value: The *p-value* is a measurement in statistical hypothesis testing that can be considered in motif detection. It describes the probability of obtaining a result at least as extreme as the result observed, given that the null hypothesis, or expected outcome, is true [Allan, 2008].

  Given a graph, G_{obs}, a subgraph S_i of G_{obs} and an ensemble, G_s, of random graphs from G_{obs}, the *p-value*, $P(S_i)$ evaluates the probability of the null-hypothesis of S_i being more frequent in G_s than in G_{obs}, that is,

$$P(S_i) = \frac{1}{|G_s|} \sum_{R \in G_s} \delta_{kr}(F_i < F_i(R)) \tag{6.16}$$

  where δ_{kr} is the Kronecker delta function and $F_i(R)$ is the probability of the subgraph S_i to appear in graph R of the ensemble, F_i is its probability to appear in the original input graph, G_{obs}.

  For example, a *p-value* of 0.01 means that there is only a 1% chance of seeing a particular pattern (i.e. the motif subgraph) as many or more times in random graphs than is observed in the original one. Thus, a subgraph with *p-value* less than a threshold is another condition for a motif subgraph to be considered as significant.

  In practice, common values of *p-value* are 0.05 and 0.01 as reported in [Mfinder, 2013, Wernicke, 2006b].

- Persistence: Another relevant parameter is to evaluate the statistical persistence of each motif. Whenever the motif detection experiment for a given graph G_{obs} is repeated, a motif might not necessarily appear in all experiments due to the random graph set, G_s, obtained each time. For a given number of experiments, l, a persistent motif is defined as the one that is always present over the l experiments.

In this study, all of these parameters (N, *Z-score* and *p-value*) are used in order to consider a subgraph as a significant motif.

At the end of the motif detection process, the input graph, G_{obs} (and thereby the supposed structural components of the graph), will be associated with a set, $\mathcal{M}$, of N_m motifs:

$$\mathcal{M} = \{M_j : M_j \in M'_{k_{max}} : 1 \leq j \leq N_m\} \tag{6.17}$$

In some graph problems, the kind of functional activities associated with each vertex is the main focus. As such, a smooth representation of each vertex in function of the detected motifs is essential to describe the vertex interactions and thereby, the type of functions, within the graph.

To illustrate this process, let's consider the motif selection related to graph G_{obs} of Figure 6.5 as an example. For simplicity reasons, we rely in this example on evaluating whether the frequency value[5] is strictly above 1, $F_i > 1$, to consider a subgraph S_i as a significant motif.

As shown previously for this example, the set of subgraphs or motifs is $\mathcal{G}_S = \{S_1^3, S_2^3\}$. The question is for now to select which subgraph, S_1^3 or S_2^3 (or even both) abides to the frequency condition and is, thereby, a significant motif. By simply comparing vectors $\overrightarrow{N}_3 =< N_1^3, N_2^3 >=< 2, 1 >$ and $\overrightarrow{M}_3 =< M_1^3, M_2^3 >=< 1, 1 >$, it can be easily noticed that in the example of Figure 6.5, and based on the frequency value, a single motif, S_1^3, has to be selected as a significant motif having 2 occurrences ($N_1^3 = 2$) in the originally observed graph, compared to a single occurrence ($M_1^3 = 1$), in the 2 random graphs G_1 and G_2. This fact does not apply, however, to motif S_2 which is as equally observed in the original graph as in random graphs ($N_2^3 = M_2^3 = 1$). Subgraph S_2 can now be named M_1, standing for the first significant motif associated with graph G_{obs}. For the example illustrated in Figure 6.5, the main outcome of the motif selection phase is the set of significant motifs, $\mathcal{M} = \{M_1\}$, for $N_m = 1\}$ and $M_1 \in M'_3$. It's based on these most significant motifs that vertices inside a graph can be represented.

6.3 Vertex Representation using Motifs

The addressed problem, that is, the identification of the protocols being used by a host, is obviously centered on the activity of each host in a network (a vertex in the graph), as opposed to the analysis of the overall activity in the network (the whole graph) or each of the individual communications or flows (the edges). On the other hand, the graph associated to the observed networking activity is supposedly generated from the repetition of the motifs. Thus, we are interested in evaluating the involvement of each node in each

[5]This applies for the example in Figure 6.5. Beside frequency, other parameters, p-value and the z-score, are regarded for the rest of this chapter.

of the obtained motifs, that is, in each of the supposed elementary kinds of interactions.

Therefore, the next problem to address, once the motifs have been identified, is how to effectively represent the participation of a vertex in one or more of those motifs instances. Thus, we shift our focus from a graph-centered description to a vertex-centered one. For this purpose, each vertex, V_i, in the observed graph G_{obs}, is associated to a vector, P_i, whose components, p_j, describes the involvement of that vertex in a specific motif, $M_j \in \mathcal{M}$. The vector, P_i, is thus a parametrization of the vertex and, consequently, of its activities, being referred to as the *vertex profile*.

In an initial approach, the components of the vertex profile may take a binary value indicating whether it is involved ($p_j = 1$) or not ($p_j = 0$) in a motif:

$$P_i =< p_j : 1 \leq j \leq N_m > \tag{6.18a}$$

$$p_j = \begin{cases} 1, & \text{if } V_i \in V(S(M_j, G_{obs})) \\ 0, & \text{Otherwise} \end{cases} \tag{6.18b}$$

where $S(M_j, G_{obs})$ is the ensemble of all the repetitions of motif M_j in G_{obs} and $V()$ is the set of vertices in the given set.

To illustrate this process, let's represent the vertices in the input graph G_{obs} shown in Figure 6.5 using the significant motifs associated with graph G_{obs}.

As shown previously, a single motif S_2 (or M_1) has been associated with graph G_{obs}. In this case, $\mathcal{M} = \{M_1\}$ and the vertex profile is a single-dimensional vector, $P_i =< p_1 >$, based on a single attribute, p_1, associated to motif M_1.

To evaluate attribute p_1 for each vertex in the graph, it is sufficient to answer whether or not the vertex is involved in the motif substructure as detected in the original input graph, G_{obs}. An insight[6] into the input graph in Figure 6.5 shows that nodes $(1, 2, 3)$ and $(6, 7, 8)$ are contributing into 2 different instances of motif S_2^3 (or M_1), highlighted in dashed-green line in the input graph Figure 6.5. As a result, $p_1 = 1$ for all of these nodes and $p_1 = 0$ for remaining nodes in the graph. Table 6.1 summarizes the vertex profiles of all nodes in the input graph.

Based on this representation, a vertex may be part of different motifs at the same time. As such, for cases where $N_m > 1$ (which are most common) this process has to be repeated for each motif in the set $\mathcal{M}$. Moreover, there are cases when the need is to analyze the same vertex through multiple input graphs at the same time. Here, the result will be a profile vector, P_i, whose dimension equals to the cardinality of the union set of all significant motifs

[6]The motif extraction tool (e.g. Fanmod-tool [Wernicke, 2006b]) usually generates for each vertex, a list of subgraphs in which the vertex is contributing in within the input graph.

Table 6.1: Example of vertex profiling using a single motif attribute

Node	**Contributes in motif S_2^3 from G_{obs}**	**Profile vector $< P_i >$**
1	yes	$P_1 =< 1 >$
2	yes	$P_2 =< 1 >$
3	yes	$P_3 =< 1 >$
4	no	$P_4 =< 0 >$
5	no	$P_5 =< 0 >$
6	yes	$P_6 =< 1 >$
7	yes	$P_7 =< 1 >$
8	yes	$P_8 =< 1 >$

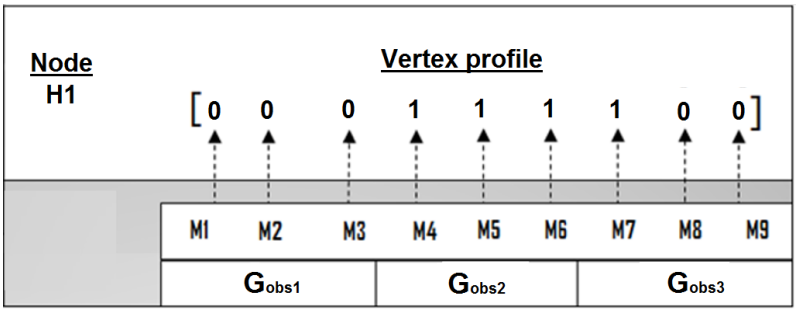


Figure 6.7: An example of a vertex profile using multiple motifs/graphs

selected from each input graph. Figure 6.7 illustrates an example of a 9-dimensional vertex profile for a node that has been analyzed simultaneously through three different input graphs, $G_{obs1}, G_{obs2}, G_{obs3}$, out of which a sum of 9 significant motifs were extracted and selected: $\{M1, M2, M3\}$ from G_{obs1}, $\{M4, M5, M6\}$ from G_{obs2} and $\{M7, M8, M9\}$ from G_{obs3}.

The usage of such cases and the interpretation of the vertex profile representation across multiple input graphs will be explained later in this chapter. Up to this point, graph and vertex representation based on motifs is shown in a general context, that is, associating vertices with profile vectors representing their contributions in motifs within the mined input graph.

In the following sections, the vertex representation is applied in traffic classification contexts in computer networks. The motif detection process and the graph vertex profiles, shown so far, are corner stones in building the traffic identification solutions that will be presented next.

6.4 Motif-based Traffic Classification: the Reference System

Once the basics of motif analysis of graphs has been presented, its application to traffic classification, as originally proposed in [Allan2009] is described next. As will be detailed in the next sections, two main building blocks are needed. The first one is related to the obtention of a suitable profile for the node or nodes being analyzed, while the second block is devoted to the classification of those profiles according to the classification method of choice. Thus, a generic motif-based traffic classification model is presented first independently from the motif-based classification method in use. In light of this model, a detailed implementation of the native classification method in [Allan, 2009] will follow[7]. To the best of our knowledge, the work in [Allan, 2009] (detailed in [Allan, 2008]) is one of the fewest attempts on using motifs for traffic classification purposes in computer networks, despite the reported accuracy in this preliminary setup.

Applying motifs to traffic classification in computer networks relies on the same assumption reported in [Milo, 2002]: the appearance of network motifs at high frequencies suggests that they may have some specific functions in the information processing, and thus in the application interactions performed inside computer networks. Motif based traffic classification consists thus on identifying the application(s) associated with each computer host based on its profile, which should represent each of the host's contributions in the set of motifs mined from a traffic exchange graph.

Thereby, given that network traffic and host exchanges can be represented with colored and directed graphs, as will be detailed next, the same methodology used for motif detection and vertex representation described in the previous section can be applied in traffic identification.

Therefore, a motif-based traffic classification system should be composed of three basic blocks, as shown in Figure 6.8, namely:

- Preprocessing: This module takes the observed traffic activity and obtains the associated networking graph.
- Motif-based parametrization: The aim of this block is to obtain the representation for each host (i.e the vertex profile) as a function of the mined motifs, according to the procedure in Section 6.3. The main outcome of this phase is the set of motif-based host vertex profiles. The association of motifs to applications is established during the training of the system, being transparent to the parametrization process. Thus, given an input graph from an observed network activity and a set of motifs,

[7]The system proposed in [Allan2009] will be reproduced as literally as possible, although some adaptations are needed. They will be mentioned when describing the system modules.

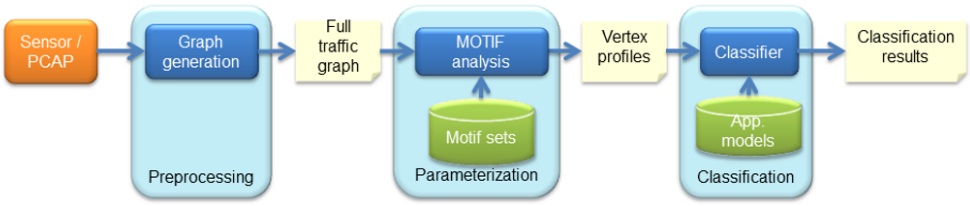


Figure 6.8: Traffic classification system based on motifs

this block will obtain the profiles for that graph by searching for motifs appearances.

- Classification: This block labels each host by the application in use[8] by analyzing its motif-based vertex profile. The classification method can be based on any statistical, pattern matching or ML algorithm.

The details of methods applying the aforementioned motif-based classification may vary in many aspects, as, for example, the classification method or the way the used models are estimated. In fact, the objective of this work is to propose and evaluate some alternatives for a motif based classifier. In order to compare the proposed methods, and as stated before, a reference system is set by referring to the original work in [Allan, 2009] and [Allan, 2008]. For this, it is necessary to obtain the motifs and train the model for each application (Figure 6.9) for its subsequent use in classifying traffic.

Although the system should operate in training and classification modes with some different functionalities, we will explain the system as a whole in the next sections, including the functions required for training in the discussions.

In the remaining of this section, each elementary operation in Figure 6.9 will be detailed and each of the choices taken by authors in the original work [Allan, 2008] at the input, techniques and output levels will be described.

6.4.1 Preprocessing: Network Graphs Construction

The first stage for motif-based traffic classification consists of handling the raw captured data in order to obtain the graph representation of the observed networking activity (preprocessing block in Figure 6.9). Furthermore, during the training of the system, some additional steps are required to label each of the flows according to the associated application and to obtain graphs including just a single application. Thus, the required functionalities are:

- *Ground truth generation* (only in training mode): As previously mentioned, the existence of a set of labeled data is mandatory for the training of the system. As it is necessary to handle a huge volume of data,

[8] In the initial approach [Allan, 2009], each host is supposed to contribute in a single application.

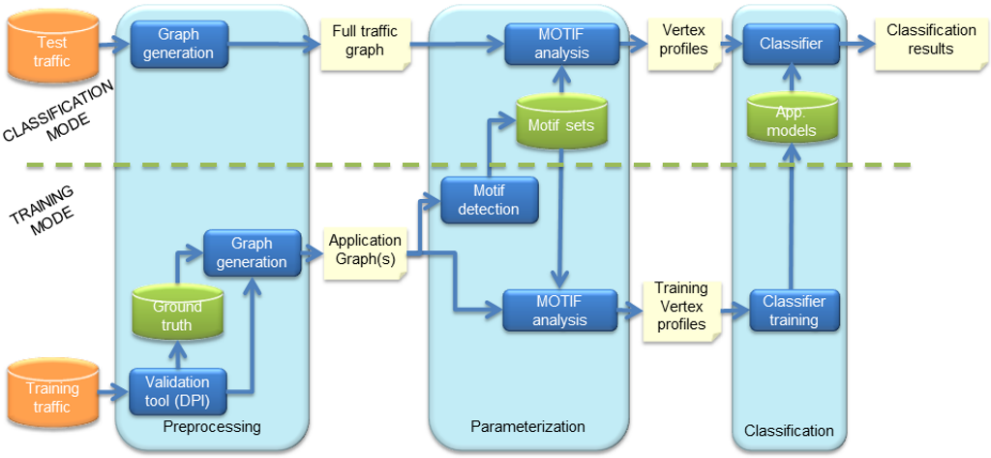


Figure 6.9: Modules for motif-based traffic classification including training stages

the labeling has to be done automatically. In the original work by Allan [Allan, 2008], the authors used a port-based classification, which is not accurate enough nowadays. Therefore, to build the reference system, in this work a DPI based approach ("Validation tool" module in Figure 6.9) is used. The details about the labeling procedures and datasets are shown in Chapter 3.

- *Graph generation*: Graph representation is the core of the preprocessing step. Obviously, it takes a capture of traffic, live or from a PCAP file, and generates the graph for the observed networking activity.

  Networking activity is represented by a colored directed graph, G, in which each of the nodes, H, is associated to a host, that is, a vertex exists for each different IP observed in the traffic. Arcs between nodes represent a communication between the associated hosts. In this case, an arc is created if at least one flow is observed between both hosts. And the direction of the arc is from the flow initiator to the flow destination. Finally, each node (host) is associated a color depending on its function [Allan, 2009]: client, server or peer. Figure 6.10 shows a simple example of a network graph.

  Thus, given a set of input traffic, the associated networking graph, $G(H, A)$, is obtained as follows. Let I be the set of different IPs in the raw traffic set

$$I = \{IP_1, IP_2, \ldots, IP_M\} \tag{6.19}$$

  and F the set of observed flows

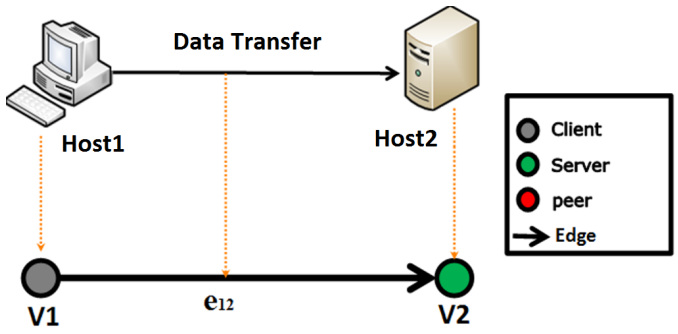


Figure 6.10: Graphical representation of exchanges between two network hosts

$$F = \{f(IP_i \rightarrow IP_j)/IP_i, IP_j \in I\} \tag{6.20}$$

where $f(IP_i \rightarrow IP_j)$ denotes a flow from IP_i to IP_j.

The set of vertex, H, is obtained by associating a node, H_i, to each IP in I

$$H = \{h_i/h_i \leftrightarrow IP_i, \forall IP_i \in I\} \tag{6.21}$$

The set of arcs, A, is obtained from the set of flows by defining an arc between nodes h_i and h_j if there exists at least one flow in F between IP_i and IP_j

$$A = \{a(i,j)/\exists f(IP_i \rightarrow IP_j) \in F\} \tag{6.22}$$

The direction of the arc is determined by the flow initiator, for which additional information from the flow is required. In particular, the initiator is considered to be the source for the first observed packet in the flow[9].

Finally, a color is assigned to each vertex (host) according to its functionality. In [Allan, 2009] this is done, for a given flow[10], according to:

$$\text{color}(h_i) = \begin{cases} 1 \text{ (client)}, & \text{if } d_p \in \phi \\ 2 \text{ (server)}, & \text{if } s_p \in \phi \\ 3 \text{ (peer)}, & \text{if } d_p, s_p \notin \phi \end{cases} \tag{6.23}$$

where d_p and s_p stand, respectively, for the destination and source ports of the flow and ϕ is the set of IANA assigned ports for services.

[9] As defined, some conflicts can appear if a host participates in more than one flow.

[10] Again, as defined, some conflicts may appear if more than one application is served by a single host.

Two types of graphs are extracted: full traffic and application graphs. The full traffic graph illustrates host communications for all applications. Therefore, it cannot be associated in any way with any individual application and, usually, contains a huge number of flows, being a complex graph. On the other hand, application graphs are extracted by filtering communications on a per application basis. Therefore, in these graphs all the edges are associated to the same application[11].

Application graphs are relevant for training the system, as the motifs should be associated to an application. Therefore, the graph generation module in the preprocessing phase when in training mode generates only application graphs (Figure 6.9). Obviously, to generate them the flows should be filtered according to their application, for which the ground truth is used in this work. It is worth mentioning that in the original work [Allan, 2009] port-based filtering was used for this purpose, which is coherent with the way in which the ground truth was obtained.

These application graphs are to be mined and associated with motifs (detailed next in Section 6.4.2). To obtain statistically significant motifs, more than one graph should be extracted for each application at the preprocessing phase. The highest the number of application graphs for a given application the more likely will be the possibility to detect enough motifs for that application. The upper limit of the number of application graphs is limited by the computational cost associated with motif detection, as will be discussed next in Section 6.4.2. For this purpose, some time slots in the analyzed traffic can be considered, each of them generating an application graph.

Another relevant concern is that different application graphs should have preferably similar number of participating nodes to prevent any bias in the obtained results. In this regard, many approaches can be considered. For example, collecting network captures for a fixed amount of time for all protocols is not efficient because certain applications can be much more heavily used than others. Another alternative (used in the native approach [Allan, 2008]) is to analyze application graphs that have a similar number of participating nodes by allowing the network capture lengths to vary. In [Allan, 2008], 10 graphs were searched for each application protocol, and only the first 40 to 80 hosts are considered for each graph.

The output of this phase are the full traffic graph and the set of all filtered application graphs which will be analyzed for various purposes including motif detection, training set generation, and motif-based parametrization, as detailed next in the following sections. From the implementation point of view, the output should be formatted according to the tool used to detect motifs, that is, Fanmod-tool [Wernicke, 2006b].

[11] An illustrative example of application graphs is detailed in Section 6.4.2, Figure 6.11.

6.4.2 Motif-based Parametrization

The next step in motif-based traffic identification is the host parametrization (Figure 6.9). The target of this phase is to build a profile for each host in the graph, that is, the vertex profile for each host (Section 6.3) which, as previously explained, should reflect the involvement of that host in each of the selected significant motifs ("Motif sets" in Figure 6.9).

Motif Detection

Obviously, the parametrization depends on the motif set, which should be obtained during the training of the system. Therefore, when in training mode, the parametrization phase includes the analysis of the graphs to search and select those significant motifs (module "Motif detection" in Figure 6.9). For this, the application graphs, as provided by the preprocessing step, are used as the inputs for the motif detection analysis, which is carried out according to the procedures in Section 6.2. Nevertheless, this analysis is made on a per application basis, that is, only the application graphs for a selected application are considered for the motif extraction each time, and the analysis is repeated for all the applications. This way, the obtained motifs are labeled according to the application for which they were obtained.

From the implementation point of view, the motif detection process (Figure 6.9) is made by mining application graphs one by one using the Fanmod-tool [Wernicke, 2006b], similarly as was done in the original work by Allan. This process is depicted in Figure 6.11, which shows a simplified motif detection process for three different application graphs. Due to space limitations, graph randomization, used in the motif detection procedure, is not shown in Figure 6.11. In this example, if the relevance criterion is a frequency of appearance greater or equal to 2, only a single size 3 motif is selected for each application (Figure 6.11).

In a real case scenario, an enormous number of motifs (e.g. up to 130 motifs in [Allan, 2008]) can be obtained, which increases the dimensionality of the profiles and, thus, the complexity of the classification problem. Therefore, it is necessary to set the required statistical level of significance for a motif to be selected as significant during the training of the system, being this a key parameter in this phase. As explained in Section 6.2, some parameters can be used for this purpose. In [Allan, 2008], the statistical information generated by the Fanmod-tool is used, and only motifs with a $frequency > 1\%$ and a $p - value = 0$ are considered relevant.

At the output of the motif detection module, a set of motifs, $\mathcal{M}$, each of them associated to an application, are obtained.

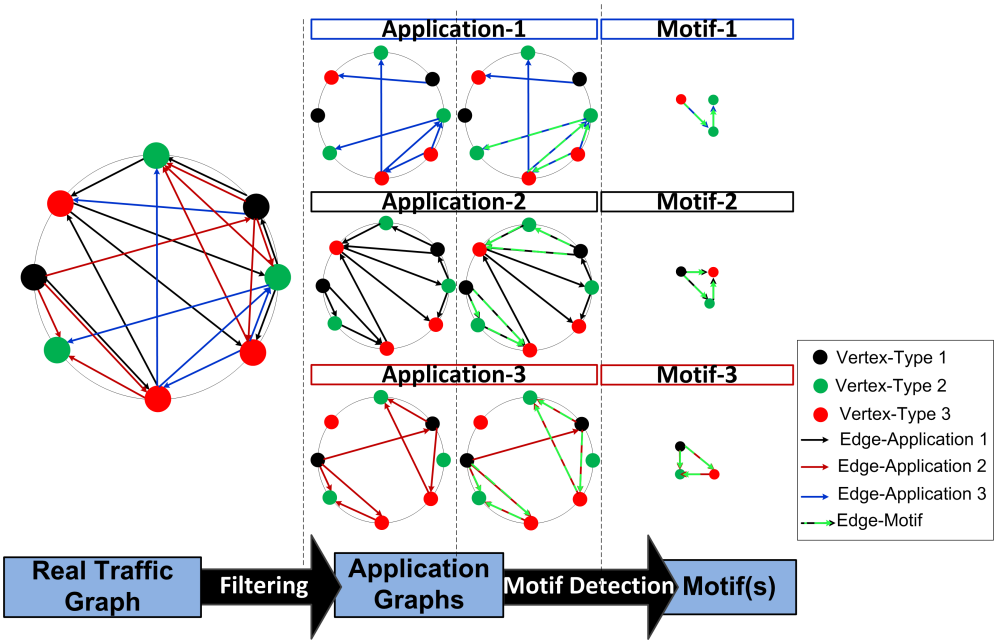


Figure 6.11: Motif extraction from network traffic application graphs

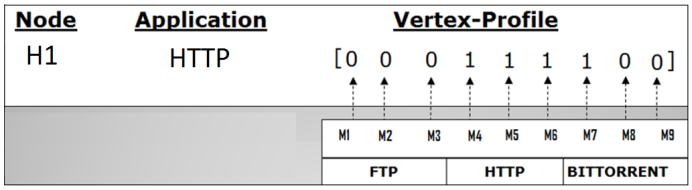


Figure 6.12: Building the vertex profile for each host

Motif Analysis

Given the motif set and an input networking graph, the procedure to obtain the vertex profiles is that described in Section 6.3. The main novelty in this case is the association of the motifs to the applications as a result of the motif detection procedure. Thus, the outcome of this phase is the set of vertex profiles for each of the hosts in the graph, which is composed by binary values in the original work by Allan [Allan, 2009] and thus in this phase of the work.

Figure 6.12 shows an example of a vertex profile for a host generating HTTP traffic built from the set of considered motifs. As shown, each of the motifs can be represented by its extended adjacency matrix and labeled by the application that generated them. Thus, groups of positions in the vertex profile can be associated to an application. In this case, positions 4 to 6 correspond to HTTP and have a value of 1, which means that the considered host, H1, is involved in the appropriate role (color) in the appearance of those motifs in the input

graph. In a naïve approach, as HTTP related motifs contribute in the profile, the traffic from host H1 could be labeled as HTTP. But, as shown in the graph, one of the BitTorrent related motifs is also contributing to the profile. Thus, an additional procedure is required to classify the profiles, which constitutes the next block of the system.

Similarly to the motif detection, the motif analysis is made using the Fanmod-tool, which generates the lists of vertices (hosts) involved in each of the considered motifs. Furthermore, as a training procedure is required for the classifier, it is necessary to obtain the vertex profiles for the training traffic, that is, for the nodes in the application graphs obtained at the output of the preprocessing phase, when in training mode (Figure 6.9).

6.4.3 Classification

The classification block (Figure 6.9) is the final phase and aims to classify the traffic for unknown hosts, given their vertex profiles as defined previously. Although many options for the classification method could be considered here, in this subsection, the classification method in [Allan, 2009] is detailed at the three different levels, as this will be used as the reference system.

1) Classification target: At the output level, the system in [Allan, 2009] is based on a multi-class, single-labeled host classification.

   In particular, the set of considered applications is, $L = \{$AIM, DNS, HTTP, Microsoft Active Directory Domain Services (MSDS), NetBIOS, SSH, Kazaa$\}$.

2) Classification input: Based on the vertex profile representation described previously, the classification input is the set of profile vectors, v_x, associated with each test vertex:

$$\overrightarrow{v_x} = [a_1, a_2, a_3, \ldots, a_n] \tag{6.24}$$

   where n is the number of significant motifs. For an unclassified host, H_x, with profile $\overrightarrow{v_x}$, the classification consists in obtaining the application label, L_x, based on the obtained model from the training profiles.

3) Classification technique: The ML technique used in [Allan, 2008] is KNN [Huang, 2009], which is a well-known method for classifying objects based on the closest samples in the model, that is composed by the whole or a subset of the training samples. Thus, as host vertex profiles are vectors in the feature space, the classification consists on assigning the class of the K-nearests profiles in the model to the input profile (Figure 6.13).

   To determine the proximity of two objects in the feature space, many distance types (Euclidean, Cityblock, Manhattan, Chebyshev, etc.) may

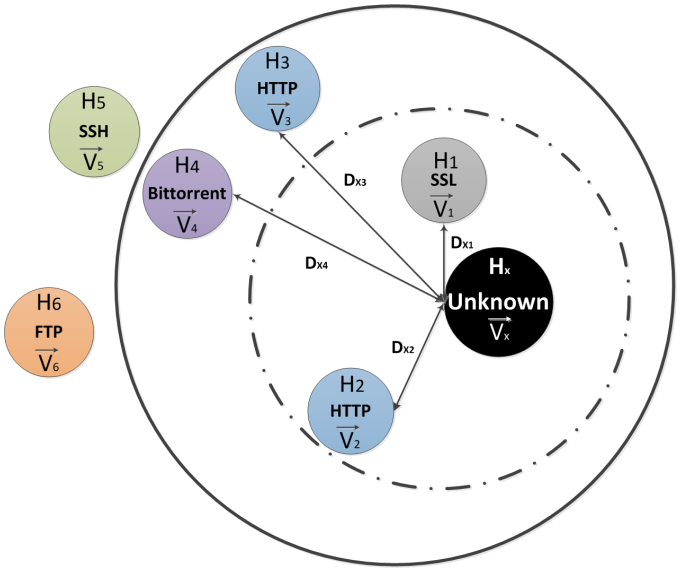


Figure 6.13: KNN classification example

be used. The Euclidean distance (used in [Allan, 2008]) between two n-dimensional points $\overrightarrow{V_a} = [a_1, a_2, \ldots, a_n]$ and $\overrightarrow{V_b} = [b_1, b_2, \ldots, b_n]$ is calculated as:

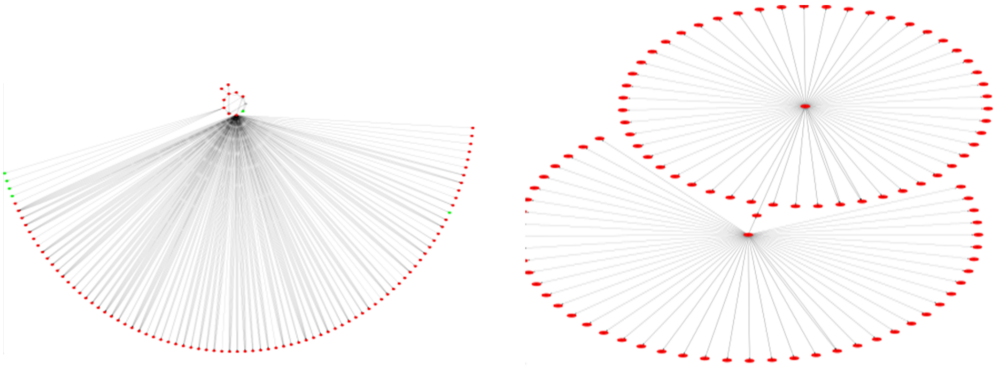
$$d(\overrightarrow{V_a}, \overrightarrow{V_b}) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2} \tag{6.25}$$

In the classification phase, K is a user-defined constant, and a profile is classified by assigning the label of the nearest neighbor (for $K = 1$), or through a majority voting process among the K nearest neighboring samples (for $K > 1$)[12]. Hence, K is a critical parameter affecting the classification decision. As an example, Figure 6.13 shows some labeled samples, H_1 to H_4 and an unknown one, H_x. For $K = 1$, H_x will be classified as SSL, as per the closest host label, H_1. However, it will be classified as HTTP for $K = 4$, as two of the four closest samples are labeled HTTP.

Therefore, after identifying the K nearest neighbors in the model set for an unlabeled host profile, H_x, with vertex profile $\overrightarrow{V_x}$, H_x label can be inferred exclusively from the closest neighbor (if K=1, as in [Allan, 2008]), or through a majority voting process if $K > 1$). It is worth mentioning that, in this case, as in [Allan, 2008], the model is composed by all the training vertex profiles obtained from the parametrization.

Up to this point, the native motif-based classification system proposed in [Allan, 2008] has been described. In the next section this basic approach will

[12]KNN can be used in voting or regression modes, as shown in Section 6.7.3.



a) b)

Figure 6.14: Application graph visualization using Graphviz for: a) BitTorrent and b) Gnutella

be experimentally evaluated and assessed in order to later propose some enhancements and be able to measure their impact over this reference system.

6.5 Reference System Results

Classification results are shown in this section for the native method in [Allan, 2008]. In order to test the classification methods proposed in this chapter, two datasets were used, namely: CS-A4 for training, CS-B1 for classification and testing. The contents and details for these datasets are shown in Chapter 3.

The analysis of the datasets after preprocessing and motif detection (Figure 6.9) evidenced that only 14 protocols met the required properties as to be considered for training and testing the classification methods. These protocols are: BitTorrent, FTP, HTTP, IRC, Mail_IMAP, Mail_POP, Mail_SMTP, MSN, NTP, Oscar, SIP, SNMP, SSH, SSL. Other applications appearing in the datasets were discarded due to an insufficient number of samples, not providing sufficiently representative associated motifs or not appearing in both the training and testing datasets.

In the preprocessing phase, up to 112 different application graphs for the 14 protocols were obtained and analyzed in search of motifs from the training dataset, that is, CS-A4. An inspection of the obtained application graphs can optionally be done using special tools like Graphviz [Gansner, 2006] visualization software as shown in Figure 6.14.

Figure 6.14 visualizes network exchanges among hosts specific to a single application. Namely, Figure 6.14a) depicts a typical BitTorrent application graph where peers communicate with a server (the tracker), and issues multiple connections to other peers. Figure 6.14b) depicts two Gnutella central nodes connected to several peer nodes.

Table 6.2: Number of selected hosts per application for training in the CS-A4 dataset

Application	**# Host training samples**
BitTorrent	43
SSL	73
HTTP	61
Mail_IMAP	12
NTP	23
Oscar	18
MSN	25
Mail_POP	26
FTP	30
SNMP	31
SIP	26
SSH	32
Mail_SMTP	21
IRC	27

However, this kind of visual intuition is only suggestive and derived from our prior knowledge about the applications and the network environment where the data is captured. Therefore, application graphs intuitions are not and should not be part of the classifier design. These graphs can be only used for assessing motif detection.

Based on these application graphs, significant motifs of sizes 3 and 4 were selected out and used as the elements to build the vertex profile.

After the motif set is chosen, all the application graphs from CS-A4 as well as the full graph from CS-B1 are used as the inputs for the MOTIF analysis block (Figure 6.9), thus providing the training and testing sets of vertex profiles.

6.5.1 Experimental Results

In this section, results obtained for each module of Figure 6.9 are shown.

6.5.1.1 Preliminary Results

Before showing the results for the motif detection process, we start first by exploring some preliminary results regarding the datasets used for training from a host classification point of view.

The dataset used for training, CS-A4, contains hosts that are engaged in a single application activities which are the ideal candidates for the motif analysis according to the original method in [Allan, 2008]. By parsing the flow-based

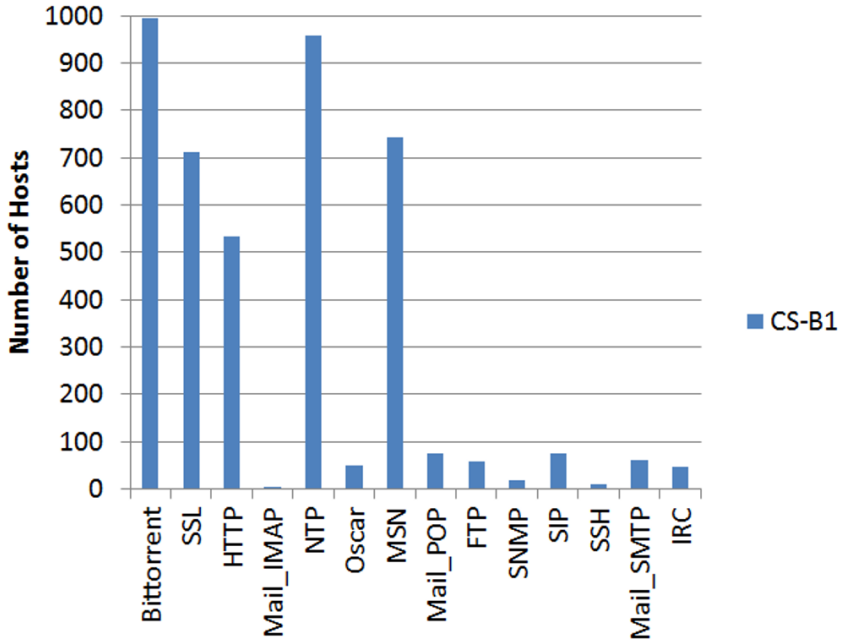


Figure 6.15: Ground truth results for hosts in the test dataset (CS-B1)

ground truth results, single-labeled hosts were selected as training samples. For this, a simple heuristic is used: a host is single-labeled with application l if all flows generated by that hosts are labeled by DPI as belonging to application l.

As show in Table 6.2, the number of selected samples per protocol were chosen below 100 as to keep the training set balanced to the far most extent. Using dataset CS-A4, 448 single-labeled hosts were used as training samples.

The distribution of hosts in the test dataset (CS-B1) per protocol are shown in Figure 6.15. As stated earlier, as not all the hosts contribute in motifs, and as not all applications can be associated with motifs, a subset of hosts can be used for classification from each dataset. Using dataset CS-B1, 4333 test hosts were used for classification.

6.5.1.2 Motif Enumeration and Selection Results

As stated earlier, to detect motifs, the Fanmod-tool is run on application graphs. The main settings used with the Fanmod-tool for motif detection are shown, altogether with the parameters for significant motif selection[13], in Table 6.3.

[13]The same Fanmod-tool settings and parameters are used for subsequent motif detection and selection experiments in Sections 6.6 and 6.7.3.

Table 6.3: Fanmod-tool Settings and parameters for significant motif selection

FANMOD Option	Possible Values	Default Value	Chosen value
subgraph (motif) size	>3	[3]	(3-4)
# of random graphs in the null-model	>1	[1000]	(5000)
# of samples	>1	[100000]	(100000)
Frequency of a subgraph	>1	-	(>10)
p-value	0.01	-	(0.01)
Z-score	>0	-	(>10)
Persistency	>1	-	(6)
Full enumeration?	1(yes)/0(no)	[1]	1 (yes)
Directed graphs?	1(yes)/0(no)	[1]	1 (yes)
Colored vertices?	1(yes)/0(no)	[0]	1 (yes)
Colored edges?	1(yes)/0(no)	[0]	0 (no)
Randomization type	0(no regard) 1(global const) 2(local const [14])	[2]	(2)
Regard vertex colors[15]?	1(yes)/0(no)	[0]	1 (yes)
Regard edge colors?	1(yes)/0(no)	[0]	0 (no)
Re-estimate subgraph number?	1(yes)/0(no)	[0]	0 (no)
# of exchanges per edge	>1	[3]	(5)
# of exchange attempts per edge	>1	[3]	(5)

Some of the default values did not fit to the the purposes of our experiments, and had thus to be changed. Most importantly, two values are critical for motif mining, namely, the *regard vertex color* condition, to maintain the colors of vertices, and the *number of random networks*.

With these conditions, and based on the application graphs generated from our datasets, 156 significant motifs of sizes 3 and 4 were selected, out of which 36 motifs are considered significant and used as the elements to build the vertex profile.

Therefore, in our implementation of the reference method, the dimensionality of the profiles is 36. The obtained motifs and their associated applications are shown in Table 6.4.

Table 6.4 shows the number of occurrences for each of the 36 significant motifs across the 14 tested protocols, over the training dataset[16]. Moreover, this table shows that motifs may collide, that is, a motif can be found in more

[16]Due to the persistency condition, the same set of motifs extracted out of the training dataset are obtained when mining the test dataset.

Table 6.4: Number of occurrences for each of the 36 significant motifs, obtained with the native method, across the 14 tested protocol (Training dataset CS-A4)

MOTIF-ID	#Occurences	BitTorrent	FTP	HTTP	IRC	Mail_IMAP	Mail_POP	Mail_SMTP	MSN	NTP	Oscar	SIP	SNMP	SSH	SSL
1	13	0	0	7	0	1	0	4	0	1	0	0	0	0	0
2	17	0	0	9	0	0	0	0	0	1	0	0	0	4	3
3	123	0	0	46	0	0	6	6	12	4	3	5	6	7	28
4	36	0	0	18	0	0	3	0	0	0	0	1	4	0	10
5	25	0	0	16	0	0	0	3	0	0	0	0	0	3	3
6	11	0	0	4	0	0	0	4	0	0	0	0	0	0	3
7	15	0	0	11	0	0	0	0	0	0	0	0	0	0	4
8	43	0	0	25	0	3	3	3	0	0	0	0	3	0	6
9	16	0	0	7	0	0	6	0	0	0	0	0	0	0	3
10	17	0	0	9	0	0	0	4	0	0	0	0	4	0	0
11	132	6	0	42	0	3	3	6	27	3	3	3	6	0	30
12	27	0	0	12	0	0	3	3	0	0	3	0	3	0	3
13	11	0	0	3	0	0	0	4	0	0	4	0	0	0	0
14	15	0	0	9	0	0	3	0	0	0	0	0	3	0	0
15	42	27	0	0	3	0	0	0	6	6	0	0	0	0	0
16	65	0	3	0	4	0	0	0	29	27	0	0	0	0	0
17	15	0	1	0	0	0	0	0	14	0	0	0	0	0	0
18	21	0	1	0	0	17	0	0	3	0	0	0	0	0	0
19	7	3	3	0	0	1	0	0	0	0	0	0	0	0	0
20	17	0	3	0	4	0	0	0	10	0	0	0	0	0	0
21	6	0	0	0	0	0	0	0	1	0	5	0	0	0	0
22	3	0	0	1	2	0	0	0	0	0	0	0	0	0	0
23	69	0	3	0	8	0	0	0	27	29	0	0	0	0	0
24	71	0	3	0	10	0	0	0	31	26	0	0	0	0	0
25	11	0	0	0	0	0	4	0	0	6	0	6	0	0	0
26	10	0	0	6	0	1	0	0	0	0	0	0	3	0	0
27	13	0	0	0	0	0	0	0	6	7	0	0	0	0	0
25	13	0	0	0	0	0	0	0	0	12	1	0	0	0	0
29	15	0	0	0	0	6	0	6	0	0	0	0	3	0	0
24	13	9	0	0	0	0	0	0	0	4	0	0	0	0	0
31	45	33	0	0	0	0	0	0	0	12	0	0	0	0	0
32	49	31	0	0	0	0	0	0	0	18	0	0	0	0	0
33	56	36	0	0	0	0	0	0	0	12	0	0	8	0	0
34	71	40	0	0	0	0	0	0	0	31	0	0	0	0	0
35	48	29	0	0	0	0	0	0	0	18	0	0	0	0	0
36	63	30	0	0	0	0	0	0	0	33	0	0	0	0	0

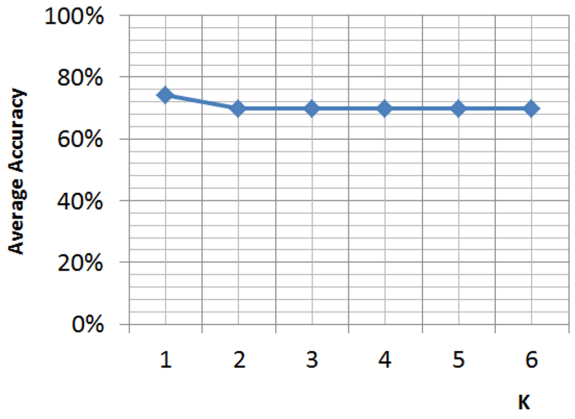


Figure 6.16: Accuracy average for different values of K

than one application's graphs. As stated earlier, this proves that the appearance of any single motif is not necessarily an indication of a given application being in use by a host but, rather, a set of multiple motifs should be regarded per application.

Table 6.5 shows 5 examples of single-labeled host vertex profiles (see Section 6.3) from the training dataset (CS-A4): H_1 (BitTorrent host), H_2 (FTP host), H_3 (HTTP host), H_4 (IRC host) and H_5 (Mail_IMAP host).

6.5.1.3 Classification Results

A KNN classifier, as explained in Section 6.4.3, is used to label each of the profiles in the evaluation set. Therefore, all the training samples are directly considered as the applications model (Figure 6.9) and used in the classification procedure without any kind of preprocessing, selection or adjustment of the model, that is, no training of the model exists in this case.

The implementation of the native motif-based classification in [Allan, 2008] is "single nature traffic", that is, based on single label test hosts, though not stated explicitly. Despite the fact that hosts might be, and most probably are, contributing in more than one application, their vertex representations, as implemented in [Allan, 2008], does not consider host activities in applications, but rather, for a single application. In fact, each host (or IP address) is assigned a different identifier in each application graph.

To evaluate the accuracy per application protocol, the value of K should be chosen first. Figure 6.16 shows the accuracy values averaged on all protocols for different values of K.

Apparently, increasing the value of K above 2 or 3 does not increase the

Table 6.5: Examples of host vertex profiles (columns H_i) from the training dataset (CS-A4)

Motif-Id	**Profile Vector Attributes** p_j	H_1	H_2	H_3	H_4	H_5
1	p_1	0	0	0	0	0
2	p_2	0	0	1	0	0
3	p_3	0	0	1	0	0
4	p_4	0	0	1	0	0
5	p_5	0	0	0	1	0
6	p_6	0	0	1	0	0
7	p_7	0	0	1	0	0
8	p_8	0	0	1	0	0
9	p_9	0	0	0	0	0
10	p_{10}	0	0	1	0	0
11	p_{11}	0	0	1	0	0
12	p_{12}	0	0	1	0	0
13	p_{13}	0	0	0	0	0
14	p_{14}	0	0	1	0	0
15	p_{15}	1	0	0	1	1
16	p_{16}	0	1	0	1	1
17	p_{17}	0	1	0	0	0
18	p_{18}	0	1	0	0	1
19	p_{19}	1	1	0	0	1
20	p_{20}	0	1	0	1	0
21	p_{21}	0	0	0	0	0
22	p_{22}	0	0	0	1	0
23	p_{23}	0	1	0	1	0
24	p_{24}	0	1	0	1	0
25	p_{25}	0	0	0	0	0
26	p_{26}	0	0	0	0	1
27	p_{27}	0	0	0	0	0
28	p_{28}	0	0	0	0	0
29	p_{29}	0	0	0	0	1
30	p_{30}	0	0	0	0	0
31	p_{31}	1	0	0	0	0
32	p_{32}	1	0	0	0	0
33	p_{33}	1	0	0	0	0
34	p_{34}	1	0	0	0	0
35	p_{35}	0	0	0	0	0
36	p_{36}	1	0	0	0	0

overall accuracy (69.81%), but eventually, adds additional computational cost. In the following, we show the classification results for $K = 3$.

To start with the classification results, the confusion matrix in Table 6.6 is illustrative from many perspectives. First, to some extent, the native motif based classification can be considered as accurate for most protocols in the dataset, given that the diagonal values are the highest in each row.

Table 6.6: Confusion matrix for reference motif-based classification system (K=3, dataset CS-B1)

Classified as → **Input class** ↓	a	b	c	d	e	f	g	h	i	j	k	l	m	n
a=BitTorrent	507	0	0	0	175	0	0	0	223	0	0	0	0	89
b=SSL	1	349	230	0	0	0	0	0	0	115	1	0	0	16
c=HTTP	0	0	218	0	0	0	57	124	0	0	0	0	0	133
d=Mail_IMAP	0	0	1	2	0	0	0	1	0	0	0	0	0	0
e=NTP	59	0	0	0	852	0	23	0	0	0	2	0	0	21
f=Oscar	0	2	5	0	0	39	0	3	0	0	0	0	0	0
g=MSN	7	3	12	0	50	0	668	0	0	0	0	0	0	2
h=Mail_POP	0	4	1	0	2	0	0	68	0	0	0	0	0	0
i=FTP	0	0	0	0	3	0	15	0	38	0	0	0	0	3
j=SNMP	0	2	5	0	0	0	0	0	0	12	0	0	0	0
k=SIP	3	1	5	0	0	0	0	1	0	0	64	0	0	0
l=SSH	0	0	4	0	0	0	0	0	0	0	0	6	0	0
m=Mail_SMTP	0	0	11	0	0	0	1	0	0	0	0	0	49	0
n=IRC	0	1	0	0	1	0	6	0	1	0	0	0	0	36

On the other hand, the confusion matrix shows the effect of colliding motifs where application protocols are confused by the motif-based classifier, which increases the FP rate and thereby, decreases the accuracy for some applications. Next, the experimental results obtained for different values of K are shown and discussed.

Figure 6.17 shows the classification accuracy for application protocols in CS-B1 for $K = 3$. As shown in this figure, host classification accuracy ranges from 41.22% to 91.25%. Averaged for all protocols in the dataset, 69.81% of accuracy can be obtained with the native method.

Up to this level, the native method in [Allan, 2008] has been reproduced to the far most extent, except for using the same protocols or PCAP datasets which are beyond our reach as being unpublished by the original authors.

Further analysis and discussions related to these results are presented in the next section.

6.5.2 Analysis and Discussions

The previous experimental results show promising properties and performance for the method proposed in [Allan, 2008], although they should be clearly

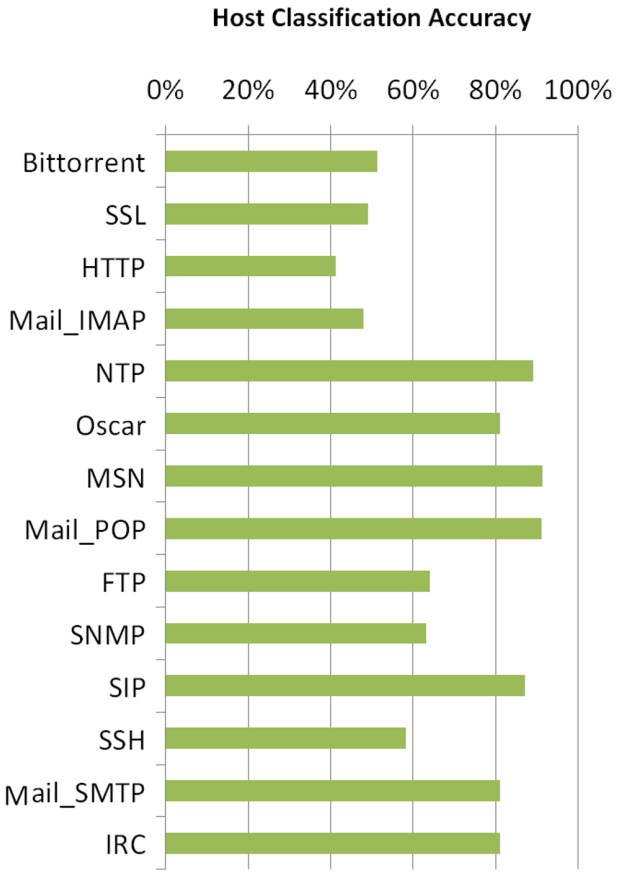


Figure 6.17: Accuracy results for the reference system, for the CS-B1 dataset trained on CS-A4 using K=3

improved. A first preliminary analysis of the results evidences dissimilar results for different protocols and a low number of significant motifs given the number of protocols (34 vs. 14). While in the best case a single motif would be enough to differentiate an application, it is not expected to find this behavior but the opposite, as many applications are expected to share some schemes of communications, i.e., motifs. Thus, it is advisable to analyze the number and nature of the selected motifs.

Furthermore, there can also exist some collisions in the motifs associated to each protocol, thus generating some problems. In summary, despite the promising results, it is necessary to address some limitations, analyze some choices and improve its performance. This can be done from a deeper analysis of the system at the different levels: preprocessing, parametrization and classification (Figure 6.9). Next, we review the design decisions and limitations at

each of those blocks in order to later propose some improvements that will be explored in this work.

- Preprocessing: The preprocessing stage is targeted at obtaining the set of graphs (full or application ones) for their further processing. The major questions at this phase are:
  - Ground truth: For the training of the system, the ground truth is built in the original work using port-based matching. This step is critical for the generation of the application graphs to be used to train the system and for the proper assessment of the results. As previously mentioned, this is not an acceptable method nowadays. Nevertheless, the previous results have been obtained by using a DPI tool for labeling the ground truth, which solves the related problem up to a reasonable limit.
  - Colors: To estimate each host type (client, server, or peer) and thus the color of the associated node, an exact match to the IANA set of assigned port numbers is used. Similarly to the previous problem, this is not a reliable solution nowadays.
  - Edges: An edge is created if a single packet is exchanged between two hosts (nodes). In fact, this condition takes no robust implications about whether the two hosts are really communicating or not, as most applications require at least 4 exchanged packets in order to build a session.
  - Single nature traffic: An underlying strong hypothesis is that each host only generates traffic from a single application. Clearly, this is not true in most common scenarios. In this regard, there is no distinction between different applications in the created edges even when using port-based labeling.
- Parametrization: Once the graphs are created, they are analyzed in search of motifs both for choosing those representative ones, when in training mode, and/or to obtain the profiles. The related major limitations are:
  - Number of motifs: Potentially, a big number of motifs can be obtained (up to 156 in the previous experiments for orders 3 and 4). The choice of the significant ones is relevant as the dimensionality of the profiles is the number of selected motifs. Thus, a compromise is needed, as they should include all the discriminant information

for the classification process, that is, they should be representative, while not introducing a big data problem due to a high dimensionality. This choice is not analyzed in [Allan, 2008].

  - Binary parametrization: The construction of the profiles is made on an on/off basis, as each of the parameters can take a value 0 or 1 depending on whether the node is involved in any occurrence of the associated motif. Obviously, this does not take into account the volume of related activity for the node, making impossible to even select the majority traffic. Furthermore, this criterion, together with the single nature traffic one, makes the discrimination of each of the activities from a node almost impossible.
  - Collision of motifs: The same motif can appear associated to more than one application during training. In the current approach, they will be considered as different, as they are obtained from different application graphs. This obviously introduces confusion for the classification process.

- Classification: Finally, a classification method is applied to the obtained parameter vectors. In this regard, the original work could be improved by addressing:

  - Labels: In concordance with the single nature traffic hypothesis, the original method in [Allan, 2008] follows a single-label classification model where a host is assumed to contribute in a single application. Obviously, as previously mentioned, this mode does not reflect real communication scenarios for most computer communications examined throughout public and locally captured traffic traces. In the real traffic, the host behavior is implicitly applicable into multi-applications context. In fact, a typical user behavior might be browsing the web, receiving an email while a system background task is fetching the latest updates through the Internet, which implies that the classification at the network host level should be multi-labeled.

In the next sections, some proposals targeted at improving the system by addressing some of the previous limitations are presented and assessed. In this regard, two main aspects are considered. First, an improved parametrization and profile creation method is proposed. The second aspect is to extend the modeling so as to consider a multi-label classification scenario in which each node should be labeled according to those most used applications.

6.6 Improved Motif-based Classification Method

Once the native system is implemented and its limitations highlighted, next we propose and evaluate some enhancements targeted at improving the performance of the method. In a first step, the changes will focus on the preprocessing and parametrization while keeping the condition of a single application per host.

The proposed enhancements regarding the preprocessing phase are:

- Edges: An edge is considered to occur between two vertices once one or more flows between the two corresponding hosts are observed, if the traffic is TCP, or once more than one data packet is exchanged, in case it is UDP. This way, signaling packets alone are not considered a communication.

- Colors: To estimate each host type or color an improved method is used. As previously shown (Section 6.4), in the native system the differentiation among these types is based on the exact match of IANA assigned port numbers. In the improved proposed method, port ranges are used instead to estimate the node types. This is motivated by the use of port obfuscation and this method is supposed to mitigate its effects when compared to exact port matching. Thus, estimating the node role relies on the port number ranges as defined by IANA: client port numbers are usually chosen in the range [1024-65535] while server port numbers are below 1024. The algorithm, shown in Algorithm 6.1, checks the source and destination port numbers together with the flow direction between the two nodes to distinguish the client role (outgoing edge) from the server role (incident edge). For other cases, nodes are assigned the peer roles, except for a few known protocols (not shown in Algorithm 6.1) above 1024 or protocols that use the same port source and destination numbers (e.g. Network Time Protocol (NTP) using UDP port 123).

Regarding the parametrization, some improvements are proposed for the motif detection and selection processes, as well as for the profile creation:

- Number of motifs: The choice of the significant motifs is the core for the parametrization. Thus, the major focus in the improvements is targeted at selecting those really relevant motifs by analyzing their statistical significance. The procedure is as follows. First, similarly to the native method, a limited number of application graphs per application are considered, each one involving a maximum number of nodes. Then, the potentially high number of motifs derived from the training set are analyzed to select those really significant ones by filtering out those that do not fit in the accepted values or ranges for the following magnitudes:

Algorithm 6.1 Algorithm for node type selection based on the transport layer port number ranges

▷

% Let $G = (V, E)$ be an application graph for application A, consisting of vertices V and edges E
% Let node $v \in V$ of unknown type (client, server, peer)
% For the observed flows between nodes $u \in V, v \in V$, let D_p and S_p be, respectively, the destination and source port numbers, at the transport layer.
% To get the type of node v
if (($D_p \geq 1024$) && ($S_p \geq 1024$)) | | ((D_p <1024) && (S_p <1024)) **then**
    v is a peer node, and labeled as type v_p
else
% v is a directed flow source and a directed flow destination
    if $\exists e_{v,u}, e_{u,v} \in E$ **then**
        v is a peer node, labeled v_p
    else
% v is a directed flow destination
        if $\exists e_{u,v} \in E$ **then**
            v is a server node, and labeled as v_s
        else
% v is a directed flow source
            if $\exists e_{v,u} \in E$ **then**
                v is a client node, and labeled as v_c
            end if
        end if
    end if
end if

- P-value, Z-score and frequency.
- Persistency. As defined in Section 6.2.2.3, this property refers to the frequency of appearance of a motif across several repetitive detection experiments. Therefore, a motif is considered to be persistent and thus relevant if it appears across each of those experiments.

The final motif set will be composed only by those considered relevant. It is relevant to note that this new condition reveals important since otherwise the training and test datasets would result in different motif sets and thus, would generate poor or inconsistent vertex profiles.

- Collision of motifs: The colliding motifs are to be identified and joined

during the vertex profile creation.

- Values for the parameters: Instead of using a binary value for each of the parameters, it is proposed to use the number of appearances of the corresponding motif to account for the volume of traffic and the proportions for each of the different motifs.

6.6.1 Evaluation of the Improvements

Based on the previous proposals, a set of experiments are carried out to evaluate them. Obviously, in order to compare the results, the same datasets used to train and evaluate the native method in the previous section are used. Thus, a set of 14 protocols is considered.

The particularization of the relevant motif selection was made as follows. First, similarly to the previous implementation of the native method, up to 10 application graphs per application composed by up to 100 nodes are considered. The extraction of the motifs was repeated 6 times, to account for the randomization of the graphs during the procedure. This provided 345 different motifs, which were filtered by using the same values for P-value, Z-score and frequency as those used in the implementation of the native method (Section 6.5.1.2, Table 6.3) and taking only those appearing in all the 6 repetitions. After that, a post-processing of the obtained motifs to join those repeated ones is carried out. The output of this phase is a set of n=24 motifs, shown in Figure 6.18 using the $< MotifID - Adjacency\_matrix >$ notation. Obviously, due to adding the persistency condition, a lower number of motifs are obtained in the improved method as compared to the reference one, although they are more significant due to this.

As shown in Figure 6.18, the motif set consists of 16 order-3 and 8 order-4 motifs. As expected, experimental results show that some motifs are exhibited by more than one application and that a single application might be associated with more than one motif.

Accordingly to the cardinality of the motif set, each of the hosts (nodes) in the training and testing sets is associated to a vertex profile composed by 24 parameters. For this, a preliminary assignment was made by applying the frequency of appearance proposal. Unfortunately, the resulting datasets were too sparse and thus not representative due to the lack of data. Consequently, we had to dismiss this approach and applied the binary parametrization as in the native method. It is worth to note at this point that the sizes of the training and testing datasets are huge when compared with other works in this field. This issue will be discussed in the conclusions.

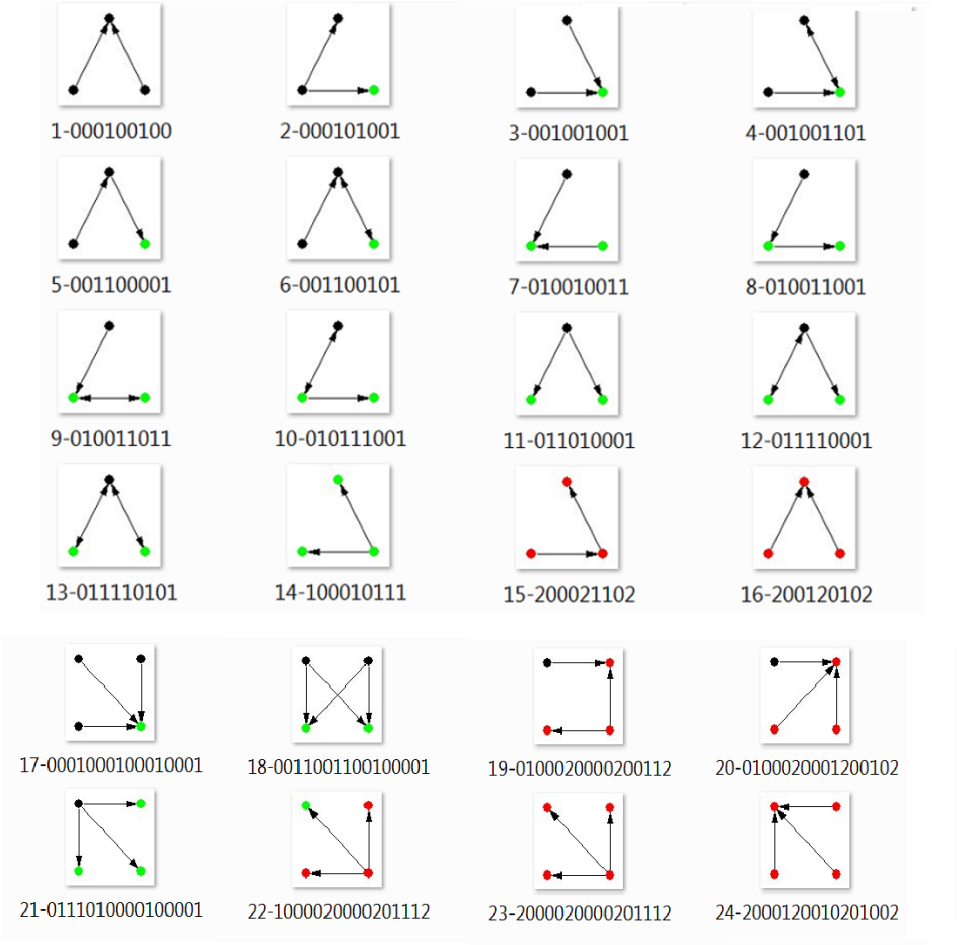


Figure 6.18: The 24 selected significant and persistent motifs in CS-A4: order-3 (1 through 16) and order-4 (17 through 24)

After obtaining the profiles for the testing dataset, the classification performance is evaluated.

Figure 6.19 shows the classification accuracy for the same set of application protocols and test sets used in the reference system. As shown in this figure, the improved method outperforms the native one on almost all of the analyzed protocols, achieving remarkable enhancements in the host classification accuracy for some of them as BitTorrent (36.7% increase), HTTP (37%), SSL (8.9%) and SSH (31%).

As a result, averaged for all protocols, 82% of accuracy was obtained with the improved method compared to 70% with the native one.

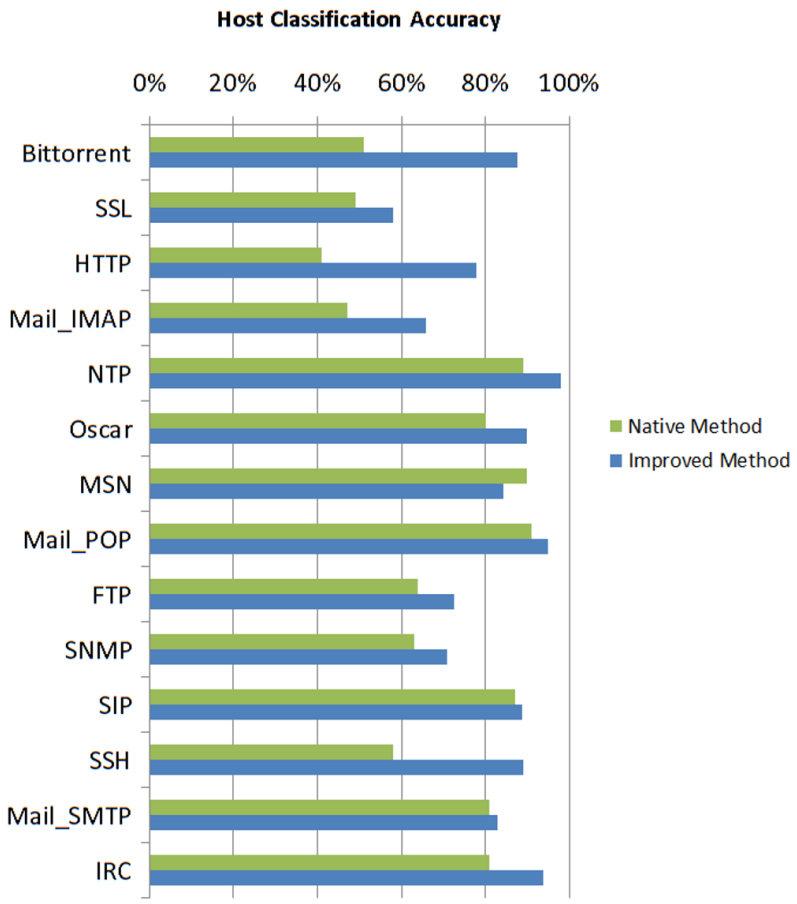


Figure 6.19: Improved v.s. native method per protocol accuracy results for the CS-B1 trained on CS-A4 dataset

6.6.2 Analysis and Discussions

In the light of the proposed improvements and experimental results, the applicability of the motif technique for host-based classification are discussed next.

Previous results show that the improved method outperforms [Allan, 2008]. Nevertheless, motif-based classification accuracy did not reach more than 82% on average, even after improvements. The main limitation seems to come from the fact that in some cases the same motif is exhibited by more than one application, thus not being discriminative enough among them. In addition, some applications might be associated with few or no motifs during motif detection phase. On the other hand, our results show that no single motif is indicative of a particular application, and the presence of applications with many motifs in common decreases the overall accuracy. Therefore, adding more discriminative

information during the motif selection and building of representative application profiles or models is required to enhance the system performance. Thus, two main lines of research are to be explored. On the one hand, richer profiles and an enhanced parametrization can provide the required additional information. Unfortunately, as previously mentioned, the volumes of required traffic to obtain statistically significant data is huge. On the other hand, it seems clear that it is the combination of motifs which is associated to the application and that some overlapping occurs. In this scenario, other classification methods different from KNN are probably better suited to represent each of the classes.

Another relevant issue concerns the potential online deployment of the method. The study presented so far and throughout this chapter focuses on assessing the classification accuracy of the system. Unfortunately, motif-based classification requires high computational costs, especially during the motif detection phase, and significant information gathering to obtain the required graphs from which to check the presence of each of the motifs. This introduces a significant delay in the classification of the activities of each of the hosts. Therefore, in its current state, the online deployment seems unfeasible.

Up to this level, host-based classification with motifs is assessed and improved. To be practical, motif-based classification should be extended to handle a multi-label classification context which is a new paradigm of host-based classification, as discussed next.

6.7 Extended Multi-label Motif-based Classification

In this section, motif-based classification is further extended to handle multi-label classification in order to be able to label a host as associated to more than one application type.

In the proposed extension, during the training phase, the motif detection process and selection are kept the same as in the previous approach (Section 6.4.2). Nevertheless, and from the perspective of the taxonomy presented in Chapter 2, key changes are proposed in the classification technique, the input and the output to adapt the system for multi-label mode.

6.7.1 Classifier Target

As mentioned, the motif-based classification methods described earlier are only applicable in single-label classification mode, that is, are able to classify a host as generating networking traffic associated just to a single class from the set of available applications. To handle multi-label classification, the classification targets should be evidently redefined.

Formally, let $\mathcal{L}$ be a set of disjoint labels, being M the cardinality of $\mathcal{L}$. The result of the classification process is the assignment of a set L of unique labels from $\mathcal{L}$, that is, $L \subseteq \mathcal{L}$, to the entity to be classified. Thus, in multi-class classification[17], which is the case considered so far, $M \geq 2$ and $|L| = 1$, whereas in multi-label classification $M \geq 2$ and $|L| > 1$.

In the considered multi-label scenario, the same set of application protocols used in Section 6.6 is targeted for classification although in a multi-label host classification context. Therefore, in this case, $\mathcal{L} = 14$ and $1 \leq |L| \leq 14$. Each of the 14 applications found in our dataset is associated to an index, $i = 1$ to $i = 14$, as shown in the first column of Table 6.8.

Therefore, in our case, in single label classification mode (Section 6.4.3), an unclassified host, e.g. H_x in Figure 6.13, is annotated with a single application label $L(H_x) = \{SSL\}$.

In the multi-label mode, the annotation consists of a set of application labels $L \subseteq \mathcal{L}$, where $\mathcal{L}$ is the set of all possible applications (see Table 6.7). For example, host H_x can be annotated with the subset $L(H_x) = \{HTTP, SSL\}$.

To ease the handling and evaluation of the labels, a vector of binary attributes[18] , $\mathbf{L_x}$, is used to represent the involvement of host H_x in any of the labels (applications). Thus, host H_x is assigned a *classes vector*:

$$L_x = [l_1, l_2 \cdots l_M] \tag{6.26}$$

where l_i is a binary attribute whose value is 1 if the i-th label (class) is present and 0 otherwise. For example, if the multi-label annotation subset of H_x is $L(H_x) = \{HTTP, SSL\}$, then the multi-label vector L_x becomes $[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$.

This representation can be used for associating labels to host H_x both for the classification and for the ground truth results. Thus, host H_x will be assigned two multi-label vectors:

- $L_x = [l_1^x, l_2^x, \cdots, l_M^x]$, as generated by the multi-label classifier, and
- $G_x = [g_1^x, g_2^x, \cdots, g_M^x]$, according to the ground truth results, generated by the DPI validation tool.

Table 6.7 summarizes the key differences between the evaluated classification methods in single and multi-label classification modes.

[17] Binary classification is a particular mode where $M = 2$ and $|L| = 1$.

[18] Decimal weighted attributes could be used to measure the level of host involvement in each of the detected applications. For instance, a 0.8 weight of attribute l_1 (BitTorrent application) would point to 80% of BitTorrent activities in terms of number of flows, connection time, etc. In this study, binary attributes are considered for simplicity reasons.

Table 6.7: Main differences between the motif-based classification in single vs. multi-label modes

Level	**Single-label Motif-based Classification Model**	**Multi-label Motif-based Classification Model**
Input	Vertex profile built considering single application is in use at any time	Vertex profile built considering multiple applications are in use at the same time
Technique	KNN algorithm based classification in single-label mode	Customized KNN algorithm in multi-label mode
Output	Single class, $L(H_x) = l, l \in \mathcal{L}$ where $\mathcal{L}$={BitTorrent, FTP, HTTP, IRC, Mail_IMAP, Mail_POP, Mail_SMTP, MSN, NTP, Oscar, SIP, SNMP, SSH, SSL}	Protocol sets, $L(H_x) = \{l_i \mid l_i \in \mathcal{L}\}$ where L={BitTorrent, FTP, HTTP, IRC, Mail_IMAP, Mail_POP, Mail_SMTP, MSN, NTP, Oscar, SIP, SNMP, SSH, SSL}

6.7.2 Multi-label Classification Input

In this subsection, extensions of the single-label motif-based classification model are shown at the input level. Similarly to most host-based classification works in the literature, the observed host behavior in [Allan, 2008] is seen from a single application perspective, that is, assuming that the host is exclusively involved in one application.

Ideally, a multi-label mode classifier should reveal the set of applications a host is involved with. Therefore, the vertex profile should be extended to answer this additional requirement. The purpose of this extension is to reveal simultaneous host involvements in each of the detected motifs, and consequently, in each application in the dataset. For example, the vertex profile depicted in Figure 6.12 shows that host H_1 is likely to be involved in two applications at the same time[19], HTTP and BitTorrent.

In order to implement single-label classification, in [Allan, 2008], host involvements in different applications are observed in separate. The underlying assumption is that a host is involved in a single application at a time[20]. As

[19]This is relative to the length of the observation time window. Host activities within the same time window are only considered. In this chapter, the time window is constrained by experimental and graph mining requirements, that is, to capture enough host exchanges, as explained in Section 6.4.2.

[20]The implementation details in [Allan, 2009] show that vertex profiles associated with the same host or IP address are assigned different identifiers within different application graphs. Although the dataset is obtained from real captures where hosts are likely engaged in multiple applications, [Allan, 2009] enforces a single-label host context.

such, by considering the targets defined in Section 6.7.1, neither of the two methods presented in [Allan, 2008] and improved in Section 6.6 can be applicable in multi-label mode.

In the extended model, we consider that hosts may be involved in multiple applications, which is the case for most real traffic captures. For this reason, vertex profiles had to be built differently. Thus, in the proposed extension, involvements of each host in various motifs are tracked across different application graphs according to the host IP address and cumulated altogether into one vertex profile.

Figure 6.20 shows examples of vertex profiles belonging to test host H_1 in single and multi-label classification. In this example, only 3 applications, FTP, HTTP and BitTorrent, are considered for illustration purposes. As shown in Figure 6.20a)), $H1$ is involved in the 3 protocols, but they are evaluated in three separate instances though belonging to the same host as it identity is not preserved across the different profiles, referred to as $H1\_1$,$H1\_2$ and $H1\_3$. However, the extended vertex profile, shown in Figure 6.20b)) represents the activities of all applications in which H_1 is engaged, by referring to H_1's IP address.

6.7.3 Multi-label Classification Technique

Hosts are to be classified according to the vertex profile showing their involvements in each application. Nevertheless, to handle multi-label classification, a customized algorithm is used at the technique level, as described next.

In most multi-label classification algorithms (e.g. ML-KNN [Zhang, 2007] and BR-KNN [Spyromitros, 2008]), the closest neighbors are considered to infer the set of classes of the sample to be classified. For this, training samples are usually multi-labeled instances. This way, the same kind of data is used both for training and evaluation, as in both cases each sample is associated to multiple classes.

Even in this case, there exist different approaches to assign the set of classes to a test sample. In the most simple one, the sample to be classified is assigned the same set of labels (classes) as those the nearest neighbor has. Nevertheless, other more complex approaches are preferred. For example, ML-KNN [Zhang, 2007] uses the KNN algorithm independently for each class l: It finds the K nearest points to the test instance and considers those that are labeled at least with class l as positive votes and the rest as negative. The class l is assigned if the result of the voting is positive.

Nevertheless, to handle multi-label classification we followed a different approach. The idea is to infer the classes vector associated to a host based

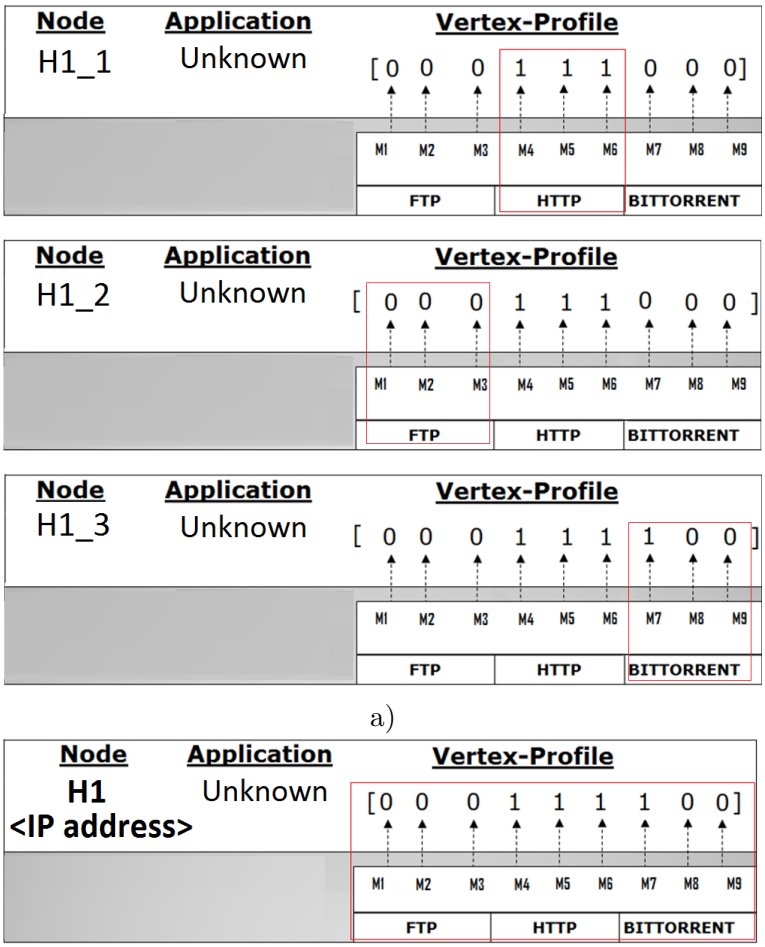


Figure 6.20: Building the vertex profile in: a) single-label mode vs. b) extended Multi-label

on the output provided by various KNN models operating in single-label and based on single-labeled training samples.

To achieve this, we had to use KNN in regression mode and to infer the multi-label annotation based on the closest single-labeled neighbors. Fortunately, KNN in regression mode is able to generate the so called Protocol Probability Distribution (PPD) for each protocol, given a test sample, according to the labels and to the inverse distances to each of the closest neighbors, as will be explained next. Based on this output, a customized ML-KNN version is used accordingly to infer the classes vector, L_x. The procedure is detailed in Algorithm 6.2.

As shown in Algorithm 6.2, in regression mode, KNN computes a weighted

Algorithm 6.2 The customized ML-KNN multi-label classification

% Notation:
% $\mathcal{M}$: Set of samples (hosts) in the model
% H_x: Host to classify
% $\overrightarrow{V_J}$: Profile for host H_j
% $L_j = l_t^j | 1 \leq t \leq M$: Classes vector for host H_j
% K: Number of neighbors
% N: Set of nearest neighbors
% M: Number of classes
% δ: Dirac's delta

% Initialization
$L_x = 0$
$N = \emptyset$
% Evaluation of distances to model profiles
$D := \{d_l = d(\overrightarrow{V_x}, \overrightarrow{V_l}) | H_l \in \mathcal{M}\}$

% Selection of k nearest neighbors
% Repeat K times
$N := N \cup \{H_{argmin(D)}\}$
$d_{argmin(D)} = \infty$

% Evaluate the total weights (inverse distances) of k nearest neighbors
$W = \sum_{t \in N} \frac{1}{d(\overrightarrow{V_x}, \overrightarrow{V_t})}$

% Evaluate the Protocol Probability Distribution (regression mode)
$P := \{ppd_j | 1 \leq j \leq M\}$ being $ppd_j = \frac{1}{W} \sum_{t \in N} \frac{1}{d(\overrightarrow{V_x}, \overrightarrow{V_t})} \delta(l_j^t = 1)$

% Assign labels
$L_x := \{l_j^x | 1 \leq j \leq M\}$ being $l_j^x = \begin{cases} 1 & \text{if } ppd_j > 0 \\ 0 & \text{otherwise} \end{cases}$

average of the K nearest neighbors according to the inverse of their distance to the sample to classify. Thus, the distances from each of the samples in the model to the test profile are ordered in increasing order, and then, an inverse distance weighted average (referred here to as PPD) is calculated using only the K-nearest neighbors (K lowest distances). Therefore, rather than a majority voting based on the nearest neighbors labels, the multi-label annotation of an unclassified host is directly obtained from the PPD values for each of the

Table 6.8: Protocol indexes and example of the use of the customized ML-KNN

Application Protocol	*ppd* values for H_x	l_i values for H_x
BitTorrent (i=1)	0.01	1
FTP (i=2)	0	0
HTTP (i=3)	*0.9	1
IRC (i=4)	0	0
Mail_IMAP (i=5)	0	0
Mail_POP (i=6)	0	0
Mail_SMTP (i=7)	0	0
MSN (i=8)	0	0
NTP (i=9)	0	0
Oscar (i=10)	0	0
SIP (i=11)	0	0
SNMP (i=12)	0	0
SSH (i=13)	0.009	1
SSL (i=14)	0.081	1

protocols [21]. An example including the values for the PPD values is provided in Table 6.8.

According to the procedure, a value $ppd_i > 0$ for application i indicates that at least one of the nearest neighbors to the test sample is labeled with that application and, therefore, according to the model, the host is involved with activities belonging to application i.

In our case, as previously stated, binary attributes are considered. Thus, the components of the classes vector, l_i, will be assigned to 1 if $ppd_i > 0$ and 0 if it is 0. Attributes of the classes vector, L_x, for the sample node H_x in Figure 6.13 are depicted accordingly in the third column of Table 6.8. Obviously, as previously stated, a more sophisticated approach is possible if non binary values are used for the classes vector. In fact, it is almost straightforward to use percentages of involvement of a node from the ppd_i values itself. This last option has not been considered in a first approach due to the lack of data to obtain a significant result.

On the other hand, it is relevant to mention that, in the current approach, a host is labeled as involved in i-th protocol if just a single one of the K nearest neighbors is labeled as i-th protocol, independently of its distance. It is the

[21] In Table 6.8, *ppd* values are used in regression mode, to predict the single class label for a host having the highest probability value. The * mark indicates the classifier's decision in single-label mode. If an instance is unclassified, the returned vector elements must be all zero.

ppd_i value which takes this distance into account. Thus, it can be advisable to set a minimum threshold for ppd_i instead of its value being not null.

Therefore, it is almost evident that the obtained classes vector L_x strongly depends on the values of K and *ppd*. Both of these parameters affect the number of protocol annotations in the multi-label classification format. If a high threshold for *ppd* is used, low probability protocols might not be included in the classes vector of any host (e.g. BitTorrent in Table 6.8). However, at higher values of K, more annotations are likely to be added to the multi-label classification.

Referring back to the example in Figure 6.13, if we set the threshold $ppd > 0$, and for $K = 3$, H_x is considered as involved in activities related to HTTP and SSL, therefore, $L_x = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$, whereas for $K = 5$, BitTorrent protocol is added and $L_x = [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]$.

Similarly, setting a minimal value on *ppd* will further affect the obtained classes vector. For example, for a fixed value of $K = 5$, if the threshold is set to $ppd \geq 0.001$ $L_x = [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]$, whereas for a threshold $ppd \geq 0.08$ then $L_x = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$.

To experimentally implement the customized ML-KNN, the classifier components are built and customized in WEKA based on the default Ibk-Classifier, where KNN is implemented in single-label mode in [WEKA, 2013].

6.7.4 Evaluation Metrics

A relevant issue associated with multi-label extension is the choice of the evaluation metrics and methodology. Multi-label classification requires different metrics than those used in traditional single-label classification. Common multi-label metrics [Tsoumakas, 2006] include measures as Hamming Loss, accuracy, or recall. most of which considers that training points are multi-labeled.

In this study, the classification accuracy [Tsoumakas, 2006] in multi-label mode is used to evaluate the proposed classifier. Thus, considering the multi-label classified hosts, the accuracy is measured in separate for each individual application.

For this purpose, we refer to problem transformation [Boutell, 2004] where M binary classifiers, $C_i : X \rightarrow \{i, \bar{i}\}$, one for each different label i in $\mathcal{M}$, are considered instead of one multi-label classifier. Consequently, the original data set is transformed into M data sets, $\mathcal{M}_i$, each containing all the samples from the original dataset, labeled as application i if the labels of the original sample contained i, and as $\bar{i}$ otherwise. Therefore, in multi-label evaluation mode, the accuracy in classifying application i (or $Accuracy_i$) equals to the accuracy of

the binary classifier C_i.

Formally, let A be the set of all the hosts, H_x, in the test set. As mentioned previously, every host $H_x \in A$ is associated to two classes vectors: the classes assigned by the classifier, L_x, and the real classes as assigned by DPI in our case, G_x.

A classification decision, in L_x, associated with host, $H_x \in A$ is necessarily in one of the following situations (Chapter 3): it is a TP, a FP, a FN or a TN, according to:

$$\begin{cases} TP, & \text{if } (l_i^x = 1 \ \& \ g_i^x = 1) \\ FP, & \text{if } (l_i^x = 1 \ \& \ g_i^x = 0) \\ FN, & \text{if } (l_i^x = 0 \ \& \ g_i^x = 1) \\ TN, & \text{if } (l_i^x = 0 \ \& \ g_i^x = 0) \end{cases} \tag{6.27}$$

Consequently, four measures can be defined for each application, i, as:

$$TP_i = \sum_{H_x \in A} \delta(l_i^x = 1)\delta(g_i^x = 1) \tag{6.28a}$$

$$FP_i = \sum_{H_x \in A} \delta(l_i^x = 1)\delta(g_i^x = 0) \tag{6.28b}$$

$$FN_i = \sum_{H_x \in A} \delta(l_i^x = 0)\delta(g_i^x = 1) \tag{6.28c}$$

$$TN_i = \sum_{H_x \in A} \delta(l_i^x = 0)\delta(g_i^x = 0) \tag{6.28d}$$

Some evaluation metrics can be defined from this measures (Chapter 3). In particular, we are interested in the accuracy for $Application_i$, that can be obtained as:

$$Accuracy_i = \frac{|TP_i|}{|TP_i| + |FP_i|} = \frac{\sum_{H_x \in A} \delta(l_i^x = 1)\delta(g_i^x = 1)}{\sum_{H_x \in A} \delta(l_i^x = 1)} \tag{6.29}$$

The overall multi-label classification accuracy of the classifier C_i can be evaluated as:

$$Accuracy = \frac{1}{M} \cdot \sum_{i=1}^{M} Accuracy_i \tag{6.30}$$

The extended multi-label (Section 6.7) and the single-label motif-based classification models (Section 6.6) have different classification targets. Despite

of this fact, the problem transformation methodology used in multi-label mode makes it easier to compare both disciplines (single and multi-label), using the accuracy metric defined on a per application basis. In the next section, the proposed multi-label host classifier is evaluated and then compared to the single-label classifier.

6.7.5 Experimental Results

In this section, multi-label classification results are shown for the extended model presented in Section 6.7. For comparison purposes, the extended multi-label classifier is tested on the same datasets used with the native system (Section 6.6.1): the CS-A4 dataset is used for training while CS-B1 is used for testing. Using the same training dataset and the same conditions for significant motif detection, the same set of significant motifs detected in single-label mode (Section 6.6) is used in multi-label mode.

As mentioned earlier, both K and *ppd* parameters affect the number of protocol annotations in the classes vectors. As previously discussed, it is necessary to find a compromise between the number of protocols considered in each classes vector and their representativeness. Thus, after some preliminary tests, we chose a threshold of 0.01 for *ppd* and $K = 3$.

6.7.5.1 Multi-label Classification Results

Host classification accuracy (Equation 6.29) is shown in Figure 6.21 on a per application basis, for the native, improved (Section 6.6) and multi-label methods.

It can be clearly observed in Figure 6.21 that in multi-label mode, the accuracy (Equation 6.30) is improved to 91.71%, averaged on all protocols, as compared to 82% as previously obtained with the improved method, or the 69.81% obtained with the native one in single-label classification mode.

Setting a threshold (red line in Figure 6.21) of 90%, the multi-label version of motif based classification seems to be convenient for most protocols like BitTorrent, SSH, Oscar and IRC. For few other protocols like Mail_POP and NTP, the accuracy is slightly degraded compared to single-label mode. Moreover, when applied in multi-label mode, host-based classification accuracy using motifs has been improved. To further illustrate this observation, Figure 6.22 shows the same results in binary classification mode: P2P vs. non-P2P.

The results in Figure 6.22 show again that extending host-based classification using motifs to multi-label mode is particularly improving P2P classification with 99.12% of accuracy[22], compared to 91.15% for non-P2P applications.

[22]Although not shown in figures, BitTorrent accuracy is maintained above 95% regardless of

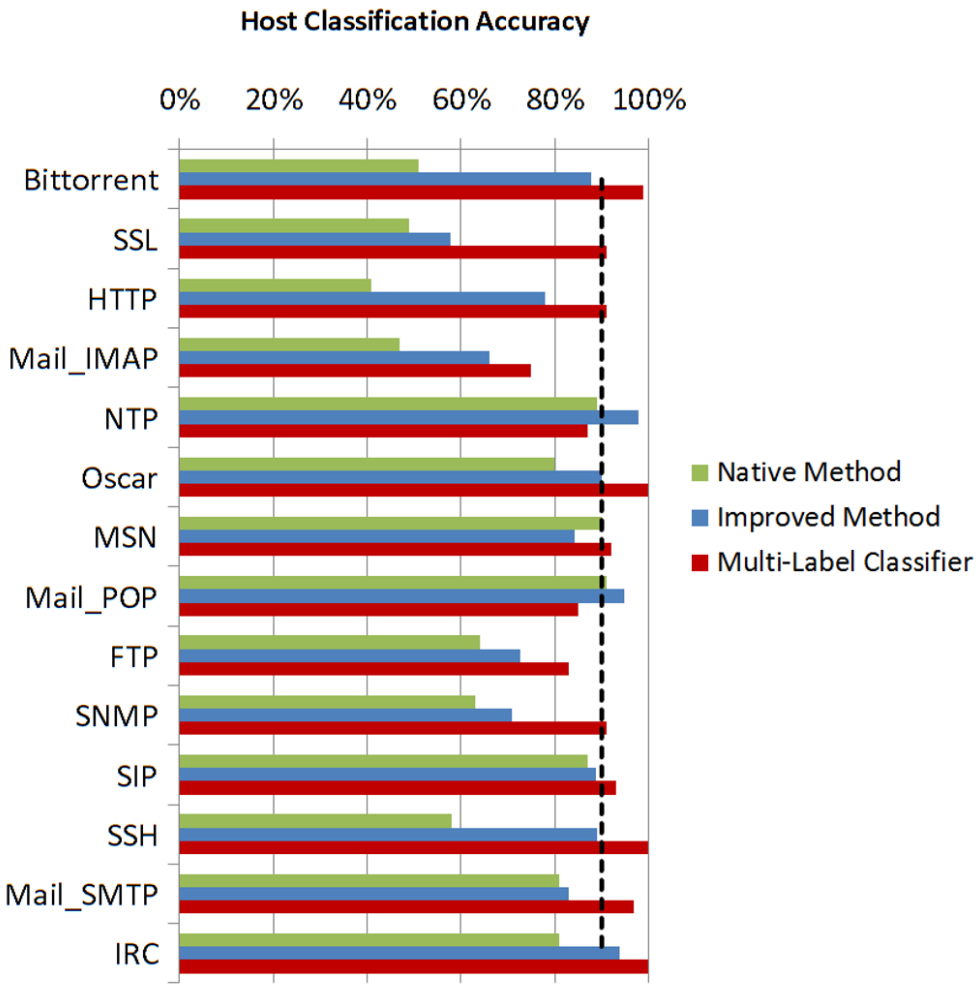


Figure 6.21: Per protocol accuracy for native, improved and multi-label models for CS-B1 dataset trained on CS-A4

Another relevant result is the average number of labels per host, both in the multi-label classification results and in the ground truth results. To enhance the classifier's performance, the average values in these results should be close, as much as possible. However, the average number of labels per host strongly depends on the choices of K and PPD values, as shown previously. In our case, for $K = 3$ and $PPD > 0.01$, the average number of labels per host in the ground truth is 2.7, compared to 2.1 in classification.

Further analysis of these results is provided in the next section.

PPD and K values.

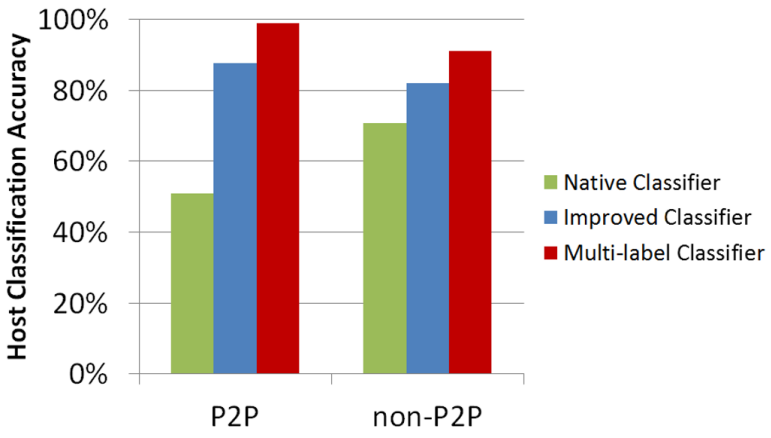


Figure 6.22: P2P v.s. non-P2P classification accuracy for CS-B1 trained on CS-A4

6.7.6 Analysis and Discussions

In this subsection, multi-label classification results are further discussed with regard to different aspects. They are:

- Extensions to multi-label mode: As per our experiments, multi-label mode is an advanced context for host classification. Extending single-label methods in the literature requires, as we have shown, extensions at the three different levels of the classifier.

  It is also important to note that in most literature work, including motif-based classification, the tested traffic is sometimes based on artificially generated traffic where the single-application behavior may be applicable. However, in real traffic captures, the host behavior is implicitly applicable to the multi-applications context. As we have shown, even when using real traffic captures, it is the classifier's design that dictates whether the classification context is in single or multi-label one.

  According to our experiments, although an observed host, in the traffic capture, might be engaged in multiple applications, considering that host's activities for each application in separate simulates single-label mode, while considering these activities altogether is supposed to simulate the multi-label mode. Particularly, taking motif-based classification as an example, the main difference between single and multi-label modes relies in the way a vertex profile is built, although the same set of motifs is used during training.

- Classification accuracy improvement in multi-label mode: In multi-label

classification mode, host-based classification using motifs showed better performance for most applications, especially for P2P. For few non-P2P protocols, accuracy is slightly degraded compared to single mode classification. For these protocols, host classification, using regression in multi-label mode, did not yield the same results as in single mode, where voting is used, that is, the classes with greater PPD values were not necessarily the most frequent ones within the set of closest neighbors associated with a host. For such cases, KNN with voting showed better results than with regression.

On the other hand, the reason why P2P applications were detectable with very high accuracy, both in single and multi-label modes, might be referred back to the distinctive patterns of P2P traffic exchanges compared to other applications. In fact, P2P motifs were distinguished with the least number of collisions compared to others. For this reason, they showed very high discriminative power for P2P detection, as compared to other applications.

- Classifier tuning in multi-label mode: The multi-label classifier's performance proposed in this chapter is strongly related to the values of both K and PPD. From one hand, an increased value of K (or a decreased value of PPD) is supposed to include a higher number of protocols in the classes vector of each host. Although this might raise the problem of predominant protocols in the training datasets, it is supposed, however, to increase the accuracy for some protocols for which additional closest neighbors, and therefore class labels, are the more likely to be included in the classes vector. Consequently, the optimal choices of these values depend on the training dataset.

The general inference underlined by our study, in the wider context, is that host-based classification methods described in the literature, should be re-evaluated in multi-label classification contexts. This will permit to assess traffic classification in real case scenarios, where multiple applications are simultaneously used by each host. From this standpoint, and based on our acquired experience, we believe that to increase the confidence in the results obtained with most host classifiers found in the literature, their evaluation in multi-label mode should be considered to simulate real traffic environments.

The analysis presented in previous chapters highlighted on key findings in payload, flow, and host-based classification techniques. This chapter discussed particularly the applicability and evaluation of host-based classification in multi-label mode. With this chapter, experimental and assessment parts of this thesis are completed.

The main outcomes and summary findings are presented in the next chapter, where rooms for future enhancements are highlighted for future works.

Chapter 7

Conclusions and Future Work

This chapter concludes the main results and the key findings obtained throughout this thesis. At the end of this chapter, discussions are issued in light of the obtained results, opening the doors for future works and follow-ups.

7.1 Conclusions

Identifying Internet traffic applications is a challenging and hot research topic for both network security and management. Throughout this thesis, we have presented an analytical review of the different approaches in the literature of Internet traffic identification. We criticized the current state of art, assessed methods for main research trends, proposed and experimentally evaluated improvements with new classification models. Our investigations highlighted on the need to reconsider some of the conventional concepts in the literature, starting by the underlined weaknesses of the existing taxonomies and reaching advanced and complex classification models in the current state of the art.

The main outcomes that can be concluded from this thesis can be summarized as follows:

- **Multilateral Taxonomy:** As the research community has neither defined a consistent terminology nor elected the best traffic identification model yet, the multilateral taxonomy proposed in this thesis helps in promoting and unifying the research efforts. In fact, with the diversity in the deployed techniques and data formats, referring to one consolidated terminology is useful for standardized benchmarks and comparison purposes.

  Amongst the different proposed categories, payload-based methods, despite of their high accuracy results, are less recommended when privacy

is the major concern and present scalability problems for high speed networks. As an alternative, non-payload methods are effective for protecting the user privacy and when the payload is encrypted. In this context, statistical and machine learning techniques were widely used. Graphical methods are used often as helper technologies. Although a slight bias toward hybrid techniques and decision trees was detected, our investigation has underlined the lack of an optimal model elected for traffic classification.

From the perspective of the proposed multilateral taxonomy, main research requirements were highlighted in this thesis. Various aspects of the existing and future methods were categorized at the input, technique and output levels.

To study payload based methods, we chose the most commonly known DPI classifiers. For blind classification, we chose to analyze flow and host, based methods as a trade-off between too fine-grained (packet-based) and too coarse-grained (community-based) classification methods.

- **An Experimental Setup for Traffic Classification:** To achieve our stated purposes, an experimental setup, based on real traffic captures, was prepared. To build this setup, several traffic sets were captured from a real network environment of a multi-branch institution, then, the DPI tool was customized to generate, for each of these captures, parametrization data and ground truth results that are used for the assessment of the proposals. It is remarkable the availability of a high volume of data and the inclusion of full payload in these captures.

- **DPI Optimization:** Payload-based classification was assessed and optimized using sampling techniques, applied to a DPI classifier. In the light of the obtained experiments, we proposed an optimization method for DPI classification with which payload disclosure is minimized to protect the user privacy while accuracy is maintained at acceptable levels.

  Specifically, we proposed a general methodology for optimizing DPI by integrating per-flow and per-packet sampling methods. Our results highlighted the importance of the first flow packets for DPI classification as holding most of the application signatures DPI is looking for. According to our testbed, a significant gain in classification time (12.47%) can be obtained by inspecting less payload (the first 1792 bytes per flow) while maintaining around 95% of the DP for most of the tested protocols.

  The major outcome of the payload assessment study is that we have proven the feasibility of DPI optimization through sampling, and, based

on experimental results, we managed to elect the best sampling model for DPI classification achieving the best possible trade-offs between performance and privacy considerations.

- **Identification Based on Message Size Analysis:** A novel flow-based blind identification model is proposed, relying on application-layer messages and using ML techniques. The proposed classifier uses the sizes of the initial messages exchanged between the hosts involved in the communication as inputs. Unlike other similar approaches, our work focused on the messages, not the packets; which is considered a differential characteristic of a protocol is the sequence of sizes of the first messages, not the sizes of the initially exchanged packets.

  Another differential characteristic of the proposed system is the use of multi-modal distributions in an attempt to summarize all the possible methods included in a protocol in a single model. Evaluated on a real captured dataset the proposed classifier showed promising results with 98% to 99% of recall levels.

  Our approach analyzes application layer messages without breaching the user privacy. Most of the previous approaches in the literature either disclosed the user payload to match application layer signatures or analyzed traffic properties at the network and transport layers with less discriminative power. An important outcome of this study is that we proved that the actual semantics of user sessions at the application layer have discriminative features that can be explored without breaching the user privacy.

- **Multi-label Host Classification:** A new context of host-based blind identification model is tested and discussed. In the new context, we discuss multi-label host classification to simulate real case scenarios where more than one application is often in use by the same host. For this purpose, we chose first to improve a host based method relying on graph mining (or motifs), second, we extend the method for multi-label identification. The method's applicability in the multi-label was discussed and conclusions were inferred for multi-label host classification in general.

  With this study, we were able to particularly surpass some of the shortcomings of native motif-based classification and, most importantly, to derive a multi-label version of the method that we evaluate for host identification. In the multi-label context, our results show classification accuracy improvement for most protocols. Particularly, P2P applications

were detectable with the highest accuracy due to the the distinctive patterns of P2P traffic exchanges.

The novelty of our work resides in tackling, for the first time in the literature, host-based classification as a multi-label classification problem. From this standpoint, and based on our acquired experience, we believe that in order to increase the confidence in the results obtained with most host classifiers found in the literature, their evaluation in multi-label mode should be considered.

7.2 Future Work

Throughout this thesis, many aspects of the classification methods were studied and analyzed. Although many of the obtained results were promising in most of the cases, our work is still opening the door for future follow-ups and additional improvements. Based on our gained experience, we present in this section our perspective for the future directions and follow-ups to the work presented throughout this thesis and the main research trends in the field.

First, in regard to the payload based study of this work, future benchmarks in DPI optimization should assess various tools with respect to larger traffic sets and many additional application protocols. Once the minimal useful DPI input is defined, algorithm and even hardware-based optimizers are supposed to become more efficient.

Critical DPI scenarios should be studied addressing cases where the first packets are delayed or lost, or when fake application signatures are intentionally inserted into the beginning of a flow to disguise network monitoring devices. Automating DPI signature extraction and adapting DPI sampling to their location should take more attention, especially when considering the steady emergence of new Internet applications.

As being the reference method in the literature, DPI should become more and more robust and accurate, yet with affordable computational costs. For this purpose, future DPI methods might need to explore the actual semantics of user sessions, not only the syntax and signatures of application protocols. Coping with newly emerging Internet applications and very high link speeds is essentially relevant to any future research related to DPI classification.

Second, in regard to the blind flow-based classification method proposed in this work, more extensive experiments using larger datasets are required in order to improve the message based classification system, and to increase the confidence and representativeness of the models. Future work should also enable the use of new and better heuristics. This is not an easy challenge, as

the data to be used has to be properly labeled and has to be big enough. In our experiments, more than 200 GB of labeled data has proven to be insufficient for some protocols and only 19 of them could be used with some confidence. We also suggested as candidate for future work the tracking of host-to-host activities in order to affect accordingly the prior probabilities of the Bayesian classifier. We expect interesting improvements from this approach, though at the cost of acceptable computational overhead.

Exploring new traffic features is essential for future blind traffic classification methods. In contrast to the conventional traffic properties obtained at the network and transport layers, application layer characteristics, beyond message sizes, can be regarded as the new generation of discriminative features that can be used for blind traffic classification.

Third, in regard to the blind host-based classification model proposed in this thesis, future works should extend many of the existing single-label host classifiers in the literature to multi-label mode. Though tested with a single method, namely motifs, our investigations have the potential to radically change the conventional view of single-label host classification. Hence, many existing host classifiers will need to be assessed in multi-label classification scenarios. As we believe that many of the existing literature works would decrease in performance when assessed in the context of multi-label classification, we urge future researchers to give more attention to the multi-label mode in traffic classification problems.

Moreover, multi-label classification can be extended further as to include weighted input and output formats and other targets (e.g. flows, host communities). In such scenario, the classification results of an object consist of probabilistic, rather than deterministic, values regarding each application label. Furthermore, the use of extended vertex profiles with non-binary values as inputs, as well as including additional information regarding the observed interactions, would significantly increase the semantic information for the classification phase.

Fourth, in the particular context of graph mining and motif-based classification, many rooms for enhancements exist. For example, motif attributes should be normalized as to mitigate any bias effect to applications having a higher number of motifs. Additional distance types should be tested in measuring the vertex profile similarity for motif KNN based classification (e.g. Chebyshev, Cityblock, Manhattan). Future accuracy improvements might need to refer to more complex motif structures such as weighted, augmented and/or higher order motif. For example, supplementary information related to both edges and nodes (e.g. the size of the transfer, number of packets, etc.) could be introduced into the motif structure. However, an important issue to be raised

concerns the location of the monitoring point, which is relevant for capturing a number of host interactions enough to build higher order motifs. Thus, obtaining higher order motifs might imply the deployment of several monitoring points at different locations.

Additionally, one of their current limitations is the lack of temporal related properties that is able to represent dynamic network interactions and their evolution over time. In the current approach, motifs are observed within a relatively small time window. In order to include time related properties in motif-based analysis, network interactions should be observed during larger time periods, and the motif structure might need to be extended accordingly.

Finally, opening new dimensions in the field of traffic classification is vital for managing future networks. Future traffic classifiers' designs should be adapted for next-generation technologies (IPv6, VDI, CCN, SDN, etc.) as being part of their network management portfolio in the long-term. Nevertheless, many trade-offs in applicability, performance, and privacy protection should be considered. Studying a relevant design and analyzing efficiency is an important part of any future work in this regard.

Appendices

Appendix A1

The Customized DPI Tool

In this appendix, we detail the experimental setup (Chapter 3) used to evaluate each of the traffic classification methods addressed throughout this thesis. Given a preprocessed traffic capture file, an essential part of the experimental setup, besides the obtention of the classification results, is to extract a large set of traffic parameters that will be used for classification purposes, as well as to label all or part of the traffic according to their class.

Specifically, we detail the labeling and parametrization blocks of the experimental setup shown in Chapter 3. The operational details shown in this appendix include file formats and tools involved in the appropriate implementation of labeling and parametrization that are based on a customized version of *openDPI* [nDPI, 2013], (`dpi_flows`). The customized version generates a list of protocol annotated packets and flows, together with a rich set of parametrization data.

The customized tool diagram is shown in Figure A1.1 where 3 programs were developed: `dpi_flows`, `dpi_bin2asc` and `dpi_bin2params`, generating files in different formats as detailed next.
The customized DPI programs's, their input and output files are detailed as follows:

- dpi_flows: which processes PCAP files and annotates both packets and flows, according to openDPI library, using a customized output format. It takes PCAP files as input, and generates a summary of the number of flows per protocol in binary and ASCII formats.

- dpi_bin2params: which processes the binary files generated by dpi_flows and parameterizes the flows. It takes binary files as input and generates ASCII file format (.PARAM).

- dpi_bin2asc: which converts files from binary to ASCII format.

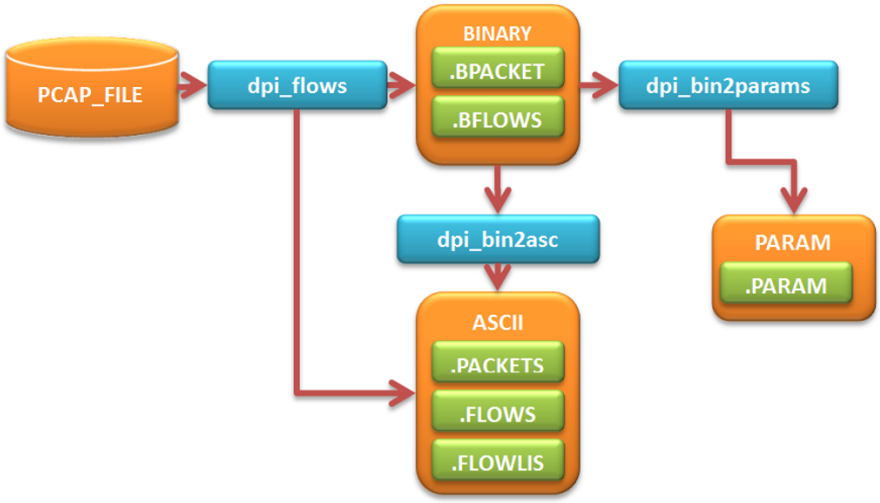


Figure A1.1: Flow diagram of the process and tools used to label and parameterize flows

The used file types are detailed as follows:

- PCAP: These are standard PCAP files.
- DPI_Binary: List of packets and packet related information, in binary format (a non-readable memory dump, used for fast data handling).
- DPI_Binary: List of flows containing flow related information, in binary format (a non-readable memory dump, used for fast data handling).
- DPI_ASCII: List of packets and packet related information, in ASCII format (readable by any text editor).
- DPI_ASCII: List of flows containing flow related information, in ASCII format (readable by any text editor).
- DPI_ASCII: List of packets in each flow in ASCII format (readable by any text editor).
- PARAM: List of parameters for each flow in ASCII format (readable by any text editor).

An example of .flow ASCII file can be shown as follows:

```
# Flows from file [/home/data/f1-test.pcap]
# Date:  [14/5/2013 8:16:55]
# Number of flows:  [2]
# Format:  Id; Detected; IP1; IP2 ; Port1; Port2; Prot; Npac;
NUp; NDown; Dir; Packets_size; Packets_size_up; Packets_size_down;
Payload_size; Payload_size_up; Payload_size_down; First; Last
1; unknown; 172.16.13.4; 172.16.13.9; 22; 36557; TCP; 3; 2; 1; UP;
382; 336; 46; 256; 256; 0; 1275473232292; 1275473232293
2; HTTP; 10.81.140.1; 92.122.212.32; 10243; 80; TCP; 21; 10; 11; DOWN;
12818; 1587; 11231; 11930; 1145; 10785; 1275473232293; 1275473398474
```

An example of .flowlist ASCII file can be shown as follows:

```
# Flows lists from file [/home/data/f1-test.pcap]
# Date:  [14/5/2013 8:16:55]
# Number of flows:  [2] # Format:  Id; Detected; IP1; IP2; Port1;
Port2; Npackets_in_flow; Packet_list
1; unknown; 172.16.13.4; 172.16.13.9; 22; 36557; 3; 1; 2; 6
2; HTTP; 10.81.140.1; 92.122.212.32; 10243; 80; 21; 3; 7; 8; 1062;
158040; 160758; 160760; 160761; 160975; 161901; 597111; 598994;
598995; 599034; 631141; 633149; 633152; 633358; 633359; 635342; 635800
```

An example of .packets ASCII file can be shown as follows:

```
# Packets from file [/home/data/f1-test.pcap]
# Date:  [14/5/2013 8:16:55]
# Number of packets:  [2] (1 fragmented or not IP)
#Format:  Id; Detected_protocol; Flow_id; IP1; IP2; port1; port2; Prot;
Direction; Packet_size; Payload_size; Timestamp; Flags[; ACTIVE_FLAGS]
1; unknown; 1; 172.16.13.4; 172.16.13.9; 22; 36557; TCP; UP; 168; 128;
1275473232292; 24; PUSH; ACK
2; HTTP; 1; 172.16.13.4; 172.16.13.9; 22; 36557; TCP; UP; 168; 128;
1275473232292; 24; PUSH; ACK
```

An example of .param parametrization file can be shown as follows:

```
### Input file [f1-test.param] ### Processed [14/5/2013 9:16:55]
### PARAMETERS: identification, basic_stats ### FROM: pcapfile
[f1-test.pcap] # Flows from file [f1-test.pcap] # Date:  [14/5/2013
9:16:55] # Number of flows:  [2] # Format:  BINARY OUTPUT (MEMORY
DUMP) # Packets from file [f1-test.pcap] # Date:  [14/5/2013
9:16:55] # Number of packets:  [24] (3 fragmented or not IP) #
Format:  FLOW_ID; ID_PROT; IP_LOW; IP_UPPPER; PORT1; PORT2; PROT;
DIR; FIRST_TIME; LAST_TIME; NPACKETS; NPACKETS_UP; NPACKETS_DOWN;
PACKETS_SIZE; PACKETS_SIZE_UP; PACKETS_SIZE_DOWN; PAYLOAD_SIZE;
PAYLOAD_SIZE_UP; PAYLOAD_SIZE_DOWN; DURATION; MEAN_PACKETS_SIZE;
MEAN_PACKETS_SIZE_UP; MEAN_PACKETS_SIZE_DOWN; MEAN_INTERARRIVAL;
MEAN_INTERARRIVAL_UP; MEAN_INTERARRIVAL_DOWN; N_SIGNALING;
N_SIGNALING_UP; N_SIGNALING_DOWN; SHORT_PACKETS; SHORT_PACKETS_UP;
SHORT_PACKETS_DOWN; LONG_PACKETS; LONG_PACKETS_UP; LONG_PACKETS_DOWN;
MAX_INTERARRIVAL; MAX_INTERARRIVAL_UP; MAX_INTERARRIVAL_DOWN;
MIN_INTERARRIVAL; MIN_INTERARRIVAL_UP; MIN_INTERARRIVAL_DOWN; MAXLEN;
MAXLEN_UP; MAXLEN_DOWN; MINLEN; MINLEN_UP; MINLEN_DOWN; NACKS; NFIN;
NSYN; NRST; NPUSH; NURG; NECE; NCWR; NACK_UP; NACK_DOWN; NFIN_UP;
NFIN_DOWN; NRST_UP; NRST_DOWN
1; unknown; 172.16.13.4; 172.16.11.9; 22; 36557; TCP; UP;
1275473232292911; 1275473232293779; 3; 2; 1; 382; 336; 46; 262; 256;
6; 868; 127.333336; 168.000000; 46.000000; 434.000000; 82.000000;
0.000000; 0; 0; 0; 1; 0; 1; 2; 2; 0; 786; 82; 0; 82; 82; 4294967295;
168; 168; 46; 46; 168; 46; 3; 0; 0; 0; 2; 0; 0; 0; 2; 1; 0; 0; 0; 0
2; HTTP; 10.81.140.1; 92.122.212.32; 10243; 80; TCP; DOWN;
1275473232293600; 1275473398474472; 21; 10; 11; 12818; 1587; 11231;
11978; 1187; 10791; 166180872; 610.380981; 158.699997; 1021.000000;
8309043.500000; 18434614.000000; 16606995.000000; 0; 0; 0; 9; 7; 2;
12; 3; 9; 113468180; 113468180; 114245394; 3; 3; 7; 1420; 445; 1420;
46; 46; 46; 21; 2; 0; 0; 7; 0; 0; 0; 10; 11; 1; 1; 0; 0
```

An example of .packets ASCII file format used for message analysis (Chapter 5) can be shown as follows:

```
# Packets from file [/home/data/f1-test.pcap]
# Date:  [14/5/2013 8:16:55]
# Number of packets:  [2] (1 fragmented or not IP)
# Format:  Id; Detected_protocol; Flow_id; IP1; IP2 ; Port1; Port2;
Prot; Direction; IP_IHL; IP_ITL; Timestamp; TCP_OFFSET; TCP_SEQ;
TCP_ACK; Flags
1; FTP; 1; 209.85.135.103; 192.168.1.190; 80; 1196; TCP; DOWN; 5;
13312; 1252655468737751; 8; 1977319717; 0; 2
2; HTTP; 1; 209.85.135.103; 192.168.1.190; 80; 1196; TCP; DOWN; 5;
39938; 1252655468781809; 5; 1994096933; 1; 24
```

Appendix A2

Resumen

A2.1 Introducción

Las redes actuales rigen cada vez más la forma en que vivimos: aplicaciones de red como las redes sociales, el e-learning o el comercio electrónico, están cambiando la forma en que se llevan a cabo las interacciones sociales, educativas y comerciales. Estas aplicaciones son diversas y están en constante evolución con el fin de cubrir las necesidades de los nuevos usuarios y ofrecer nuevos servicios. Consecuentemente, las tecnologías de red subyacentes deben ser adaptadas y / o mejoradas para proporcionar los nuevos servicios.

Como resultado de la gran impacto y uso de Internet, el tráfico de red está en constante crecimiento [Brodkin, 2012, Kende, 2012] y la propia red se está volviendo cada vez más compleja. Así, el ancho de banda disponible para los usuarios finales ha crecido hasta 2 órdenes de magnitud en la última década, mientras que, al mismo tiempo, el número de usuarios está aumentando exponencialmente. Desde la perspectiva de la gestión de red, este continuo aumento de los requisitos y la complejidad representa un gran reto en muchos aspectos. Entre ellos, el incremento en el tráfico puede afectar a la calidad del servicio (QoS, del inglés Quality of Service). La influencia del tráfico en la infraestructura de la red y sobre los servicios alojados es evidente. Si no se gestiona correctamente, la red podría ser abrumada con enormes cantidades de tráfico incontrolado que afectaría el rendimiento general de la red, llegando a la congestión e incluso interrupción de los servicios. Por otro lado, al aumentar el volumen de tráfico, será más difícil diferenciar el tráfico ilegítimo que puede representar un problema de seguridad.

Para proporcionar la calidad de servicio demandada, es necesaria una adecuada gestión de la infraestructura de red subyacente. En este sentido, una de las tareas más comunes consiste en monitorizar constantemente los flujos de tráfico y establecer su naturaleza en tiempo real, lo que es relevante desde

los puntos de vista de seguridad y de rendimiento. A partir de esta identificación, será posible proporcionar a cada servicio/flujo la QoS requerida o, si es necesario, incluso proceder a su filtrado. En este sentido, la descripción del tráfico de red a partir del ancho de banda consumido, aunque importante, no es suficiente. Por lo tanto, un paso importante en cualquier análisis de tráfico es identificar lo que el usuario está realmente haciendo en cada sesión, es decir, identificar la aplicación en uso.

La identificación de las aplicaciones de red, es decir, la atribución de tráfico de red a la aplicación o tipo de aplicación que la genera [Callado, 2009], es, por tanto, la piedra angular para una adecuada ingeniería del tráfico en la red. A partir de ella será posible, por ejemplo, proporcionar diferentes QoS a los distintos servicios, deshabilitar el tráfico de una aplicación dada o planificar la mejora de la infraestructura como respuesta a un nuevo servicio de alto consumo. Por otra parte, el tráfico ilegítimo puede utilizar técnicas de ofuscación en un intento de atravesar los mecanismos de seguridad tales como los cortafuegos. En este escenario, la identificación del tráfico se hace también necesaria para aplicar las acciones correctivas.

En un enfoque ingenuo en la actualidad, el tráfico se puede atribuir a una aplicación o servicio a partir de los puertos asociados al servicio [IANA, 2013]. Históricamente, esta aproximación se ha venido usando por ser la más rápida, simple y precisa. Sin embargo, prácticas actualmente habituales como la ofuscación de puertos, la traducción de direcciones (NAT, del inglés, Network Address Translation), la redirección de puertos y los túneles, junto con el uso de puertos no registrados y aplicaciones multicanal, han invalidado esta técnica. Se necesitan, en consecuencia, aproximaciones más sofisticadas que deben implicar el análisis de múltiples aspectos y características del tráfico, como las cargas útiles, las propiedades generales de los flujos y paquetes, o incluso las interacciones entre los hosts, por mencionar algunas. A modo de ejemplo, los paquetes de gran tamaño y/o flujos de larga duración podrían ser indicadores de una transferencia de archivos de gran tamaño; un gran número de conexiones podría apuntar a la utilización de aplicaciones peer-to-peer; etc. En un sentido similar, la existencia de paquetes malformados o flujos incompletamente establecidos puede indicar anomalías en el uso de un protocolo y/o actividades sospechosas.

En consecuencia, la identificación de las aplicaciones de red no responde exclusivamente a la simple necesidad de asociar el tráfico a las aplicaciones. A pesar de que no es suficiente por sí misma para mejorar la gestión de la red y la seguridad, puede ayudar a proporcionar una visión más profunda de lo que está ocurriendo en la red, resultando relevante para muchas tareas relacionadas, en especial las relativas a la ingeniería de tráfico. Por otra parte,

un paso preliminar antes de la aplicación de cualquier política de acceso a la red es la identificación de la aplicación en uso, ya que dichas políticas a menudo implican condiciones sobre el tipo de aplicación utilizado.

En este contexto, la constante aparición de nuevas aplicaciones, junto con los avances en los mecanismos de ofuscación que se utilizan para evitar los mecanismos de control, hacen de la identificación de tráfico un tema candente de investigación.

A2.2 Clasificación del tráfico de red

Como se ha mencionado con anterioridad, la clasificación del tráfico [Callado, 2009] consiste en atribuir los diferentes elementos que constituyen el tráfico de red a las aplicaciones o tipos de aplicación que los generan. De acuerdo al tipo de elemento a clasificar se pueden diferenciar tres niveles básicos [Callado, 2009]: a nivel de flujo, a nivel de paquetes y a nivel de host. En el primer caso, los elementos a clasificar son los flujos en la red, mientras que en el segundo caso, se consideran los paquetes individuales. Por el contrario, la clasificación a nivel de host consiste en etiquetar cada host de acuerdo a la aplicación o aplicaciones en uso por el mismo. El caso más habitual es la clasificación de flujos, ya que este es el elemento más directamente relacionado con la conformación del tráfico y la QoS, a la vez que constituye la unidad natural de comunicación entre aplicaciones. Por otra parte, la clasificación a nivel de host, a pesar de ser un problema más complejo, resulta un escenario de enorme interés para la planificación de la red y cuestiones relacionadas con la seguridad, ya que permitiría la identificación de equipos que realizan actividades maliciosas.

La identificación de tráfico es un término que también se utiliza en este contexto para referirse a la clasificación de tráfico, aunque, estrictamente, se refiere a la categorización del tráfico a un mayor nivel de granularidad que en el caso de clasificación. En cualquier caso, es habitual utilizar indistintamente ambos términos, que será el caso utilizado en la presente tesis.

Por otra parte, de acuerdo con el número de clases utilizadas en la clasificación, se pueden considerar algunas variantes del problema. Así, en escenarios de clasificación binaria, el problema es decidir si una instancia a clasificar pertenece o no a una categoría determinada. Por el contrario, en la clasificación multiclase, que es la habitualmente considerada, una instancia debe ser asignada a una de las múltiples clases existentes. Sin embargo, existen escenarios en los que se requiere una clasificación multietiqueta. En estos casos, por lo general relacionados con la clasificación basada en hosts, cada elemento puede ser asignado a varias de las categorías consideradas. Este es el escenario más

complejo desde el punto de vista de la clasificación, ya que por lo general el número de clases (etiquetas) que se deben asignar a cada elemento es también desconocido.

A2.2.1 Aproximaciones básicas

Aunque se incluye un análisis detallado del estado del arte en la clasificación del tráfico en el Capítulo 2, a continuación se presenta una visión general de los enfoques de clasificación de tráfico existentes y los principales desafíos, a fin de situar el trabajo en el contexto adecuado. En resumen, los métodos descritos en la literatura analizan bytes de la carga útil [Erman, 2007a], los paquetes [Sen, 2004b], los flujos [Zhang, 2013], o los hosts [Karagiannis, 2005], siendo la clasificación basada en flujos la más usada. El alcance de las aplicaciones detectadas varía de protocolos conocidos (por ejemplo, HTTP, SMTP) a aplicaciones que entrañan mayor dificultad de clasificación, como P2P o tráfico cifrado (por ejemplo, SSL [McCarthy, 2011] y túneles de red encriptados [10]).

Por otra parte, la mayoría de los enfoques existentes en la actualidad se pueden clasificar en dos grandes grupos: los basados en el análisis de la carga útil, (DPI, del inglés Deep Packet Inspection) [Sen, 2004b] y aquellos que no inspeccionan dicha carga útil. Estos últimos se denominan ciegos (blind, en inglés). Históricamente, los métodos de clasificación se basaban en los números de puerto a nivel de la capa de transporte [IANA, 2013]. Con la ofuscación de puertos, este método resulta obviamente obsoleto. En cualquier caso, puede ser de utilidad para algunas aplicaciones concretas como DNS o SMTP [Callado, 2009, Aceto, 2010], como método auxiliar o incluso en los casos en los que la precisión en la clasificación no sea relevante.

Como consecuencia, surgen los métodos basados en la inspección de la carga útil [Sen, 2004b], siendo los más precisos que actualmente se encuentran en la literatura. Estos identifican las aplicaciones a partir de la búsqueda de cadenas o firmas conocidas dentro de las cargas útiles de los paquetes, lo que plantea, sin embargo, importantes limitaciones relacionadas con el rendimiento, el tráfico cifrado y la privacidad del usuario. Por lo tanto, existe un gran interés en el desarrollo de métodos ciegos, esto es, no basados en la inspección de la carga útil, ya que teóricamente son capaces de analizar el tráfico cifrado y no plantean problemas respecto de la privacidad del usuario. En este caso, cada unidad objeto de análisis, típicamente flujos, se representa mediante un conjunto de parámetros que se utiliza posteriormente como entrada para el método de clasificación de elección. En este sentido, son múltiples las técnicas descritas en la literatura, con distinto éxito, incluyendo técnicas estadísticas [Yildirim, 2010], basadas en aprendizaje automático [Nguyen, 2008] y/o aproximaciones gráficas

[Iliofotou, 2007], así como algunas híbridas [Keralapura, 2010, Trestian, 2010].

A2.2.2 Retos

La clasificación de tráfico constituye una tarea de enorme dificultad por múltiples motivos. En particular, algunas aplicaciones pueden utilizar protocolos complejos y dinámicos y que incluyen técnicas avanzadas de ofuscación de tráfico [Zink, 2012], como por ejemplo, la ofuscación de puertos, la traducción de direcciones de red (NAT), túneles y cifrado [McCarthy, 2011]. Adicionalmente, el cifrado hace que sea muy difícil clasificar el tráfico en base a los mecanismos de inspección de carga útil, lo que se ve agravado por la existencia de protocolos propietarios cerrados (por ejemplo, Skype [Sen, 2004b]).

Otras cuestiones pueden constituir un reto adicional para el desarrollo de técnicas de clasificación de tráfico. En particular, son relevantes la protección de la privacidad del usuario, que limita e incluso prohíbe la inspección de los contenidos de los paquetes, así como las altas velocidades de comunicación, que plantean problemas de escalabilidad.

Desde el punto de vista científico, y a pesar del enorme interés evidenciado en los últimos años en este problema, no existe ningún estudio reciente y de suficiente extensión que posibilite una comparativa adecuada de las numerosas técnicas descritas en la literatura. En este sentido, la mayoría de los clasificadores existentes carecen de generalidad, ya que a menudo son evaluados en contextos especiales (aplicaciones y/o entorno de red específicos), sobre conjuntos de datos muy diversos, y en muy pocas ocasiones se comparan los resultados obtenidos por la técnica propuesta con otras descritas en la bibliografía. Para añadir mayor complejidad al problema, se consideran diferentes niveles de granularidad en la clasificación, tanto desde el punto de vista del elemento a clasificar, como las categorías o tipos de categorías en las que deben categorizarse. Así, se aborda la clasificación de diversos elementos, desde paquetes individuales a grupos de ordenadores, pasando, obviamente, por los flujos. En este contexto, con la diversidad en las técnicas desplegadas y formatos de datos considerados, la literatura actual todavía carece de puntos de referencia adecuados, que consideren capturas de tráfico de referencia y que se expresen en una terminología consistente [Khalife, 2014].

A2.3 Objetivos y contribuciones

El principal objetivo del presente trabajo es desarrollar y evaluar nuevos métodos de clasificación ciega de tráfico, tanto a nivel de flujo como de host. Para ello, también es necesario establecer un marco experimental que incluya cap-

turas de tráfico reales, que deben ser etiquetadas de acuerdo a sus clases, y que se utilizará como referencia para entrenar y evaluar los clasificadores. Adicionalmente, es necesario desarrollar los programas y algoritmos necesarios para parametrizar el tráfico, de acuerdo con las necesidades de los métodos de clasificación propuestos.

Los sistemas explorados para la clasificación de tráfico se basan en análisis del tamaño de los mensajes, para el caso de los flujos, y en la descomposición en grafos elementales, denominados motif, del grafo de interacciones de red, para el caso de los host. Antes de presentar estas propuestas, se hace una revisión a fondo del estado de la técnica y se realiza una propuesta de taxonomía para categorizar las diferentes aproximaciones y variantes. Adicionalmente, tras el procesamiento para clasificar el tráfico adquirido utilizando DPI, se proponen y se evaluan algunas optimizaciones basadas en el muestreo. De este modo, las contribuciones de esta tesis pueden resumirse de la siguiente manera:

a) Propuesta de una taxonomía: El número de trabajos en la bibliografía reciente relacionados con la clasificación del tráfico es elevado. Sin embargo, existen pocos tutoriales y/o artículos que proporcionen una visión general del estado de la técnica y menos aun son los que clasifican sistemáticamente las contribuciones existentes. En este contexto, se propone una taxonomía en tres niveles diferentes (entrada, técnica y salida) que permite categorizar los métodos existentes, a la luz de la que se presentan las técnicas y trabajos más destacados en cada categoría. Como resultado del estudio del estado de la técnica, se analizan los principales retos existentes y las características que debería presentar un clasificador de tráfico ideal.

b) Escenario experimental: La evaluación y comparación de métodos de clasificación de tráfico requiere de la utilización de trazas de tráfico significativas. Dada la inexistencia de conjuntos de datos de referencia, se procedió a la captura de tráfico real en un volumen significativo después de analizar los requisitos que debería reunir para posibilitar una evaluación adecuada. Adicionalmente, se desarrolla un conjunto de herramientas para manejar las bases de datos y extraer todos los parámetros descritos en la literatura para su uso potencial en el proceso de clasificación. También se adapta una herramienta de clasificación basada en DPI para el etiquetado de las capturas obtenidas a fin de su utilización como técnica de referencia.

c) Evaluación y optimización de los métodos de clasificación basados en DPI: Los principales factores limitantes para el uso de la clasificación

DPI son la invasión de la privacidad del usuario y el elevado coste computacional asociado. Tras evaluar el rendimiento de la técnica DPI, se presenta una optimización basada en diversas políticas de muestreo para las cargas útiles en cada flujo. Usando esta técnica, se evalúa experimentalmente el mejor compromiso entre el rendimiento, la precisión y la privacidad de los usuarios.

d) Método de clasificación ciega basada en flujo: Los mensajes intercambiados por entidades pares en la capa de aplicación incluyen información que podría enriquecer las capacidades de un clasificador. Desde este punto de vista, se propone un nuevo método ciego para explorar el poder discriminativo de los mensajes de la capa de aplicación. La novedad de nuestro enfoque consiste en basarse en el tamaño y la dirección de los mensajes, en lugar de en la de los paquetes, y la aplicación de un modelado de Markov junto con gaussianas multipico para representar las posibles distribuciones de tamaño de los diferentes métodos en cada protocolo de la capa de aplicación. Los resultados evidencian la bondad de la propuesta y la existencia de información discriminativa respecto a la clasificación del tráfico en los tamaños de los mensajes intercambiados.

e) Evaluación y mejora de modelos ciegos de clasificación basados en host: Los modelos actuales de clasificación basados en host no reflejan escenarios de uso reales, en los que un host puede estar usando más de una aplicación en el periodo de tiempo considerado. Por lo tanto, tras seleccionar un método basado en host descrito en la bibliografía, exploramos la aplicación de una clasificación multi-etiqueta. El método elegido se basa en una descripción gráfica de las interacciones entre hosts, que se representan a partir de unas unidades elementales denominadas motifs. El método original se evalúa y, a continuación, se proponen y evalúan algunas mejoras a nivel de parametrización. Por último, se explora la extensión del método para la identificación de host multi-etiqueta.

La presente tesis se estructura en 5 capítulos adicionales a la introducción, que se resumen a continuación.

A2.4 Estado del arte en clasificación de tráfico

En primer lugar, en el capítulo 2, se analiza el estado de la técnica de una manera integral tras proponer una taxonomía, que se utiliza para describir y organizar las aportaciones existentes. Si bien en la última década varios grupos de investigación han estado trabajando en el tema de la clasificación

del tráfico, lo que ha generado una gran cantidad de publicaciones, no existía ningún estudio significativo que proporcionase una visión global del campo. La mayoría de los existentes se encontraban anticuados [Zhang, 2009, Callado, 2009], puesto que el número de las contribuciones se ha duplicado en los últimos años, incluyendo una gran cantidad de enfoques recientes y prometedores; mientras que otros se centraban en una línea de investigación específica. En este sentido, aunque cada uno de los métodos de identificación descritos en la literatura se considera óptima o relevante desde el punto de vista de sus autores, en la mayoría de los casos se consideran condiciones específicas y/o aplicaciones particulares. Además, las comparaciones existentes utilizan solamente algunas técnicas de identificación, presentando resultados raramente verificables y centrados en la reducción de las tasas de error. Al parecer, no hay tendencias claras de investigación hacia la definición del mejor modelo de identificación del tráfico.

Por lo tanto, este capítulo tiene como objetivo proporcionar un enfoque sistemático para clasificar y caracterizar los métodos de identificación del tráfico a través de una taxonomía jerárquica integral de tres niveles. De acuerdo con esta, cada método se clasifica en tres categorías diferentes respecto de: el tipo de elementos de entrada, la técnica utilizada para la clasificación y la salida proporcionada, esto es, los tipos de clasificaciones obtenidas. De esta forma, cada técnica descrita en la bibliografía se puede categorizar según cada uno de estos tres criterios, quedando caracterizada por una tríada, lo que permite determinar qué sistemas pueden ser adecuadamente comparados entre sí.

A continuación, esta taxonomía se utiliza como vehículo para presentar la literatura actual existente en la clasificación de tráfico a partir del análisis de un volumen significativo de los trabajos existentes y recientes en el campo. Estos trabajos se describen de acuerdo con la terminología y las categorías establecidas en la taxonomía propuesta. A partir de este estudio, se presentan y discuten los requisitos que debe cumplir el método óptimo para la identificación del tráfico y se destacan los desafíos existentes en la investigación en este campo.

A2.5 Escenario experimental

Uno de los principales objetivos de esta tesis, como se dijo anteriormente, es desarrollar y evaluar nuevos métodos para mejorar los sistemas de identificación de tráfico actuales. La consecución de estos objetivos requiere de algunos objetivos adicionales relacionados con el manejo y adquisición de datos de tráfico de la red que, a su vez, son esenciales para el desarrollo y la evaluación del sistema de clasificación.

En este sentido, antes de pasar a cada una de las nuevas propuestas que requieren la validación experimental, tenemos que establecer un banco de pruebas apropiado. Por lo tanto, el Capítulo 3 está dedicado a explicar la configuración experimental y el escenario de pruebas utilizado en esta tesis, incluyendo la descripción de los conjuntos de datos capturados. El capítulo comienza con una discusión sobre los requisitos para los conjuntos de datos y el escenario con el fin de ser capaz de evaluar adecuadamente los métodos que se propongan. A partir de este análisis, se concluye que, entre otras propiedades, se necesita un conjunto de datos de tráfico real y etiquetado adecuado Esto representa un reto importante, ya que no es trivial etiquetar cada uno de los elementos en el conjunto de datos reales. Por lo tanto, después de describir el escenario de red y los procedimientos de adquisición de datos, se presentan el método y las herramientas utilizadas para etiquetar cada una de las muestras. El capítulo continúa con una descripción de los conjuntos de datos capturados y las distintas particiones establecidos para entrenar, evaluar y validar cada uno de los métodos de clasificación de tráfico propuestos en esta tesis. A continuación, se describen los procedimientos de extracción de características y los parámetros obtenidos.

Por otra parte, la evaluación del desempeño de las nuevas propuestas requiere de algunas métricas que son descritas y discutidas al final del capítulo, ya que existen diferentes posibilidades.

A2.6 Optimización de las técnicas DPI mediante muestreo

Como se mencionó con anterioridad, la clasificación de tráfico mediante DPI se basa en el examen completo de la carga útil de todos los paquetes. Consecuentemente, la invasión de privacidad de las comunicaciones y el rendimiento, en términos de coste computacional, constituyen las mayores debilidades de los clasificadores DPI.

A lo largo de este capítulo, proponemos y evaluamos una metodología general para la optimización de DPI mediante muestreo. La propuesta está motivada por el potencial para reducir el coste computacional de la inspección de las cargas útiles y mejorar la privacidad de los usuarios al reducir la cantidad de bytes de carga útil analizados. A pesar de su interés, no se describe en la bibliografía ningún estudio detallado ni comparativo del impacto sobre el rendimiento de DPI de la aplicación de diversas políticas de muestreo, a pesar de la existencia de algunas propuestas al respecto [Chen, 2009b]. Por tanto, se estudian y evalúan diversas políticas de muestreo. En particular, se con-

sideran 4 aproximaciones básicas: limitar el tamaño inspeccionado de la carga útil, limitar el número de paquetes inspeccionados por flujo, la combinación de ambos y, finalmente, limitar el número de bytes inspeccionados al comienzo de un flujo, independientemente del número de paquetes necesarios.

El objetivo de la experimentación es determinar la existencia, y, en su caso, su ubicación, de secciones de la carga útil en los flujos en las que se identifican con frecuencia las firmas de aplicación utilizadas por DPI. Analizar estas partes de la carga útil debe ser, naturalmente, el objetivo de las técnicas de muestreo a aplicar.

La evaluación experimental de las optimizaciones propuestas evidencia que las firmas de aplicación suelen encontrarse en posiciones regulares al inicio de los flujos. Consecuentemente, la técnica de muestreo que mejores prestaciones proporciona es la que se ha denominado muestreo contiguo de flujo, que se basa en el análisis de los bytes al inicio de cada flujo. Mediante esta aproximación, es posible reducir significativamente los tiempos necesarios para clasificar un flujo, esto es, el coste computacional asociado, a la vez que se minimiza la invasión de la privacidad, sin apenas impacto en el número de flujos clasificados correctamente.

Durante el análisis experimental de la propuesta se evidencia que el número de bytes a analizar depende de la aplicación concreta a clasificar, siendo posible determinar un umbral mínimo para este valor a partir del conjunto de aplicaciones que se pretende clasificar y del porcentaje de clasificación correcta deseado. Evidentemente, estos umbrales deben ser determinados experimentalmente.

A2.7 Clasificación de tráfico basada en el tamaño de los mensajes

En este capítulo se propone un nuevo método para la identificación del tráfico. Después de concluir en el estudio anterior que la mayoría de las firmas utilizadas por DPI se encuentran dentro de los paquetes iniciales de los flujos, el método propuesto se centra en los mensajes iniciales intercambiados en cada flujo por las entidades de nivel de aplicación.

Como se indica en el Capítulo 2, la selección del conjunto de parámetros a utilizar es una opción estratégica para los clasificadores de tráfico, especialmente para los ciegos, ya que se basan en características extraídas generalmente de las capas de red y de transporte [Khalife, 2014, Moore, 2005a]. Lo ideal sería que las propiedades seleccionadas resulten discriminativas, inmunes a la dinámica de la red y las técnicas de ofuscación, al tiempo que protejan la

privacidad del usuario.

Así, en la técnica propuesta se analizan los tamaños y sentidos relativos de los mensajes iniciales en cada flujo. El método sugerido es ciego, ya que no se requiere de la inspección de la carga útil, ya que los tamaños de los mensajes se pueden inferir directamente a partir de las cabeceras de la capa de transporte mediante una heurística diseñada al efecto. Consecuentemente, cada flujo queda parametrizado por un vector cuyas componentes son los tamaños de los L mensajes iniciales, con un signo que depende del sentido en el que se intercambia dicho mensaje.

La clasificación de cada flujo se realiza, a partir de dicho vector, mediante un clasificador basado en modelos de Markov cuyas probabilidades de observación se obtienen a partir de distribuciones gaussianas multipico. Cada una de dichas distribuciones es obtenida experimentalmente a partir de la aplicación de un algoritmo de agrupamiento basado en k-medias iterativo a los vectores de entrenamiento en el que se usa una métrica no euclídea. Dicha métrica, propuesta y evaluada también en conjunción con el método de clasificación, tiene como finalidad evaluar el grado de similitud de dos mensajes en base a sus diferencias relativas de tamaño.

Los resultados experimentales obtenidos muestran un buen comportamiento de la técnica propuesta, tanto a nivel de clasificación como de coste computacional. Además, dada la naturaleza de la misma, constituye un método de clasificación temprana, puesto que únicamente se requiere un reducido número de paquetes al inicio de cada flujo para clasificarlo con un alto nivel de confianza.

A2.8 Clasificación basada en hosts mediante motifs

Después de proponer un método de clasificación de flujo, el capítulo 6 se centra en la clasificación basada en host. En particular, se dedica a la evaluación de un enfoque basado en la descomposición de la actividad de red observada en grafos elementales denominados motifs.

Este enfoque fue propuesto y explorado por otros autores con anterioridad [nDPI, 2013], aunque presenta algunas limitaciones. En este sentido, la revisión de la bibliografía (Capítulo 2) revela la existencia de varias contribuciones de interés basadas en la clasificación de host. Sin embargo, la mayoría de ellas siguen un modelo de clasificación de etiqueta única, asumiendo que cada host sólo presenta actividad asociada a una única aplicación. Como se puede argumentar fácilmente, este modo de operación no refleja situaciones reales, ya que un comportamiento típico implica que la actividad de red de un host incluye protocolos diferentes. En consecuencia, la clasificación debería realizarse aten-

diendo a varias etiquetas. Por lo tanto, el objetivo principal de este trabajo es extender estos métodos a un escenario de clasificación multi-etiqueta.

Para abordar las técnicas propuestas en [nDPI, 2013], se describen los fundamentos de la teoría de grafos subyacente y la extracción de los llamados motif a partir de un gráfico que representa las interacciones entre los ordenadores. A continuación, cada host se parametriza mediante un perfil que representa su participación en cada uno de los motif obtenidos.

Tras valorar y evaluar los resultados obtenidos a partir de este método en un escenario real, se identifican las limitaciones del método utilizado. En este sentido, se proponen y evalúan algunas mejoras relativas a la extracción de características y la parametrización. A continuación, con el fin de ser realmente aplicable en escenarios reales, el clasificador debe ser capaz de clasificar los hosts de acuerdo con el conjunto de aplicaciones en uso, en lugar de sólo uno de ellos.

Los resultados obtenidos muestran una mejora en la precisión de la clasificación, tanto en el escenario multi-clase como en el multi-etiqueta, aunque es evidente que el sistema debe ser mejorado aún más. En este sentido, para finalizar el capítulo se identifican las posibles actuaciones de mejora.

Acronyms

ACK	ACKnowledgment.
ANN	Artificial Neural Network.
BoF	Bag of Flows.
BP-NN	Back Propagation trained Neural Networks.
CCN	Content-Centric Networks.
DFI	Deep Flow Identification.
DNS	Domain Name System.
DPI	Deep Packet Inspection.
EM	Expectation Maximization.
FANMOD	Fast Network Motif Detection.
FF-NN	Feed-Forward Neural Networks.
FN	False Negative.
FP	False Positive.
FPR	False Positive Rate.
FTP	File Transfer Protocol.
HMM	Hidden Markov Models.
HTTP	Hyper Text Transfer Protocol.
HTTPS	Hypertext Transfer Protocol Secure.
IANA	Internet Assigned Numbers Authority.
IMAP	Internet Message Access Protocol.
IP	Internet Protocol.
IRC	Internet Relay Chat.
KNN	K-nearest Neighbors.
MAP	Maximum A posteriori Probability.
ML-KNN	Multi-label KNN.
MPG	Multi-Peak Gaussian model.
MSS	Maximum Segment Size.
MTU	Maximum Transmission Unit.

NAT	Network Address Translation.
NBAR	Network-based Application Recognition.
NTP	Network Time Protocol.
P2P	Peer-to-Peer.
P2PTV	Peer-to-Peer TV.
PCAP	Packet Capture.
PDF	Probability Density Functions.
PISA	Protocol Identification via Statistical Analysis.
PNN	Probabilistic Neural Network.
POP3	Post Office Protocol.
QoS	Quality of Service.
RL	Reinforcement Learning.
ROC	Receiver Operating Characteristic Curve.
RTP	Real-time Transport Protocol.
SCE	Cisco Service Control Engine.
SDN	Software Defined Networks.
SMTP	Simple Mail Transfer Protocol.
SOM	Self-Organizing Maps.
SPID	Statistical Protocol IDentification.
SSH	Secure Shell.
SSL	Secure Sockets Layer.
SVM	Support Vector Machine.
TAG	Traffic Activity Graphs.
TCP	Transmission Control Protocol.
TDG	Traffic Dispersion Graphs.
TN	True Negative.
TP	True Positive.
TPR	True Positive Rate.
UDP	User Datagram Protocol.
UEP	Unconstrained Endpoint Profiling.
URL	Uniform Resource Locator.
VDI	Virtualization Desktop Infrastructure.
VFDT	Very Fast Decision Trees.
VoIP	Voice Over IP.

Bibliography

[Aceto, 2010] Aceto, G.; Dainotti, A.; De Donato, W.; and Pescapé, A. (2010). PortLoad: Taking the Best of Two Worlds in Traffic Classification. In *IEEE INFOCOM Conference on Computer Communications Workshops*, pages 1–5.

[Allan, 2008] Allan, E. (2008). *Identifying Application Protocols in Computer Networks using Vertex Profiles*. PhD thesis, Wake Forest University.

[Allan, 2009] Allan, E.; Turkett, W.; and Fulp, E. (2009). Using Network Motifs to Identify Application Protocols. In *Global Telecommunications Conference (GLOBECOM)*, pages 1–7.

[Alshammari, 2011] Alshammari, R. and Zincir-Heywood, A. (2011). Can Encrypted Traffic be Identified without Port Numbers, IP Addresses and Payload Inspection? *Computer networks*, 55:1326–1350.

[Auld, 2007] Auld, T.; Moore, A.; and Gull, S. (2007). Bayesian Neural Networks for Internet Traffic Classification. *IEEE Transactions on Neural Networks*, 18:223–239.

[Ban, 2011] Ban, T.; Guo, S.; Zhang, Z.; Ando, R.; and Kadobayashi, Y. (2011). Practical Network Traffic Analysis in P2P Environment. In *7th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1801–1807.

[Bernaille, 2007] Bernaille, L. and Teixeira, R. (2007). Early recognition of encrypted applications. In *LNCS 4427 - Passive and Active Network Measurement*, pages 165–175. Springer.

[Bernaille, 2006] Bernaille, L.; Teixeira, R.; Akodkenou, I.; Soule, A.; and Salamatian, K. (2006). Traffic Classification on the Fly. In *ACM SIGCOMM Computer Communication Review*, pages 23–26.

[Bittorrent, 2013] Bittorrent (2013). Bittorrent software. `http://www.bittorrent.com`. Last accessed: 2013-04-21.

[Bo, 2009] Bo, X.; Ming, C.; and Lan, F. (2009). Distributed P2P Traffic Identification Method. In *5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCom'09)*, pages 1–4.

[Bolla, 2008] Bolla, R.; Canini, M.; Rapuzzi, R.; and Sciuto, M. (2008). Characterizing the Network Behavior of P2P Traffic. In *4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks (IT-News)*, pages 14–19.

[Bonfiglio, 2007] Bonfiglio, D.; Mellia, M.; Meo, M.; Rossi, D.; and Tofanelli, P. (2007). Revealing Skype Traffic: when Randomness Plays with You. In *ACM SIGCOMM Computer Communication Review*, pages 37–48.

[Boutell, 2004] Boutell, M. R.; Luo, J.; Shen, X.; and Brown, C. (2004). Learning Multi-label Scene Classification. *Pattern Recognition*, 37:1757–1771.

[Brodkin, 2012] Brodkin, J. (2012). The Future of Bandwitdh, Ars Technica. `http://arstechnica.com/business/2012/05/bandwidth-explosion-as-internet-use-soars-can-bottlenecks-be-averted/`. Last accessed: 2013-04-21.

[Bujlow, 2014] Bujlow, T.; Carela-Español, V.; and Barlet-Ros, P. (2014). Extended Independent Comparison of Popular Deep Packet Inspection (DPI) Tools for Traffic Classification. Technical report, Universitat Politècnica de Catalunya.

[Bujlow, 2015] Bujlow, T.; Carela-Español, V.; and Barlet-Ros, P. (2015). Independent Comparison of Popular DPI Tools for Traffic Classification. *Computer Networks*, 76:75–89.

[Callado, 2009] Callado, A.; Kamienski, C.; Szabó, G.; Gero, B.; Kelner, J.; Fernandes, S.; and Sadok, D. (2009). A Survey on Internet Traffic Identification. *IEEE Communications Surveys & Tutorials*, 11:37–52.

[Canini, 2009] Canini, M.; Fay, D.; Miller, D.; Moore, A.; and Bolla, R. (2009). Per-flow Packet Sampling for High-Speed Network Monitoring. In *First International Communication Systems and Networks and Workshops (COMSNETS)*, pages 1–10.

[Carela-Español, 2011] Carela-Español, V.; Barlet-Ros, P.; Cabellos-Aparicio, A.; and Solé-Pareta, J. (2011). Analysis of the Impact of Sampling on NetFlow Traffic Classification. *Computer Networks*, 55:1083–1099.

[Cascarano, 2011] Cascarano, N.; Ciminiera, L.; and Risso, F. (2011). Optimizing Deep Packet Inspection for High-Speed Traffic Analysis. *Journal of Network and Systems Management*, 19:7–31.

[Cascarano, 2009] Cascarano, N.; Este, A.; Gringoli, F.; Risso, F.; and Salgarelli, L. (2009). An Experimental Evaluation of the Computational Cost of a DPI Traffic Classifier. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–8.

[Chang, 2009] Chang, S. and Daniels, T. (2009). Correlation based Node Behavior Profiling for Enterprise Network Security. In *Third International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'09)*, pages 298–305.

[Chen, 2009a] Chen, H.; Hu, Z.; Ye, Z.; and Liu, W. (2009a). Research of P2P Traffic Identification based on Neural Network. In *International Symposium on Computer Network and Multimedia Technology (CNMT)*, pages 1–4.

[Chen, 2009b] Chen, H.; You, F.; Zhou, X.; and Wang, C. (2009b). The Study of DPI Identification Technology based on Sampling. In *International Conference on Information Engineering and Computer Science (ICIECS 2009)*, pages 1–4.

[Cheng, 2007] Cheng, W.; Gong, J.; and Ding, W. (2007). Identifying BT-like P2P Traffic by the Discreteness of Remote Hosts. In *32nd Conference on Local Computer Networks (LCN 2007)*, pages 237–238.

[Choi, 2012] Choi, Y.; Chung, J.; Park, B.; and Hong, J. (2012). Automated Classifier Generation for Application-level Mobile Traffic Identification. In *Network Operations and Management Symposium (NOMS)*, pages 1075–1081.

[Chunzhi, 2010] Chunzhi, W.; Wei, J.; Hong, C.; Luo, W.; and Fang, H. (2010). Research on a Method of P2P Traffic Identification based on Multi-dimension Characteristics. In *5th International Conference on Computer Science and Education (ICCSE)*, pages 1010–1013.

[Cisco, 2013] Cisco (2013). Cisco Systems. `http://www.cisco.com`. Last accessed: 2013-04-21.

[Cong, 2010] Cong, R.; Yang, J.; and Cheng, G. (2010). Research of Sampling Method Applied to Traffic Classification. In *12th International Conference on Communication Technology*, pages 112–115.

[Corpora, 2009] Corpora, D. (2009). M57-Patents. `http://digitalcorpora.org/corpora/scenarios/m57-patents-scenario`. Last accessed: 2015-08-17.

[Crotti, 2007] Crotti, M.; Dusi, M.; Gringoli, F.; and Salgarelli, L. (2007). Traffic Classification through Simple Statistical Fingerprinting. *ACM SIGCOMM Computer Communication Review*, 37:5–16.

[Dainotti, 2009] Dainotti, A.; De Donato, W.; and Pescapé, A. (2009). Tie: A Community-oriented Traffic Classification Platform. In *LNCS 5537 - Traffic Monitoring and Analysis*, pages 64–74. Springer.

[Dainotti, 2008] Dainotti, A.; De Donato, W.; Pescape, A.; and Salvo, P. (2008). Classification of Network Traffic Via Packet-level Hidden Markov Models. In *IEEE Global Telecommunication Conference (GLOBECOM)*, pages 1–5.

[Dainotti, 2012] Dainotti, A.; Pescape, A.; and Claffy, K. (2012). Issues and Future Directions in Traffic Classification. *IEEE Network*, 26:35–40.

[De Comité, 2003] De Comité, F.; Gilleron, R.; and Tommasi, M. (2003). Learning Multi-label Alternating Decision Trees from Texts and Data. In *LNCS 2734 - Machine Learning and Data Mining in Pattern Recognition*, pages 35–49. Springer.

[Dedinski, 2009] Dedinski, I.; De Meer, H.; Han, L.; Mathy, L.; Pezaros, D.; Sventek, J.; and Zhan, X. (2009). Cross-layer Peer-to-peer Traffic Identification and Optimization Based on Active Networking. In *LNCS 4388 - Active and Programmable Networks*, pages 13–27. Springer.

[Dehghani, 2010] Dehghani, F.; Movahhedinia, N.; Khayyambashi, M.; and Kianian, S. (2010). Real-time Traffic Classification Based on Statistical and Payload Content Features. In *2nd International Workshop on Intelligent Systems and Applications (ISA)*, pages 1–4.

[Dempster, 1977] Dempster, A. P.; Laird, N.; and Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, pages 1–38.

[Dietterich, 2000] Dietterich, T. (2000). Ensemble Methods in Machine Learning. In *LNCS 1857 - Multiple classifier systems*, pages 1–15. Springer.

[Elisseeff, 2001] Elisseeff, A. and Weston, J. (2001). A Kernel Method for Multi-labelled Classification. In *Advances in Neural Information Processing Systems*, pages 681–687.

[Erman, 2006] Erman, J.; Arlitt, M.; and Mahanti, A. (2006). Traffic Classification using Clustering Algorithms. In *IEEE SIGCOMM workshop on Mining Network Data*, pages 281–286.

[Erman, 2007a] Erman, J.; Mahanti, A.; Arlitt, M.; Cohen, I.; and Williamson, C. (2007a). Offline/realtime Traffic Classification using Semi-supervised Learning. *Performance Evaluation*, 64:1194–1213.

[Erman, 2007b] Erman, J.; Mahanti, A.; Arlitt, M.; and Williamson, C. (2007b). Identifying and discriminating between Web and Peer-To-Peer Traffic in the Network Core. In *16th International Conference on World Wide Web (WWW)*, pages 883–892.

[Fernandes, 2009] Fernandes, S.; Antonello, R.; Lacerda, T.; Santos, A.; Sadok, D.; and Westholm, T. (2009). Slimming down Deep Packet Inspection Systems. In *IEEE INFOCOM Workshops*, pages 1–6.

[Ficara, 2010] Ficara, D.; Antichi, G.; Di Pietro, A.; Giordano, S.; Procissi, G.; and Vitucci, F. (2010). Sampling Techniques to Accelerate Pattern Matching in Network Intrusion Detection Systems. In *International Conference on Communications (ICC)*, pages 1–5.

[Gallagher, 2010] Gallagher, B.; Iliofotou, M.; Eliassi-Rad, T.; and Faloutsos, M. (2010). Link homophily in the application layer and its usage in traffic classification. In *IEEE INFOCOM*, pages 1–5.

[Gansner, 2006] Gansner, E.; Koutsofios, E.; and North, S. (2006). Drawing Graphs with Dot. `http://www.graphviz.org`. Last accessed: 2013-04-21.

[Gartner, 2011] Gartner (2011). Gartner Report for Next Generation Intrusion Prevention Systems. `http://www.gartner.com`. Last accessed: 2013-04-21.

[Gnutella, 2013] Gnutella (2013). Gnutella software. `http://rfc-gnutella.sourceforge.net/`. Last accessed: 2013-04-21.

[Gomes, 2013] Gomes, J.; Inácio, P.; Pereira, M.; Freire, M.; and Monteiro, P. (2013). Detection and Classification of Peer-to-peer Traffic: A Survey. *ACM Computing Surveys*, 45:1–40.

[Gomes, 2008] Gomes, J.; Inácio, P. R.; Freire, M. M.; Pereira, M.; and Monteiro, P. (2008). Analysis of Peer-to-peer Traffic using a Behavioural Method based on Entropy. In *International Performance, Computing and Communications Conference (IPCCC)*, pages 201–208.

[Grochow, 2007] Grochow, J. and Kellis, M. (2007). Network Motif Discovery using Subgraph Enumeration and Symmetry-Breaking. In *LNCS 4453 - Research in Computational Molecular Biology*, pages 92–106. Springer.

[Gu, 2010] Gu, C. and Zhuang, S. (2010). A Novel P2P Traffic Classification Approach using Back Propagation Neural Network. In *12th International Conference on Communication Technology (ICCT)*, pages 52–55.

[Guo, 2008] Guo, Z. and Qiu, Z. (2008). Identification of Peer-to-peer Traffic for High Speed Networks using Packet Sampling and Application Signatures. In *9th International Conference on Signal Processing (ICSP)*, pages 2013–2019.

[Hu, 2012] Hu, L. and Zhang, L. (2012). Real-time Internet Traffic Identification based on Decision Tree. In *World Automation Congress (WAC)*, pages 1–3.

[Huang, 2012] Huang, N.; Jai, G.; Chen, C.; and Chao, H. (2012). On the Cloud-based Network Traffic Classification and Applications Identification Service. In *International Conference on Selected Topics in Mobile and Wireless Networking (iCOST)*, pages 36–41.

[Huang, 2009] Huang, S.; Chen, K.; Liu, C.; Liang, A.; and Guan, H. (2009). A Statistical-feature-based Approach to Internet Traffic Classification using Machine Learning. In *International Conference on Ultra Modern Telecommunication & Workshops (ICUMT)*, pages 1–6.

[Hullár, 2011] Hullár, B.; Laki, S.; and Gyorgy, A. (2011). Early identification of Peer-to-peer Traffic. In *International Conference on Communications (ICC)*, pages 1–6.

[Iacovazzi, 2010] Iacovazzi, A. and Baiocchi, A. (2010). Optimum Packet Length Masking. In *22nd International Teletraffic Congress (ITC)*, pages 1–8.

[Iacovazzi, 2014] Iacovazzi, A. and Baiocchi, A. (2014). Internet Traffic Privacy Enhancement with Masking: Optimization and Tradeoffs. *IEEE Transactions on Parallel and Distributed system*, 25:353–362.

[IANA, 2013] IANA (2013). IANA Port Numbers. `http://www.iana.org/assignments/port-numbers`. Last accessed: 2013-04-21.

[Ichino, 2010] Ichino, M.; Maeda, H.; Yamashita, T.; Hoshi, K.; Komatsu, N.; Takeshita, M.; Iwashita, M.; and Yoshino, H. (2010). Internet Traffic Classi-

fication using Score Level Fusion of Multiple Classifier. In *9th International Conference on Computer and Information Science (ICIS)*, pages 105–110.

[Iliofotou, 2009] Iliofotou, M.; Faloutsos, M.; and Mitzenmacher, M. (2009). Exploiting Dynamicity in Graph-based Traffic Analysis: Techniques and Applications. In *5th International Conference on Emerging Networking Experiments and Technologies*, pages 241–252.

[Iliofotou, 2011] Iliofotou, M.; Kim, H.; Faloutsos, M.; Mitzenmacher, M.; Pappu, P.; and Varghese, G. (2011). Graption: A Graph-based P2P Traffic Classification Framework for the Internet Backbone. *Computer Networks*, 55:1909–1920.

[Iliofotou, 2007] Iliofotou, M.; Pappu, P.; Faloutsos, M.; Mitzenmacher, M.; Singh, S.; and Varghese, G. (2007). Network Monitoring using Traffic Dispersion Graphs (TDGs). In *ACM SIGCOMM Conference on Internet Measurement*, pages 315–320.

[Jaber, 2009] Jaber, M. and Barakat, C. (2009). Enhancing Application Identification by Means of Sequential Testing. In *8th International Networking Conference (NETWORKING)*, pages 287–300.

[Jaber, 2011] Jaber, M.; Cascella, R.; and Barakat, C. (2011). Can we Trust the Inter-packet Time for Traffic Classification? In *International Conference on Communications (ICC)*, pages 1–5.

[Jaber, 2012] Jaber, M.; Cascella, R. G.; and Barakat, C. (2012). Using Host Profiling to Refine Statistical Application Identification. In *IEEE INFOCOM*, pages 2746–2750.

[Jacobson, 1989] Jacobson, V.; Leres, C.; and McCanne, S. (1989). The tcpdump manual page.

[Jin, 2009] Jin, Y.; Sharafuddin, E.; and Zhang, Z. (2009). Unveiling Core Network-Wide Communication Patterns through Application Traffic Activity Graph Decomposition. *ACM SIGMETRICS Performance Evaluation Review*, 37:49–60.

[Jinsong, 2009] Jinsong, W.; Weiwei, L.; Yan, Z.; Tao, L.; and Zilong, W. (2009). P2P Traffic Identification Based on NetFlow TCP Flag. In *International Conference on Future Computer and Communication*, pages 700–703.

[Jungnickel, 2013] Jungnickel, D. (2013). *Graphs, Networks and Algorithms*. Springer-Verlag.

[Juniper, 2013] Juniper (2013). Juniper Networks TechLibrary. `http://www.juniper.net/techpubs`. Last accessed: 2013-04-21.

[Jurga, 2007] Jurga, M. and Hulbój, M. (2007). Packet Sampling for Network Monitoring. Technical report.

[Karagiannis, 2004] Karagiannis, T.; Broido, A.; Faloutsos, M.; et al. (2004). Transport layer Identification of P2P Traffic. In *ACM SIGCOMM Conference on Internet Measurement*, pages 121–134.

[Karagiannis, 2005] Karagiannis, T.; Papagiannaki, K.; and Faloutsos, M. (2005). BLINC: Multilevel Traffic Classification in the Dark. In *SIGCOMM Computer Communication Review*, pages 229–240.

[Kashtan, 2004] Kashtan, N.; Itzkovitz, S.; Milo, R.; and Alon, U. (2004). Efficient Sampling Algorithm for Estimating Subgraph Concentrations and Detecting Network Motifs. *Bioinformatics*, 20:1746–1758.

[Kende, 2012] Kende, M. (2012). Internet Global Growth: Lessons for the Future. `http://www.analysysmason.com/Research/Content/Reports/Internet-global-growth-lessons-for-the-future/#10%20September%202012`. Last accessed: 2013-04-21.

[Keralapura, 2010] Keralapura, R.; Nucci, A.; and Chuah, C. (2010). A Novel Self-learning Architecture for P2P Traffic Classification in High Speed Networks. *Computer Networks*, 54:1055–1068.

[Khalife, 2013a] Khalife, J. (2013a). IMDC. `http://imdc.datcat.org/collection/1-070W-6=tcrsg-collection`. Last accessed: 2014-04-21.

[Khalife, 2011a] Khalife, J.; Hajjar, A.; and Díaz-Verdejo, J. (2011a). On the Performance of OpenDPI in Identifying P2P Truncated Flows. In *Third International Conference on Advances in P2P Systems (AP2PS)*, pages 79–84.

[Khalife, 2011b] Khalife, J.; Hajjar, A.; and Diaz-Verdejo, J. (2011b). Performance of OpenDPI to Identify Truncated Network Traffic. In *International Conference on Data Communication Networking (DCNET)*, pages 1–6.

[Khalife, 2013b] Khalife, J.; Hajjar, A.; and Díaz-Verdejo, J. (2013b). Performance of OpenDPI in identifying sampled network traffic. *Journal of Networks*, 8:71–81.

[Khalife, 2014] Khalife, J.; Hajjar, A.; and Diaz-Verdejo, J. (2014). A multilevel Taxonomy and Requirements for an Optimal Traffic-Classification Model. *International Journal of Network Management*, 24:101–120.

[Kim, 2011] Kim, J.; Shah, K.; and Bohacek, S. (2011). Detecting P2P Traffic from the P2P Flow Graph. In *7th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1795–1800.

[Kim, 2012] Kim, J.; Yoon, S.; and Kim, M. (2012). Study on Traffic Classification Taxonomy for Multilateral and Hierarchical Traffic Classification. In *14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4.

[Kim, 2006] Kim, Y.; Jo, J.; and Suh, K. (2006). Baseline profile Stability for Network Anomaly Detection. In *Third International Conference on Information Technology: New Generations (ITNG)*, pages 720–725.

[Klaus, 2009] Klaus, M. and Schulze, H. (2009). Deep Packet Inspection Technology, Applications & Net Neutrality. Technical report, ipoque.com.

[Knuth, 1998] Knuth, D. E. (1998). *The art of computer programming: sorting and searching*. Pearson Education.

[Kochut, 2009] Kochut, A. (2009). Power and Performance Modeling of Virtualized Desktop Systems. In *International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS'09)*, pages 1–10.

[Kolaczyk, 2009a] Kolaczyk, E. (2009a). *Handbook of Peer-to-peer Networking*. Springer-Verlag.

[Kolaczyk, 2009b] Kolaczyk, E. (2009b). *Statistical Analysis of Network Data*. Springer-Verlag.

[Kurt, 2012] Kurt, B.; Cemgil, A.; Mungan, M.; Polat, N.; Ozdogan, A.; and Saygun, E. (2012). Network Management without Payload Inspection: Application Classification Via Statistical Analysis of Bulk Flow Data. In *Future Network & Mobile Summit (FutureNetw)*, pages 1–8.

[L7-filter, 2013] L7-filter (2013). L7-filter software. `http://l7-filter.clearfoundation.com`. Last accessed: 2013-04-21.

[La Mantia, 2010] La Mantia, G.; Rossi, D.; Finamore, A.; Mellia, M.; and Meo, M. (2010). Stochastic Packet Inspection for TCP Traffic. In *International Conference on Communications (ICC)*, pages 1–6.

[Lee, 2007] Lee, D. and Brownlee, N. (2007). A Methodology for Finding Significant Network Hosts. In *32nd Conference on Local Computer Networks (LCN 2007)*, pages 981–988.

[Lee, 2011] Lee, M.; Hajjat, M.; Kompella, R.; and Rao, S. (2011). RelSamp: Preserving Application Structure in Sampled Flow Measurements. In *IEEE INFOCOM*, pages 2354–2362.

[Li, 2007] Li, J.; Zhang, S.; Shidong, L.; and Ye, X. (2007). P2P Traffic Identification Technique. In *International Conference on Computational Intelligence and Security*, pages 37–41.

[Li, 2008] Li, S.; Lu, Y.; and Yan, J. (2008). Real-time P2P Traffic Identification. In *IEEE Global Telecommunication Conference (GLOBECOM)*, pages 74–2478.

[Lian, 2010] Lian, W.; Monrose, F.; and McHugh, J. (2010). Traffic Classification using Visual Motifs: an Empirical Evaluation. In *Seventh International Symposium on Visualization for Cyber Security*, pages 70–78.

[Lin, 2008] Lin, P.; Lin, Y.; Lee, T.; and Lai, Y. (2008). Using String Matching for Deep Packet Inspection. *IEEE Computer*, 4:23–28.

[Liu, 2010] Liu, F.; Li, Z.; Hu, Z.; Zhou, L.; Liu, B.; and Yu, J. (2010). Weight Based Multiple Support Vector Machine Identification of Peer-to-Peer Traffic. *Journal of Networks*, 5:577–585.

[Lu, 2014] Lu, W. and Xue, L. (2014). A Heuristic-Based Co-clustering Algorithm for the Internet Traffic Classification. In *28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 49–54.

[Lu, 2007] Lu, X.; Duan, H.; and Li, X. (2007). Identification of P2P Traffic Based on the Content Redistribution Characteristic. In *International Symposium on Communications and Information Technologies (ISCIT'07)*, pages 596–601.

[Maglogiannis, 2007] Maglogiannis, I. (2007). *Emerging Artificial Intelligence Applications in Computer Engineering: Real World AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. Springer Science & Business Media.

[McCarthy, 2011] McCarthy, C. and Zincir-Heywood, A. (2011). An Investigation on Identifying SSL Traffic. In *Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 115–122.

[McGregor, 2004] McGregor, A.; Hall, M.; Lorier, P.; and Brunskill, J. (2004). Flow Clustering using Machine Learning Techniques. In *LNCS 3015 - Passive and Active Network Measurement*, pages 205–214. Springer.

[Mfinder, 2013] Mfinder (2013). Network Motif Detection tool, Mfinder tool guide. `http://www.weizmann.ac.il`. Last accessed: 2013-04-21.

[Milo, 2002] Milo, R.; Shen-Orr, S.; Itzkovitz, S.; Kashtan, N.; Chklovskii, D.; and Alon, U. (2002). Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298:824–827.

[Miruta, 2012] Miruta, R.; Stanuica, C.; and Borcoci, E. (2012). Content Aware Classification Method. In *9th International Conference on Communications (COMM)*, pages 241–244.

[Mochalski, 2009] Mochalski, K. and Schulze, H. (2009). Mobile Systems IV. Technical report, IPOQUE.

[Moore, 2005a] Moore, A.; Zuev, D.; and Crogan, M. (2005a). Discriminators for use in Flow-based Classification. Technical report, Queen Mary University of London.

[Moore, 2005b] Moore, A. W. and Zuev, D. (2005b). Internet Traffic Classification using Bayesian Analysis Techniques. In *International Conference on Measurement and Modelling of Computer systems (SIGMETRICS)*, pages 50–60.

[Mujtaba, 2009] Mujtaba, G. and Parish, D. (2009). Detection of Applications within Encrypted Tunnels using Packet Size Distributions. In *International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 1–6.

[nDPI, 2013] nDPI (2013). nDPI software. `http://www.ntop.org`. Last accessed: 2013-04-21.

[Nguyen, 2008] Nguyen, T. and Armitage, G. (2008). A Survey of Techniques for Internet Traffic Classification using Machine Learning. *Communications Surveys & Tutorials*, 10:56–76.

[Park, 2011] Park, B.; Hong, J.; and Won, Y. (2011). Toward Fine-grained Traffic Classification. *IEEE Communications Magazine*, 49:104–111.

[Pironti, 2014] Pironti, A. (2014). Monitoring Message Size to Break Privacy, Current Issues and Proposed Solutions. In *W3C/IAB workshop on Strengthening the Internet Against Pervasive Monitoring (STRINT)*, pages 1–5.

[Priami, 2005] Priami, C. and Zelikovsky, A. (2005). *Transactions on Computational Systems Biology III*. Springer-Verlag.

[CAIDA, 2013] CAIDA (2013). Center for Applied Internet Data Analysis Datasets. `http://www.caida.org/data/overview/`. Last accessed: 2013-04-21.

[CRAWAD, 2013] CRAWAD (2013). Community Resource for Archiving Wireless Data At Dartmouth. `http://crawdad.cs.dartmouth.edu/`. Last accessed: 2013-04-21.

[Provost, 2001] Provost, F. and Fawcett, T. (2001). Robust Classification for Imprecise Environments. *Machine Learning*, 42:203–231.

[Qiang, 2011] Qiang, W. and Zhongli, Z. (2011). Reinforcement Learning Model, Algorithms and its Application. In *International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pages 1143–1146.

[Raahemi, 2008] Raahemi, B.; Zhong, W.; and Liu, J. (2008). Peer-to-peer Traffic Identification by Mining IP Layer Data Streams using Concept-adapting very Fast Decision Tree. In *20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'08)*, pages 525–532.

[Rao, 2010] Rao, A. and Udupa, P. (2010). A Hardware Accelerated System for Deep Packet Inspection. In *8th International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 89–92.

[Rasmussen, 2006] Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

[Rizzi, 2013] Rizzi, A.; Colabrese, S.; and Baiocchi, A. (2013). Low Complexity, High Performance Neuro-Fuzzy System for Internet Traffic Flows Early Classification. In *9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 77–82.

[Schulze, 2009] Schulze, H. and Mochalski, K. (2009). Ipoque Internet Study 2008/2009. `http://ipoque.com`. Last accessed: 2014-04-21.

[Sen, 2004a] Sen, S.; Spatscheck, O.; and Wang, D. (2004a). Accurate, Scalable in-network Identification of P2P Traffic using Application Signatures. In *13th International Conference on World Wide Web*, pages 512–521.

[Sen, 2004b] Sen, S.; Spatscheck, O.; and Wang, D. (2004b). Accurate, Scalable In-network Identification of P2P Traffic using Application Signatures. In *13th International Conference on World Wide Web*, pages 512–521.

[Shen, 2010] Shen, X. S.; Yu, H.; Buford, J.; and Akon, M. (2010). *Handbook of Peer-to-peer Networking*. Springer Science & Business Media.

[Snort, 2013] Snort (2013). Snort Software. `http://www.snort.org`. Last accessed: 2013-04-21.

[Sourcefire, 2013] Sourcefire (2013). Sourcefire Network Security Solutions. `http://www.sourcefire.com/`. Last accessed: 2013-04-21.

[Spyromitros, 2008] Spyromitros, E.; Tsoumakas, G.; and Vlahavas, I. (2008). An empirical Study of Lazy Multilabel Classification Algorithms. In *LNCS 5138 - Artificial Intelligence: Theories, Models and Applications*, pages 401–406. Springer-Verlag.

[Sun, 2010a] Sun, M.; Zhang, Y.; Chen, J.; and Shi, T. (2010a). A P2P Traffic Identification Method Based on VFDT. In *International Conference on Intelligent Computation Technology and Automation (ICICTA)*, pages 281–284.

[Sun, 2010b] Sun, R.; Yang, B.; Peng, L.; Chen, Y.; Zhang, L.; and Jing, S. (2010b). Traffic Classification using Probabilistic Neural Networks. In *Sixth International Conference on Natural Computation (ICNC)*, pages 1914–1919.

[Szabó, 2008] Szabó, G.; Orincsay, D.; Malomsoky, S.; and Szabó, I. (2008). On the Validation of Traffic Classification Algorithms. In *LNCS 4979 - Passive and Active Network Measurement*, pages 72–81. Springer.

[Szabo, 2007] Szabo, G.; Szabó, I.; and Orincsay, D. (2007). Accurate Traffic Classification. In *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–8.

[Tabatabaei, 2012] Tabatabaei, T. S.; Karray, F.; and Kamel, M. (2012). Early Internet Traffic Recognition based on Machine Learning Methods. In *25th Canadian Conference on Electrical & Computer Engineering (CCECE)*, pages 1–5.

[Theodoridis, 2009] Theodoridis, S. and Koutroumbas, K. (2009). *Pattern Recognition (4th edition)*. Academic Press.

[Ting, 2010] Ting, H.; Yong, W.; and Xiaoling, T. (2010). Network Traffic Classification Based on Kernel Self-Organizing Maps. In *International Conference on Intelligent Computing and Integrated Systems (ICISS)*, pages 310–314.

[Trestian, 2010] Trestian, I.; Ranjan, S.; Kuzmanovic, A.; and Nucci, A. (2010). Googling the Internet: Profiling Internet Endpoints Via the World Wide Web. *IEEE/ACM Transactions on Networking*, 18:666–679.

[Tsoumakas, 2006] Tsoumakas, G. and Katakis, I. (2006). *Multi-label Classification: An Overview*. Department of Informatics, Aristotle University of Thessaloniki, Greece.

[Tstat, 2013] Tstat (2013). Tstat software. `http://www.tstat.polito.it`. Last accessed: 2013-04-21.

[Ullah, 2012] Ullah, I.; Doyen, G.; Bonnet, G.; and Gaiti, D. (2012). A Survey and Synthesis of User Behavior Measurements in P2P Streaming Systems. *IEEE Communications Surveys & Tutorials*, 14:734–749.

[Vens, 2008] Vens, C.; Struyf, J.; Schietgat, L.; Džeroski, S.; and Blockeel, H. (2008). Decision Trees for Hierarchical Multi-label Classification. *Machine Learning*, 73:185–214.

[Verbeek, 2003] Verbeek, J.; Vlassis, N.; and Krose, B. (2003). Efficient Greedy Learning of Gaussian Mixture Models. *Neural Computation*, 15:469–485.

[Verma, 2009] Verma, D. C. (2009). *Principles of Computer Systems and Network Management*. Springer.

[Verticale, 2008] Verticale, G. and Giacomazzi, P. (2008). Performance Evaluation of a Machine Learning Algorithm for Early Application Identification. In *International Multiconference on Computer Science and Information Technology (IMCSIT)*, pages 845–849.

[Waizumi, 2011] Waizumi, Y.; Tsukabe, Y.; Tsunoda, H.; Nemoto, Y.; and Tanaka, K. (2011). Network Application Identification Based on Communication Characteristics of Application Messages. *Journal of Communication and Computer*, 8:111–119.

[Wang, 2009a] Wang, C.; Zhou, X.; You, F.; and Chen, H. (2009a). Design of P2P Traffic Identification based on DPI and DFI. In *International Symposium on Computer Network and Multimedia Technology (CNMT)*, pages 1–4.

[Wang, 2007] Wang, J.; Yan, Z.; Qing, W.; and GongYi, W. (2007). Connection Pattern-Based P2P Application Identification Characteristic. In *IFIP International Conference on Network and Parallel Computing Workshops*, pages 437–441.

[Wang, 2009b] Wang, P.; Guan, X.; and Qin, T. (2009b). P2P Traffic Identification based on the Signatures of key Packets. In *14th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–5.

[Wang, 2010] Wang, X. and Parish, D. (2010). Optimised multi-stage TCP Traffic Classifier based on Packet Size Distributions. In *Third International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ)*, pages 98–103.

[Wang, 2012] Wang, Y.; Xiang, Y.; Zhang, J.; and Yu, S. (2012). Internet Traffic Clustering with Constraints. In *8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 619–624.

[WEKA, 2013] WEKA (2013). A Collection of Machine Learning Algorithms for Data Mining. `http://www.cs.waikato.ac.nz/ml/weka/`. Last accessed: 2013-04-21.

[Wernicke, 2006a] Wernicke, S. (2006a). Efficient Detection of Network Motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3:347–359.

[Wernicke, 2006b] Wernicke, S. and Rasche, F. (2006b). Efficient Detection of Network Motifs. *Journal of Bioinformatics*, pages 1152–1153.

[Wright, 2006] Wright, C. V.; Monrose, F.; and Masson, G. (2006). On Inferring Application Protocol Behaviors in Encrypted Network Traffic. *Journal of Machine Learning Research*, 7:2745–2769.

[Wu, 2009] Wu, H.; Huang, N.; and Lin, G. (2009). Identifying the Use of Data/Voice/Video-Based P2P Traffic by DNS-query Behavior. In *International Conference on Communications (ICC'09)*, pages 1–5.

[Wu, 2012] Wu, X.; Li, W.; Liu, F.; and Yu, H. (2012). Packet Size Distribution of Typical Internet Applications. In *International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP)*, pages 276–281.

[Xu, 2009] Xu, B.; Chen, M.; and Hu, C. (2009). DEAPFI: A Distributed Extensible Architecture for P2P Flows Identification. In *International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pages 59–64.

[Yang, 2011] Yang, H. and Fong, S. (2011). Optimized Very Fast Decision Tree with Balanced Classification Accuracy and Compact Tree Size. In *3rd International Conference on Data Mining and Intelligent Information Technology Applications (ICMiA)*, pages 57–64.

[Yang, 2007] Yang, Y.; Wang, R.; Liu, Y.; and Zhou, X. (2007). Solving P2P Traffic Identification Problems Via Optimized Support Vector Machines. In *International Conference on Computer Systems and Applications (AICCSA'07)*, pages 165–171.

[Yang, 2012] Yang, Z.; Li, L.; Ji, Q.; and Zhu, Y. (2012). Cocktail Method for BitTorrent Traffic Identification in Real Time. *Journal of Computers*, 7:85–95.

[Yeganeh, 2012] Yeganeh, S.; Eftekhar, M.; Ganjali, Y.; Keralapura, R.; and Nucci, A. (2012). CUTE: Traffic Classification using Terms. In *21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9.

[Yildirim, 2010] Yildirim, T. and Radcliffe, P. (2010). A Framework for Tunneled Traffic Analysis. In *12th International Conference on Advanced Communication Technology (ICACT)*, pages 1029–1034.

[Yuan, 2008] Yuan, J.; Li, Z.; and Yuan, R. (2008). Information Entropy Based Clustering Method for Unsupervised Internet Traffic Classification. In *International Conference on Communications (ICC'08)*, pages 1588–1592.

[Zander, 2005] Zander, S.; Nguyen, T.; and Armitage, G. (2005). Automated Traffic Classification and Application Identification using Machine Learning. In *Conference on Local Computer Networks (LCN)*, pages 250–257.

[Zhang, 2012] Zhang, J.; Chen, C.; Xiang, Y.; and Zhou, W. (2012). Semi-supervised and Compound Classification of Network Traffic. *International Journal of Security and Networks*, 7:252–261.

[Zhang, 2013] Zhang, J.; Xiang, Y.; Wang, Y.; Zhou, W.; Xiang, Y.; and Guan, Y. (2013). Network Traffic Classification using Correlation Information. *Transactions on Parallel and Distributed Systems*, 24:104–117.

[Zhang, 2010] Zhang, L.; Li, D.; Shi, J.; and Wang, J. (2010). P2P-based Weighted Behavioral Characteristics of Deep Packet Inspection Algorithm. In *International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE)*, pages 468–470.

[Zhang, 2009] Zhang, M.; Wolfgang, J.; Claffy, K.; and Brownlee, N. (2009). State of the Art in Traffic Classification: A Research Review. In *10th International Conference on Passive and Active Measurement (PAM09)*, pages 3–4.

[Zhang, 2006] Zhang, M. and Zhou, Z. (2006). Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18:1338–1351.

[Zhang, 2007] Zhang, M. and Zhou, Z. (2007). ML-KNN: A Lazy Learning Approach to Multi-label Learning. *Pattern Recognition*, 40:2038–2048.

[Zhang, 2011] Zhang, W. and Wang, H. (2011). Identification of Peer-to-peer Traffic based on Process Fingerprint. In *International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pages 1559–1562.

[Zhang, 2014] Zhang, X.; Shao, S.; Stanley, H.; and Havlin, S. (2014). Dynamic Motifs in Socio-economic Networks. *Europhysics Letters*, 108:1–6.

[Zhenxiang, 2011] Zhenxiang, L.; Mingbo, H.; Song, L.; and Xin, W. (2011). Research of P2P Traffic Comprehensive Identification Method. In *International Conference on Network Computing and Information Security (NCIS)*, pages 307–310.

[Zhou, 2011] Zhou, W.; Dong, L.; Bic, L.; Zhou, M.; and Chen, L. (2011). Internet Traffic Classification using Feed-forward Neural Network. In *International Conference on Computational Problem-Solving (ICCP)*, pages 641–646.

[Zhou, 2012] Zhou, Z.; Song, T.; and Fu, W. (2012). RocketTC: a High Throughput Traffic Classification Architecture. In *International Conference on Computing, Networking and Communications (ICNC)*, pages 407–411.

[Zhu, 2010] Zhu, K.; Hu, H.; and Yi, P. (2010). Identifying P2P Flow with Behavior Characteristics. In *2nd International Conference on Future Computer and Communication (ICFCC)*, pages 1–140.

[Zink, 2012] Zink, T. and Waldvogel, M. (2012). Bittorrent Traffic Obfuscation: A Chase towards Semantic Traffic Identification. In *12th International Conference on Peer-to-Peer Computing (P2P)*, pages 126–137.