Contributions on detection and classification of internet traffic anomalies

Silvia Farraposo

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *L'Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Informatique*

Présentée et soutenue par *Sílvia dos Santos Farraposo*
Le *17 Juin 2009*

Titre : *Contributions on Detection and Classification of Internet Traffic Anomalies*

JURY
Fernando Pedro Lopes Boavida Fernandes, Président
Teresa Vazão Vasques, Rapporteur
Patrice Abry, Rapporteur
Mário Marques Ferreira, Examinateur
Abdelmalek Benzekri, Examinateur
Philippe Owezarski, Directeur de thèse
Edmundo Heitor da Silva Monteiro, Directeur de thèse

Ecole doctorale : *École Doctorale Système*
Unité de recherche : *CNRS-LAAS*
Directeur(s) de Thèse : *Philippe Owezarski, Edmundo Heitor da Silva Monteiro*

Acknowledgements

I would like to express my deepest gratitude to Dr. Philippe Owezarski and Dr. Edmundo Monteiro for their advice and support throughout this research. I am greatly indebted to my advisors for the invaluable guidance and encouragement which they have provided me through my study. They were always open to new ideas and I really appreciated the freedom they gave me while working on my research. This research could not have been possible without their brilliant ideas and support. I owe much for their unending help to do the research and for their financial support during my graduate study.

I also would like to express my sincere gratitude to the institutional support that I had received from several entities:

- Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, that gave me the time required to accomplish this work.
- Fundação para a Ciência e Tecnologia for the PhD bourse POS_C – Desenvolver Competência – Medida 1.2 SFRH/BD/13609/2003 that allowed me to go and stay in France.
- The European Community 6th Framework Programmes: E-NEXT – SATIN EDRF (FP6 – 506869) and CONTENT (FP6 – 0384239) for the international meetings that they offered.

Most of all, I am grateful to my family for their encouragement and patience over the past four years. I also thank my friends, particularly those that I have made at LAAS in “pièce A.43”, for giving me smiles and coffee breaks, and my stethoscope’s fellows!

Résumé

Il est évident aujourd'hui que le trafic Internet est bien plus complexe et irrégulier qu'escompté, ce qui nuit grandement à un fonctionnement efficace des réseaux, ainsi qu'à la garantie de niveaux de performances et de qualité de service (QdS) satisfaisants. En particulier, le comportement du réseau est surtout mis à mal lorsque le trafic contient des anomalies importantes. Différentes raisons peuvent être à la source de ces anomalies, comme les attaques de déni de service (DoS), les foules subites ou les opérations de maintenance ou de gestion des réseaux. De fait, la détection des anomalies dans les réseaux et leurs trafics est devenue un des sujets de recherche les plus chauds du moment.

L'objectif de cette thèse a donc été de développer de nouvelles méthodes originales pour détecter, classifier et identifier les anomalies du trafic. La méthode proposée repose notamment sur la recherche de déviations significatives dans les statistiques du trafic par rapport à un trafic normal. La thèse a ainsi conduit à la conception et au développement de l'algorithme NADA : Network Anomaly Detection Algorithm. L'originalité de NADA – et qui garantit son efficacité – repose sur l'analyse du trafic selon 3 axes conjointement : une analyse multi-critères (octets, paquets, flux, …), multi-échelles et selon plusieurs niveaux d'agrégations. A la suite, la classification repose sur la définition de signatures pour les anomalies de trafic. L'utilisation des 3 axes d'analyse permettent de détecter les anomalies indépendamment des paramètres de trafic affectés (analyse multi-critères), leurs durées (analyse multi-échelles), et leurs intensités (analyse multi-niveaux d'agrégation). Les mécanismes de détection et de classification d'anomalies proposés dans cette thèse peuvent ainsi être utilisés dans différents domaines de l'ingénierie et des opérations réseaux comme la sécurité des réseaux, l'ingénierie du trafic ou des réseaux superposés, pour citer quelques exemples.

Une contribution importante de la thèse a trait à la méthode de validation et d'évaluation utilisée pour NADA. NADA a ainsi été validé sur une base de trace de trafic contenant des anomalies documentées, puis évalué sur les principales traces de trafic disponibles. Les résultats obtenus sont de très bonne facture, notamment lorsqu'ils sont comparés avec ceux

obtenus par d'autres outils de détection d'anomalies. De plus, la qualité des résultats est indépendante du type de trafic analysé et du type d'anomalie. Il a été en particulier montré que NADA était capable de détecter et classifier efficacement les anomalies de faible intensité, qui sont apparues comme des composantes essentielles des attaques DOS. NADA apporte donc une contribution intéressante en matière de sécurité réseau.

Table of Contents

Acknowledgements **3**
Résumé **5**
Table of Contents **7**
List of Figures **9**
List of Tables **13**
List of Acronyms **15**
Chapter 1 Introduction **19**
1.1. Motivation 19
1.2. Objectives and Contributions 21
1.3. Structure of the Thesis 22
Chapter 2 From Regular to Irregular Traffic **25**
2.1. Regular Traffic 25
2.1.1. Self-Similarity Concept 27
2.1.2. Long Range Dependence 32
2.1.3. Self-Similarity, LRD and Networking 35
2.2. Irregular Traffic 37
2.2.1. Network Anomalies 38
2.2.2. Detection and Classification of Anomalies 38
2.3. Intrusion Detection Systems 43
2.3.1. IDS Classification: Scope of Protection 44
2.3.2. IDS Classification: Detection Method 45
2.4. Conclusion 48
Chapter 3 Network Anomaly Detection Algorithm – NADA **51**
3.1. NADA Fundaments 51
3.2. Algorithm Details 53
3.2.1. Multi-Criteria 53
3.2.2. Multi-Scale 54
3.2.3. Multi-Aggregation 55
3.3. Implementation 56
3.3.1. Detection Stage 56
3.3.2. Classification Stage 59

3.3.3. Identification Stage 61
3.4. Signatures for Traffic Anomalies 63
3.4.1. Flash Crowd 64
3.4.2. Distributed Denial of Service Attacks 65
3.4.3. Network Scanning 67
3.4.4. Port Scanning 68
3.5. Conclusion 69
Chapter 4 Experimental Results – Validation of Classification Mechanism 71
4.1. Repository of Traffic Traces 72
4.1.1. Documented Traces 72
4.1.2. Non-Documented Traces 76
4.2. Anomaly Signatures 78
4.2.1. Flash Crowd 79
4.2.2. Distributed Denial of Service Attacks 83
4.2.3. Network Scanning 89
4.2.4. Port Scanning 92
4.2.5. Multiple Anomalies 95
4.3. Conclusion 96
Chapter 5 Statistical Evaluation of NADA 99
5.1. Classical Evaluation of an IDS 99
5.2. Performance of NADA 101
5.2.1. ROC Curves 102
5.3. Comparison of NADA with other IDS 106
5.3.1. Gamma-FARIMA Approach 107
5.3.2. Packet Header Anomaly Detection 108
5.3.3. Results 109
5.4. Conclusion 113
Chapter 6 Conclusion 115
6.1. Main Contributions 115
6.2. Application of NADA 116
6.3. Future Work 118
Bibliography 121
Annex A Development of NADA Tool 131
A.1. Implementation 132
A.1.1. Input Data 132
A.1.2. Developed Functions 132
A.2. Utilization 142
A.2.1. Offline Execution 142
A.2.2. Online Execution 143
Abstract in English 145

List of Figures

Figure 2.1 Different traffic models commonly used for Internet traffic: (a) "almost" Constant Bit Rate traffic, (b) Poisson traffic and, (c) Real Internet traffic 26

Figure 2.2 2-dimensional Cantor set 27

Figure 2.3 Stochastic self-similarity across different time scales: 100s, 10s, 1s, 100ms 28

Figure 2.4 Wavelet analysis of traffic Internet. At each iteration, the wavelet function uses a new frequency f_i, and looks for an identical pattern in data time series 31

Figure 2.5 Representation of an LRD process. (a) Representation at the time domain. (b) Representation at the frequency domain 33

Figure 2.6 Usage of the LDEstimate tool to determine the Hurst parameter. Exhibition of the bi-scaling effect due to the existence of elephant and mouse flows in Internet traffic 34

Figure 3.1 Representation of the IP address space decomposition, used during the tomography process 57

Figure 3.2 Pseudo-Code of NADA's detection stage 58

Figure 3.3 Example of a file, created by NADA, containing the network traffic anomalies detected during a time slot 62

Figure 3.4 Signature for a Flash Crowd anomaly 65

Figure 3.5 Signature for a DDoS anomaly of type: *n* IP source addresses x 1 IP source port → 1 IP destination address x *n* IP destination ports 66

Figure 3.6 Signature for a DDoS anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x 1 IP destination port 66

Figure 3.7 Signature for a DDoS anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports 66

Figure 3.8 Signature for a Network Scanning anomaly of type: 1 IP source address x 1 IP source port → *n* IP destination addresses x 1 IP destination port 67
Figure 3.9 Signature for a Network Scanning anomaly of type: 1 IP source address x *n* IP source ports → 1 IP destination address x 1 IP destination port 67
Figure 3.10 Signature for a Network Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → *n* IP destination addresses x 1 IP destination port 68
Figure 3.11 Signature for a Port Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports 69
Figure 4.1 Representation of the MetroSec attacking framework 73
Figure 4.2 Detection of Flash Crowd anomalies using higher levels of network aggregation 80
Figure 4.3 Detection of Flash Crowd anomalies using lower levels of network aggregation 81
Figure 4.4 Impact of lowering the time granularity and the level of network flow aggregation in Flash Crowd detection and classification 82
Figure 4.5 Example of a high intensity DDoS attack. High intensity attacks can be recognized just by looking at the number of packets sent 83
Figure 4.6 Low intensity DDoS attack of type: *n* IP source addresses x 1 IP source port → 1 IP destination address x *n* IP destination ports 85
Figure 4.7 Zoom on the low intensity DDoS attack of type: *n* IP source addresses x 1 IP source port → 1 IP destination address x *n* IP destination ports. Usage of straight a flow 86
Figure 4.8 Low intensity DDoS attack of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x 1 IP destination port 87
Figure 4.9 Low intensity DDoS attack of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports 88
Figure 4.10 Identification of a low intensity DDoS attack of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports, counting the number of packets sent to a target destination 89
Figure 4.11 Network Scanning anomaly of type: 1 IP source address x 1 IP source port → *n* IP destination addresses x 1 IP destination port 90
Figure 4.12 Network Scanning anomaly of type: 1 IP source address x *n* IP source ports → 1 IP destination address x 1 IP destination port 91

Figure 4.13 Network Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → *n* IP destination addresses x 1 IP destination port 92

Figure 4.14 Port Scanning anomaly. The high number of destination port numbers used during the probing period is signalled by the vertical plane. Particularly in this example, several sources are being used 92

Figure 4.15 Port Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports 93

Figure 4.16 Zoom in the Port Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports 94

Figure 4.17 Port Scanning anomaly of type: 1 IP source address x *n* IP source ports → 1 IP destination address x *n* IP destination ports. DDoS attack of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports 96

Figure 5.1 Statistical performance of NADA using the MetroSec dataset. Probability of Detection (P_D) vs. Probability of False Alarm (P_F), $P_D = f(P_F)$ 103

Figure 5.2 Probability of Detection (P_D) and Probability of False Alarm (P_F) vs. threshold parameter *k*, $P_D = f(k)$ and $P_F = g(k)$ 104

Figure 5.3 Statistical performance of NADA using the KDD'99 dataset: Probability of Detection (P_D) vs. Probability of False Alarm (P_F), $P_D = f(P_F)$ 106

Figure 5.4 Statistical performance of PHAD using the MetroSec dataset: Probability of Detection (P_D) vs. Probability False Alarm (P_F), $P_D = f(P_F)$ 110

Figure 5.5 Probability of Detection (P_D) and Probability False Alarm (P_F) vs. threshold parameter *k*, $P_D = f(k)$ and $P_F = g(k)$ 111

Figure 5.6 Statistical performance of Gamma-FARIMA using the MetroSec dataset: Probability of Detection (P_D) vs. Probability of False Alarm (P_F), $P_D = f(P_F)$ 111

Figure 5.7 Probability of Detection (P_D) and False Alarm probability (P_F) vs. threshold parameter *k*, $P_D = f(k)$ and $P_F = g(k)$ 112

Figure A.1 Example of a visual signature of a Network Scan of type *n* IP sources – 1 IP destination 140

Figure A.2 Example of content of file Report.txt 141

List of Tables

Table 3.1 Features considered by NADA when defining an anomaly signature 59

Table 3.2 Generic set of flags used by NADA to identify traffic.anomalies and how they affect each axis..... 60

Table 3.3 Classification of traffic anomalies by NADA..... 61

Table 3.4 Parameters used in anomaly identification..... 62

Table 4.1 The MetroSec repository of traffic traces: summary of anomalies.. 74

Table 4.2 Distribution of network traffic anomalies at the aim of the KDD'99 project 75

Table 5.1 Basic characteristics of the KDD'99 datasets in terms of the number of samples..... 105

Table A.1 Input data formats recognized by NADA..... 132

Table A.2 Input parameters of function *scanSlot*() 133

Table A.3 Input parameters of function *scanWindow*() 134

Table A.4 Input parameters of functions *scanTrace*() and *scanTraceOscar*()... 135

Table A.5 Input parameters of family functions *2Dplot_*() 138

Table A.6 Input parameters for function *classificationModule*() 141

Table A.7 Initialization parameters to run the scripts *NADA_OSCAR.bash* and *NADA_EUQoS.bash*..... 142

Table A.8 Input arguments of application *nada_oscar*() 144

Table A.9 Input arguments of application *nada_euqos*() 144

List of Acronyms

ARIMA	Auto-Regressive Integrated Moving Average
ARMA	Auto-Regressive Moving Average
BGP	Border Gateway Protocol
CBR	Constant Bit Rate
CIDS	Collaborative Intrusion Detection System
CONTENT	Content Networks and Services for Home Users
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
EDRF	European Doctoral Research Foundation
EMERALD	Event Monitoring Enabling Responses to Anomalous Live Disturbances
ENS	Ecole Normale Supérieure
ENST	Ecole Normale Supérieure en Télécommunications
ERF	Extensible Record Format
EUQoS	European Quality of Service
FARIMA	Fractionally Auto-Regressive Integrated Moving Average
FGN	Fractional Gaussian Noise
H	Hurst
HIDS	Host-based Intrusion Detection System
HPC	High Performance Connection
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
INRIA	Institut National de Recherche en Informatique et Automatique

IP	Internet Protocol
IPFIX	Internet Protocol Flow Information eXport
IPS	Intrusion Prevention System
ISP	Internet Service Provider
ISS	Informatique, Signaux et Systèmes
KDD	Knowledge Discovery in Databases
LAAS	Laboratoire d'Analyse et d'Architecture des Systèmes
LAN	Local Area Network
LIAFA	Laboratoire d'Informatique Algorithmique, Fondements et Applications
LIP6	Laboratoire d'Informatique de Paris 6
LIUPPA	Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour
LRD	Long Range Dependence
MIB	Management Information Base
MIT	Massachusetts Institute of Technology
METROSEC	METrology for SECurity and quality of service
NADA	Network Anomaly Detection Algorithm
NAI	Network Analysis Infrastructure
NATE	Network Analysis of Anomalous Traffic Events
NIDS	Network Intrusion Detection System
NLANR	National Laboratory for Applied Network Research
NoE E-NEXT	Network of Excellence on Emerging Networking Technologies
OD	Origin-Destination
OSCAR	Overlay network Security: Characterization, Analysis and Recovery
OSCARFIX	OSCAR Flow Information eXchange
P2P	Peer to Peer
PCA	Principal Component Analysis
PCAP	Packet CAPture
P_D	Probability of Detection
P_F	Probability of False Alarm
PHAD	Packet Header Anomaly Detection
PMA	Passive Measurement and Analysis
QoS	Quality of Service
R2L	Remote to Local
RAM	Random Access Memory

RENATER	Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche
ROC	Receiver Operatoring Characteristic
R/S	ReScale Range
RST	ReSeT
SATIN	School on Advanced Topics In Networking
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SSSI	Self-Similar with Stationary Increments
TCP	Transport Control Protocol
TE	Traffic Engineering
TFN2K	Tribal Flood Network 2K
TTL	Time To Live
U2R	User to Root
UDP	User Datagram Protocol
WWW	World Wide Web

Chapter 1

Introduction

The handling of data network traffic anomalies is still a critical issue that needs to be solved, since anomalies may affect at some point and with some extent parties of a connection.

Major attempts to limit the main drawbacks due to network anomalies require detection paradigms. Several approaches are able of accomplishing that, each using its own specificities. While some are focused on the anticipation of anomaly occurrences by using predictive models, others are focused on early detection of anomalies. Such approaches try to detect anomalies before they start damaging the network. Current efforts are also concerned with the type of anomaly itself. Hence, the gathering of appropriate information in order to detect and classify traffic anomalies, and like this to restrain the negative effects of anomalies are the challenging topics addressed by this thesis.

Particularly, this chapter presents the main motivations for this work, as well the main contributions intended to be reached with its accomplishment. At the end, the structure of the thesis is presented.

1.1. Motivation

Variability is a characteristic that is always present in traffic. It can be ignored or not, it can be smoothed or not, it can be enhanced or not, but it cannot be removed. Some of the variability depends on intrinsic causes, meaning that it exists whenever there is a communication between two or more parties. Such variability is mainly due to the communication protocols being used and the approaches used to generate and transfer data traffic: live or streaming audio and video, distance education, entertainment, peer-to-peer,

telephony and video-conferencing, as well as numerous new and often still evolving communication protocols.

Extrinsic variability results from the interaction between data flows. Connections are continuously being established over the Internet, ranging in capacity from hundreds of Mb/s to several Gb/s, and continuously sharing resources between them. However, such environments do not have a central authority regulating and checking communication quality, nor any feedback structure to throttle unfriendly practices or products. This significantly hardens data traffic control, and together with intrinsic variability makes network traffic characterization a challenging task.

Common extrinsic causes responsible for traffic variability are network disruptions, which detour traffic from its regular flow and are responsible for enhancing traffic variability. Attempting to stop these events is very difficult due to several factors. The leading ones are the size of the Internet, and the unpredictable behaviour of anomalies. Uncontrolled disruptions are incompatible with providing stable and guaranteed Quality of Service (QoS) – which is one of the main goals of network operators since a few years. Thus, controlling the extent and harshness of disruptions, is one point where major contributions are arising.

Internet QoS is highly sensitive to traffic variability and to a wide variety of disruptions, often designated as unexpected traffic. Disruptions may be induced by failures, by the Byzantine behaviours of some network elements, or more simply by the significant, though not abnormal, increase in traffic levels related for instance to the live diffusion of some popular event. Traffic disruptions more generally include all events that provoke a large change in network traffic characteristics, and that can badly impact the QoS provided by the network. In particular, such disruptions make Internet traffic non stationary. It is important to mention that with a non-stationary traffic, for which the throughput average often changes during time, guaranteeing a stable QoS is even more difficult.

Network attacks are currently one of the most important classes of disruptions that network operators are being faced with. Such events may assume different forms, hardening the task of dealing with them. More, there is a trade-off between the range of network attacks that could be detected, and the accuracy of detection. So, even with all the knowledge available about different classes of anomalies, each one is not yet completely described. Moreover, it seems to be difficult that anomalies will be soon fully described. For example, one of the most popular network attacks are related with Denial of Service/Distributed Denial of Service (DoS/DDoS) attacks. At the beginning such attacks were mainly accomplished by the exploitation of TCP/IP flaws at one target, flooding it with packets. Then, attacks evolved to a distributed configuration, hardening the identification of the sources involved. Currently DDoS attacks use a collection of inoffensive, low intensity flows to reach their goals. The same drawbacks are common to all other classes of anomalies. So, network attacks seem to be always one step ahead of the methods developed to counter back them!

Because of this, current research is mainly focused on increasing the network robustness in the presence of disruptions, and more specifically attacks. Such assignment involves a very accurate and deep analysis of the impact of disruptions on traffic characteristics, in order to explain how they work and how they can decrease the network QoS. Moreover, the diversity and similarities between regular and anomalous traffic may be a burden. On one hand approaches need to perform correctly, on the other hand they must know how to distinguish between what is traffic attack and what is not. Every approach has to solve the dichotomy True Positive versus False Alarm.

Measures to deal with the continuous change of network anomalies and their implications in network quality are diversified, and detection algorithms and Intrusion Detection Systems (IDS) are probably the most common ways of accomplishing it. However, despite their number a lot of available approaches do not attend one of the major concerns: they are not appropriate approaches for real working networks. For instance, some of those approaches are directed to very specific targets, or require specific environments to work “perfectly”. Thus, most of them have low probability of detection, which is far from 100%!

From all these considerations it is clear that traffic disruptions and traffic anomalies in particular, are a main concern in actual networks. This explains the increasing efforts that have been settled until now and still are, in order to restrain, or at least to limit the extent of damage. Hence, it is noteworthy that a major concern of network operators is to possess a set of tools able to help them in assuring acceptable levels of QoS in their networks, and minimum levels of quality agreed with their clients. For all of this, it is the purpose of this work to define a tool able of responding to these major concerns, or at least to contribute to the restriction of network anomalies negative effects.

1.2. Objectives and Contributions

Normal and anomalous computer network traffic share some characteristics. Learning how to differentiate such types of traffic, and taking advantage of such knowledge is an important issue when addressing methods and methodologies to act on irregular/anomalous traffic. The gathering and processing of traffic information involves some level of network monitoring and traffic analysis, which are the basis of the work being developed in this thesis.

The objectives of this work are twofold. First, it is its intention to characterize traffic, be it regular or not, in order to define accurate features able of differentiating its instances. Second, this work intends to implement a tool able of detecting anomalous traffic, and providing enough information about the involved entities. Such knowledge is important to restrain the damages produced by anomalous traffic. Moreover, this tool is intended to limit the QoS degradation and defeat the lack of security, currently experienced by most of networks.

The tool developed, NADA – Network Anomaly Detection Algorithm, is the main contribution of this work. Given a traffic trace NADA allows:

- The detection of anomalies, i.e., determining whether an occurrence is an anomaly or not;
- The classification of anomalies, i.e., determining what kind of anomaly arises. This means determining whether the anomaly is legitimate or illegitimate, and its precise kind (HTTP Flash Crowd, scanning, SYN flooding, etc.);
- The identification of anomalies, i.e., determining all the anomaly constituting packets and flows, sources and destinations.

In addition, NADA aims at being completely generic, and able of working on any kind of data time series issued from incoming traffic or packet traces. Three goals could be reached with such behaviour. First, working with different data time series guarantees accuracy, since each type of anomaly acts differently over traffic parameters, such as the number of packets, the number of bytes, or the number of new flows, to name a few. Second, the diversity of information provided by each data time series used during the monitoring and analysis process improves the correctness of detection, classification and identification of anomalies. Third, because one main concern is also to develop a tool easily understandable and efficiently exploitable by network operators and administrators, it has to work on network and traffic representative features. And these are bytes, packets and flows, and simple statistics.

The detection capability of NADA allows marking time intervals where a significant traffic variation has occurred, by screening a trace offline or online. The classification capability permits among a set of possible traffic anomalies to signal which anomaly is occurring. This point is particularly important, since a variation previously signalled may not correspond to a traffic anomaly, but only to a normal traffic variation. The information required to perform the classification of anomalies is produced by NADA through the simultaneous analysis of the different data time series, at different time scales, and using different IP levels of aggregation. Finally, the identification feature of NADA gives information about the entities involved in the anomaly, such as source and destination addresses and ports.

1.3. Structure of the Thesis

The reminder of this thesis is organized in five chapters, which are shortly described below.

Chapter 2 introduces the major related research necessary to understand the work being developed. Some main aspects introduced in this chapter are related with traffic variability, traffic anomalies and anomaly recognition. Traffic variability is presented as an intrinsic characteristic of traffic, believed to be due to Long Range Dependence (LRD) or self-

similarity, but which can be enhanced by external factors, such as disruptions or anomalies. Traffic anomalies are presented as the result of inconvenient behaviours perpetrated as a legitimate or illegitimate act. And, anomaly recognition introduces signatures as a way of accurately identifying anomalies. Timely and accurate detection of anomalies, allows the selection of the best action to limit damages.

Chapter 3 describes NADA as recursive approach which is threefold: it accomplishes the detection, classification and identification of traffic anomalies. Moreover, NADA fully provides all information required to limit the extent of anomalies by locating them in traffic traces, identifying their classes (e.g., if they are a Denial of Service, a Network Scan, or another type of anomaly), and giving associated features such as, the source and destination addresses and ports being involved. For this purpose, NADA uses a generic multi-featured approach executed at different time scales and at different levels of IP aggregation. Besides that, NADA contributed to the definition of a set of traffic anomaly signatures. The use of these signatures makes NADA suitable and efficient to be used in a monitoring environment.

Chapter 4 presents a first evaluation of NADA, by executing the application over a set of traces. Two types of traffic traces compose such set: documented and non- documented traces. Documented traces were captured during the experimental part of the MetroSec project, and are a collection of traces with well-defined anomalies. Considerations related with the KDD'99 dataset are also presented. Such traces were used to confirm previously defined theoretical anomaly signatures, and to evaluate the performance of NADA. Thus, theoretical signatures were recognized when selecting anomalous flows and relating the involved features. Non-documented random traces are mainly used to perform traffic behaviour characterization. Such traces are available from different projects such as: the National Laboratory for Applied Network Research (NLANR), and the OSCAR project.

Chapter 5 reports the statistical evaluation of NADA, which is a twofold approach. First, ROC curves are considered as a mean to evaluate NADA and the impact of its filtering parameter *K*. To obtain the curves, all traffic traces available in MetroSec database were used, as well a range of different values for filtering parameter. Moreover, an analysis of the best value for the filtering parameter was done, in order to determine the best value to maximize the probability of detection and minimize the False Alarm rate. Second, the performance of NADA in detecting anomalies was compared with other approaches: PHAD and Gamma-FARIMA. PHAD was a result of the DARPA initiative, while Gamma-FARIMA was developed along with NADA during the MetroSec project.

The conclusions drawn from the work described in this thesis are presented in Chapter 6, including the most important results and the main contributions. The experience gained from this work is then used to point out open issues to be addressed in future work.

Chapter 2

From Regular to Irregular Traffic

Network monitoring and measurement tools are used to gather network information such as the real nature of traffic, the way a network is working, or the Quality of Service (QoS) being provided. One major concern of monitoring and measurement tools is that network behaviour is a moving target. Notably, network traffic, being regular or irregular, evolves along time, changing its intensity, its pattern, its variability, and this impacts the development of approaches to study networks and traffic. Particularly, it is being responsible for the change in how analysis is being performed: from macroscopic to microscopic views, from general to specific approaches.

This chapter intends to overview regular network traffic main characteristics, such as self-similarity and long-range dependence, as well how they impact traffic flowing. Such aspect is important, since the aim of this work is to deal with anomalous traffic, particularly the one due to unexpected occurrences such as network attacks. Also, countermeasures such detection systems are presented as a mean of limiting the negative impact of such traffic irregularities.

2.1. Regular Traffic

Depending on final intentions, different sentences could be used to define Internet traffic. For example, an academic definition would sound like "... *network traffic is the conveyance of messages or data through a system of communication such as a router that manages Internet traffic ...*" [FreeDictionary08]. On the other hand, definitions oriented to traffic characterization would use concepts such as self-similarity and long range dependence in their explanations [Willinger97][Crovella96]. Self-similarity and long range dependence concepts were introduced by Leland *et al.* in 1991 during their analysis of high resolution Ethernet LAN traffic [Leland91]. Since then, both concepts are recurrently used to explain

the variability of traffic, and to justify why the simpler traffic models sources commonly used, such as Constant Bit Rate (CBR), Markov or Poisson are not suitable. Figure 2.1 shows that clearly, by exhibiting what would look like traffic from an "almost" CBR source (a), a Poisson source (b) and a real traffic source (c). As it can be seen, artificial sources that follow simpler models are almost constant when compared with real traffic sources, which are continuously variable. Moreover such models exhibit limitations when representing traffic bursts or the dependence relationship that exists between flows, packets and Internet losses – they are unable to capture such features.

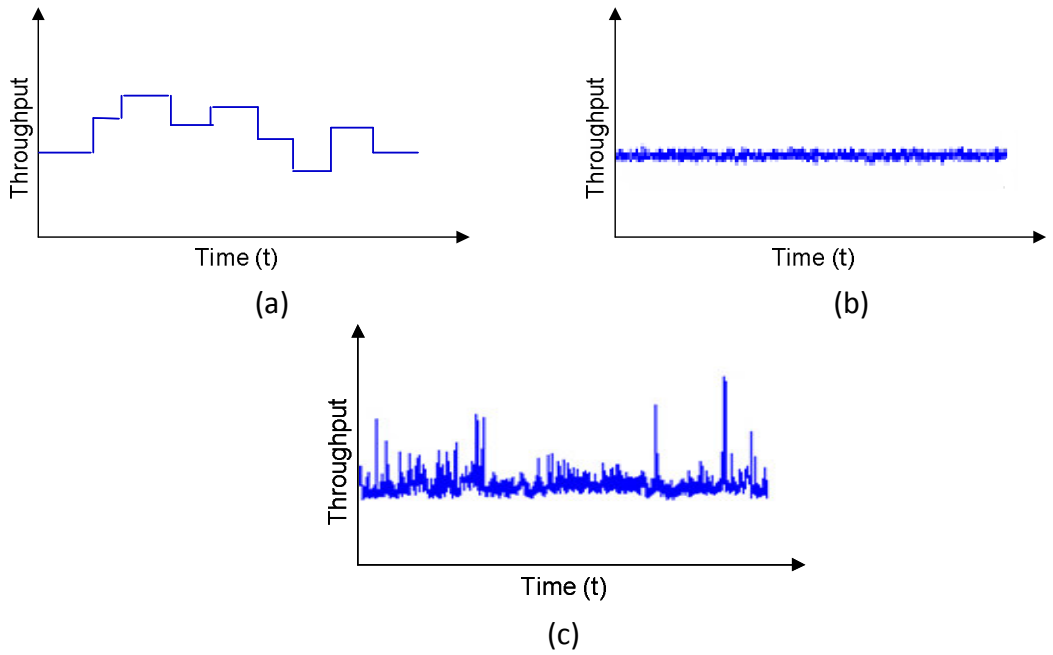

Figure 2.1: Different traffic models commonly used for Internet traffic: (a) "almost" Constant Bit Rate traffic (b) Poisson traffic and (c) Real Internet traffic.

However, simpler models can still be useful under certain conditions. For instance, Poisson model is considered appropriate when modelling core networks, as showed by Cao *et al.* [Cao01]. In their work, they sustained that altogether core network characteristics (such as the link capacities, the size of router buffers, the size of queues, the level of traffic aggregation, etc.) smoothed traffic, allowing its representation by Poisson model. Moreover, it was also showed that such features minimized the impact of anomalous occurrences over traffic behaviour, which remained almost the same.

As stated before, traffic models such as Poisson and similar ones are not suited to represent traffic from access networks. Most of the traffic at access networks is resource constrained, which has reflections on their profile, sensitivity and variability. Changes on traffic profiles difficult the appropriate reservation of resources, as well the straight control of traffic flowing. Sensitivity and variability harden the definition of patterns able of representing traffic.

Hence, control of like-random traffic and improvement of network performance are challenging tasks that despite all, require a full characterization of network traffic. Self-

similarity and Long Range Dependence (LRD) were two of the most important contributions for understanding traffic variability and at some extent quantify it.

2.1.1. Self-Similarity Concept

Mandelbrot introduced the definition of fractal models [Mandelbrot82] in 1982, as an object property that is preserved with respect to scaling in space and/or time. The fractal property may be encountered in a diversity of objects such as natural images, the convergent sub-domain of certain dynamical systems or data time series.

One classical example of an object's fractality is illustrated by Park *et al.* in [Park00] – the 2-dimensional Cantor set living on $A = [0,1] \times [0,1]$. The Cantor set is obtained by starting with a solid unit square, scaling its size by 1/3, then placing four copies of the scaled solid square at the four corners of *A*. If the same process of scaling followed by translation is applied recursively to the resulting objects *ad infinitum*, the limit set thus reached defines the 2D-Cantor set. This constructive process is illustrated in Figure 2.2. The limiting object – defined as the infinite intersection of the iterates – has the property that if any of its corners are "blown up" suitably, then the shape of the zoomed-in part is similar to the shape of the whole, i.e., it is fractal.

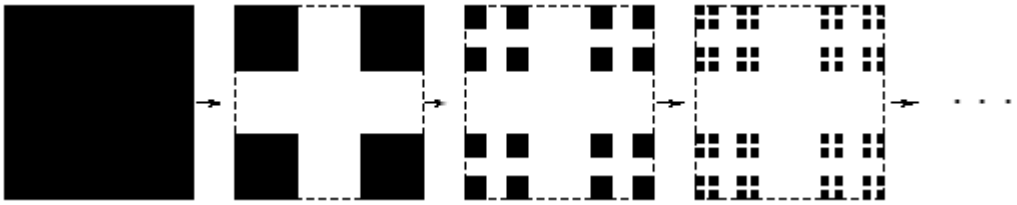

Figure 2.2: 2-dimensional Cantor set. [Willinger97]

In networking context, a classical example of this fractal phenomenon is bursty traffic. A burst of traffic is a sequence of packets that has not a natural length, but always exhibits the same shape independently of the time scale considered. Such behaviour was described statistically using the notion of self-similarity, introduced by Harold Hurst [Hurst51] in 1951, while studying water storage. Self-similarity is used in the distributional sense: when viewed at varying scales, the object's distribution looks the same, i.e., its statistical properties are similar. So, if an object is self-similar, its parts when magnified resemble – in a suitable sense – the shape of the whole.

Figure 2.3 shows self-similar bursty traffic. From right to left, and up to down, the same data time series is plotted: throughput in bytes against time, where time granularity is 100s, 10s, 1s and 100ms, respectively. Each new plot zooms in further on the initial segment by rescaling successively by factors of 10. Unlike deterministic self-similarity, the objects corresponding to Figure 2.3 do not possess exact resemblance of their parts with the whole at finer details (see 2-Dimensional Cantor set above). Here, it is assumed that the measure of "resemblance" is the shape of the graph. Indeed, for measured traffic traces, it would be too much to expect to observe exact, deterministic self-similarity given the stochastic nature of

many network events (e.g., source arrival behaviour) that collectively influence actual network traffic.

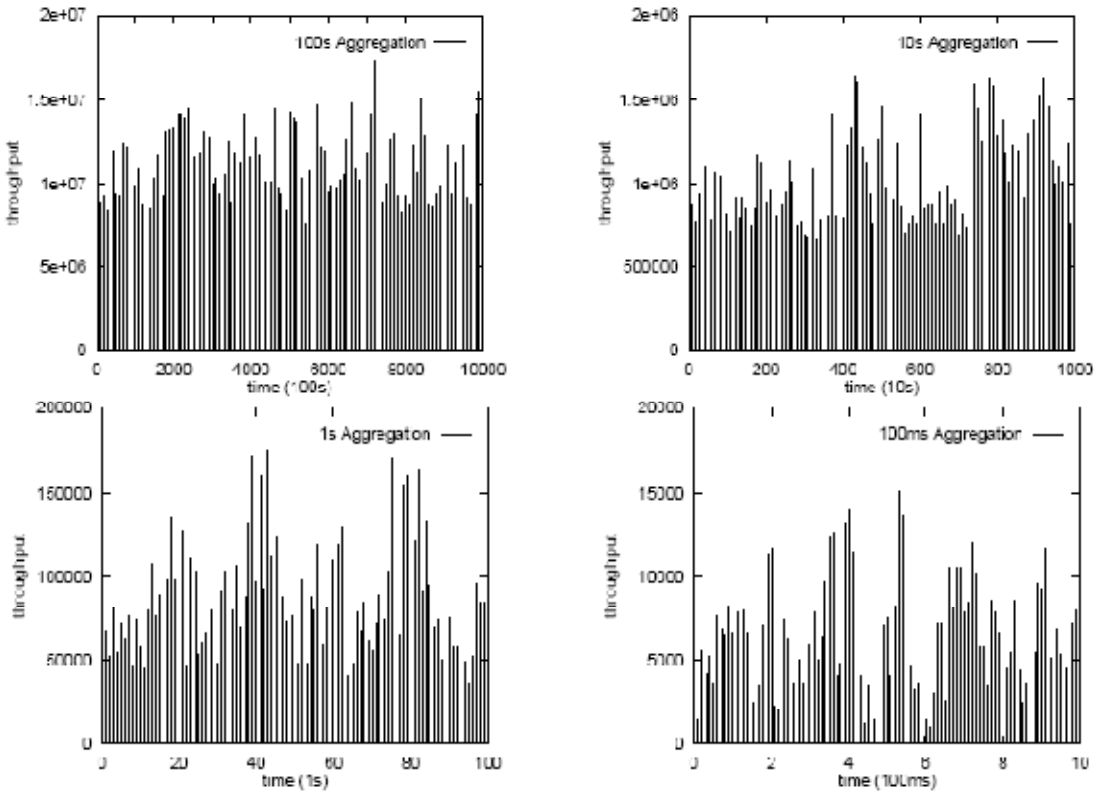

Figure 2.3: Stochastic self-similarity across time different scales 100s, 10s, 1s, 100ms (top left, top right, bottom left and bottom right, respectively). [Willinger97]

Definition 2.1 (Self-Similarity)

Let, $X = \{X(t), t \in \mathrm{N}\}$ be a discrete-time stationary process, and define $X^{(m)}$ and $X^{'(m)}$ as in equations 2.1 and 2.2, respectively:

$$X^{(m)}(t) = \frac{1}{m}\sum_{j=(t-1)m+1}^{mt} X(j) \qquad \textbf{Equation 2.1}$$

$$X^{'(m)}(t) = A_m \sum_{j=(t-1)m+1}^{mt} X_{(j)} = mA_m X^{(m)}(j) \qquad \textbf{Equation 2.2}$$

The process X is said to exhibit self-similarity if X and $X^{'(m)}$ have the same auto-covariance functions for all $m \in Z^+$, where A_m is a sequence (or a set of sequences) of predefined normalizing constants, and A_m is given as $A_m = m^{-H}$, with $H \in [0,1]$. Thus, self-similarity applied to discrete time processes requires the equivalence of X and $m^{(1-H)}X^{(m)}(t)$.

H, the Hurst parameter was defined by Hurst while studying fluid storage (it is not a networking context parameter). However, H is being applied in several areas such as fractals and chaos theory, long memory processes and spectral analysis, for example. Particularly, in networking context, H expresses in some way, the speed at which the data

time series autocorrelation function decays, and is a measure of data roughness. Even if $H \in [0,1]$, for self-similar series, the Hurst parameter must be greater than ½ and less than 1, and as $H \to 1$, the degree of self-similarity increases. Therefore, the fundamental test for self-similarity of data time series reduces to the question of whether H is significantly different from ½. Depending on the value that the Hurst parameter takes, three levels of self-similarity are defined: exactly self-similar when $H = 1$, second-order self-similarity and asymptotically second-order self-similarity. Second order self-similarity (in the exact or asymptotic sense) requires $0.5 < H < 1$, and it is the dominant framework used in network traffic context.

Assigning a value to the Hurst parameter is a challenging task. Nevertheless, the Hurst parameter is perfectly well-defined mathematically, obtaining an exact value for H is not straightforward and, therefore a value to quantify burstiness or data variability. Two classical approaches to estimate the Hurst parameter rely on visual representation of variance and data aggregation information: Variance versus Time plot and Rescale Range (R/S) plot, respectively.

Variance versus Time Plot

This method relies on the slowly decaying variance of self-similar data series. So, if traffic is self-similar, then its slope β is greater than -1. For some historical reason, the relationship between the slope β and H is given by Equation 2.3.

$$H = 1 - \frac{\beta}{2} \qquad \textbf{Equation 2.3}$$

Therefore, H can be calculated by looking at the slope of the variance time function. The values of the Hurst parameter range between 0 and 1. A value of 0.5 indicates true random data series (for example, a Brownian time series), in which there is no correlation between any element and a future element. A Hurst parameter value of $0.5 < H < 1$ indicates "persistent behaviour" (e.g., a positive autocorrelation): if there is an increase from time step t_{i-1} to step t_i, probably there will be an increase from step t_i to step t_{i+1}. The same is true for decreases, i.e., a decrease will tend to follow a previous decrease. When the Hurst parameter value is between $]0,0.5[$ it is said that data time series have "anti-persistent behaviour" (or negative autocorrelation). This means that an increase will tend to be followed by a decrease, or a decrease will be followed by an increase. Such behaviour is sometimes called "mean reversion".

Rescaled Range (R/S) Plot

R/S plot was one of the first methods deployed in Hurst parameter estimation [MANDELBROT69]. It was introduced by Mandelbrot itself and his co-workers in papers published between 1968 and 1979. The Hurst parameter is estimated by calculating the average rescaled range over multiple regions of the data. In statistics, the average (mean) of

a data set X is sometimes written as the expected value, $E[X]$. Using this notation, the expected value of R/S, calculated over a set of regions converges on the Hurst parameter power function, as showed in Equation 2.4.

$$E\left[\frac{R(n)}{S(n)}\right] \cong n^{H} \quad \text{as} \quad n \to \infty \qquad \textbf{Equation 2.4}$$

A linear regression line through a set of points, composed of the log of n (the size of the areas on which the average rescaled range is calculated) and the log of the average rescaled range over a set of regions of size *n*, is calculated. The slope of the regression line is the estimate of the Hurst parameter.

Both approaches, Variance versus Time plot and R/S plot, require that data to be measured is at high lags/low frequencies where fewer readings are available. Also, early estimators (as the previous ones) are biased and converge slowly as the amount of data available increases. But, the most important is that almost all estimators are vulnerable to trends and periodicity in data to be analyzed, and also to external sources of corruption. As a countermeasure, some estimators assume specific functional forms for the underlying model and perform poorly if miss specifications occur.

The Whittle estimator and Wavelets are newer techniques to measure the Hurst parameter, which generally fare well in comparative studies. Both perform correctly even in the presence of highly variable traffic – which is characteristic of current network traffic.

Whittle Estimator

Whittle Estimator [Whittle53] is based on the minimization of a likelihood function, which is applied to the Periodogram of the data time series under analysis. The method gives an estimation of H, as well as the associated confidence interval. It does not produce a graphical output.

Whittle Estimator has a major drawback: the form of the underlying stochastic process must be supplied. The two forms that are most commonly used are Fractional Gaussian Noise (FGN) with parameter $0.5 < H < 1$, and Fractional Auto-Regressive Integrated Moving Average (FARIMA). These two models differ in their assumptions about the short-range dependences in the datasets: FGN assumes no short-range dependence, while FARIMA can assume a fixed degree of short-range dependence.

If the user miss-specifies the underlying model then errors may occur. Local Whittle is a semi-parametric version of Whittle Estimator which only assumes a functional form for the spectral density at frequencies near zero [Robinson95].

Wavelet Analysis

Wavelet analysis has been used with success both to measure the Hurst parameter and also to simulate data [Doukhan03]. Wavelets can be thought as a kind of Fourier transforms but using waveforms other than sine waves.

The fundament of this analysis consists in the extraction from traffic of all possible wavelets. To accomplish this, a wavelet structure is chosen among the ones available (the octave structure is the most common choice when the intent is to look for Hurst parameter), and the associated functions are used at different frequencies in order to capture different scales of oscillation in data series.

A wavelet spectral density plot is generated from the wavelet power spectrum. The equation for calculating the normalized power for octave *j* is shown in Equation 2.5, where the power is calculated from the sum of the squares of the wavelet coefficients (the result of the forward wavelet transform) for octave *j*.

$$Pj = \frac{1}{2^j} \sum_{i=0}^{2^j - 1} C_i^2 \qquad \textbf{Equation 2.5}$$

A wavelet octave contains 2^j wavelet coefficients. The sum of the squares is normalized by dividing it by 2^j, giving the normalized power. In spectral analysis it is not always necessary to use the normalized power. In practice the Hurst parameter calculation requires normalized power.

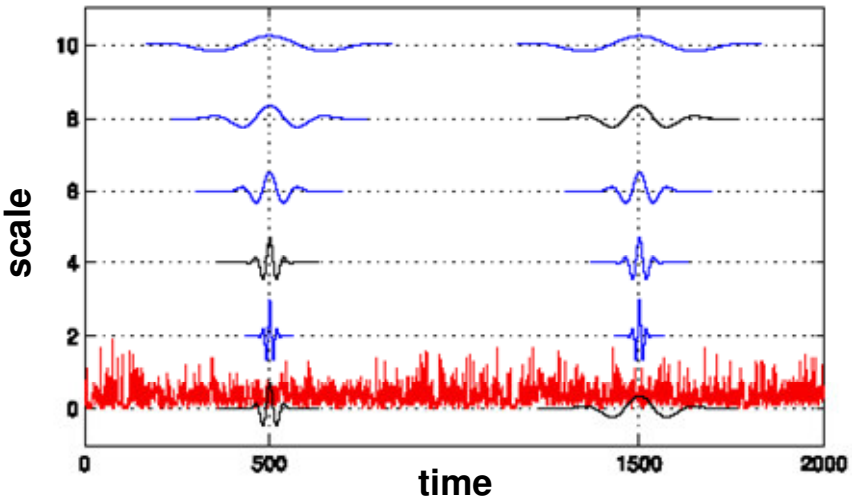

Figure 2.4: Wavelet analysis of traffic Internet (time granularity: 1 second). At each iteration, the wavelet function uses a new frequency f_i, and looks for an identical pattern in data time series. [Larrieu03]

The wavelet analysis is probably one of the best approaches to represent an oscillation (as it can be seen in Figure 2.4). The functions with the higher periods are associated with more persistent waves, i.e., those that generate longer flows or bursts.

The representative characteristic of self similar traffic is its persistent burstiness at different time scales. Because wavelet approaches conduct their analysis by looking at different frequencies, i.e. time scales, for a same data time series, it seems to be appropriate to highlight self-similarity. Thus, the estimated Hurst parameter will reflect the occurrence or not of traffic variability at different time scales.

With more or less extent, all the previous methods are able of capturing traffic burstiness. However traffic variability is also captured by second order statistics. Particularly, the autocorrelation function is a yardstick with respect to which scale invariance can be fruitfully defined. The shape of the autocorrelation function plays an important role. In particular, correlation, as a function of time lag, is assumed to have a polynomial decrease as opposed to exponential. The existence of nontrivial correlation "at a distance" is referred to as Long Range Dependence – LRD.

2.1.2. Long Range Dependence

A process exhibits Long Range Dependence whenever its values at any instant are typically correlated with values at all future instants. The process has some memory of past events, which is progressively "forgotten", as time moves forward.

In traffic variability context, LRD is used many times instead of self-similarity. However, this is not always correct. Some studies such as the one of Beran *et al.* [Beran94] showed that there are self-similar processes that are not long range dependent, and vice-versa. For example, Brownian motion is self-similar with stationary increments and $H = 1/2$, but its incrementing process is white Gaussian noise which is not long range dependent. Conversely, certain FARIMA time series generate long range dependence but they are not self-similar in the distributional sense.

However, in the case of asymptotic second order self-similarity by the restriction $0.5 < H < 1$ in the definition, self-similarity implies long range dependence, and vice versa. It is for this reason and the fact that asymptotic second order self-similar processes are employed as "canonical" traffic models, that sometimes self-similarity and long-range dependence are used interchangeably when the context does not lead to confusion.

LRD can be specified in two different domains: temporal and spectral. In the time domain, LRD manifests as a high degree of correlation between distantly separated data points, and mathematically it is expressed as in Equation 2.6.

$$r(k) \approx k^{-\beta} \text{ as } k \to \infty,\ 0 < \beta < 1,\ \sum r(k) \to \infty \qquad \textbf{Equation 2.6}$$

where $r(k)$ is the second-order data time series autocorrelation function which decays hyperbolically and is non-summable. When depicted the autocorrelation function of an LRD process exhibits a heavy-tailed distribution (see Figure 2.5 (b)), meaning that there are

objects from the data time series that have a value significantly different from the mean, and such objects cannot be ignored.

Data time series $X = \{X(t), t \in \mathrm{N}\}$ can be converted to the frequency domain using a Fourier transform. Let $f(\lambda)$ be the spectral density of the series at frequency λ. So, in the spectral domain, LRD is expressed accordingly to Equation 2.7, and it manifests as a significant level of power at frequencies near zero.

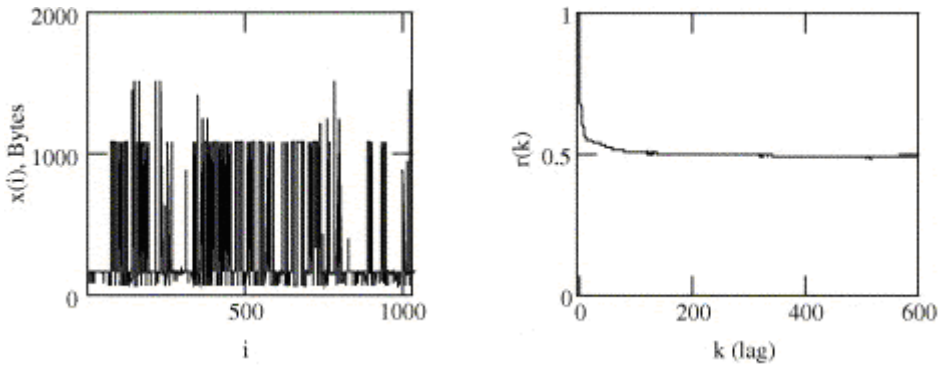

Figure 2.5: Representation of an LRD process. (a) Representation at the time domain. (b) Representation at the frequency domain. [Willinger97]

$$f(\lambda) \approx C_f |\lambda|^{-(2H-1)} \quad f(\lambda) \approx C_f |\lambda|^{-(2H-1)} \quad \text{as } \lambda \to 0,\ C_f > 0$$

Equation 2.7

From equations 2.6 and 2.7, it is possible to conclude that LRD is a difficult statistical property to work with. In the time-domain it is measured only at high lags, strictly at infinite lags, where only a few samples are available and the measurement errors are larger. In the frequency domain LRD is measured at frequencies near zero, again where it is harder to make measurements.

Moreover, as with self-similarity, LRD is well defined mathematically but difficult to calculate. Nonetheless due to the commutability between self-similarity and LRD, the estimation of the Hurst parameter is used to spot the existence or not of LRD in data time series. So, a second-order data time series exhibits LRD if $0.5 < H < 1$.

For instance, approaches such as the LDEstimate tool [Veitch99] can be used to detect LRD and estimate the Hurst parameter. LDEstimate is based on wavelet analysis and it plots the laws that regulate the level of dependence of traffic at different time scales. Also, it represents the level of variability of oscillations according to the observation granularity.

Figure 2.6 shows a typical curve obtained with the LDEstimate tool, when using as input Internet traffic. Because Internet traffic is variable, some level of LRD is expected, as well as a Hurst value between 0.5 and 1. When using the LDEstimate tool, the Hurst parameter is directly obtained from the slope of the LRD curve in the plot. As in other cases, if the Hurst

parameter is between 0.5 and 1, LRD is highlighted, and as much as H approximates 1, the higher is LRD. Inversely, if Hurst parameter is smaller than 0.5 there is no LRD.

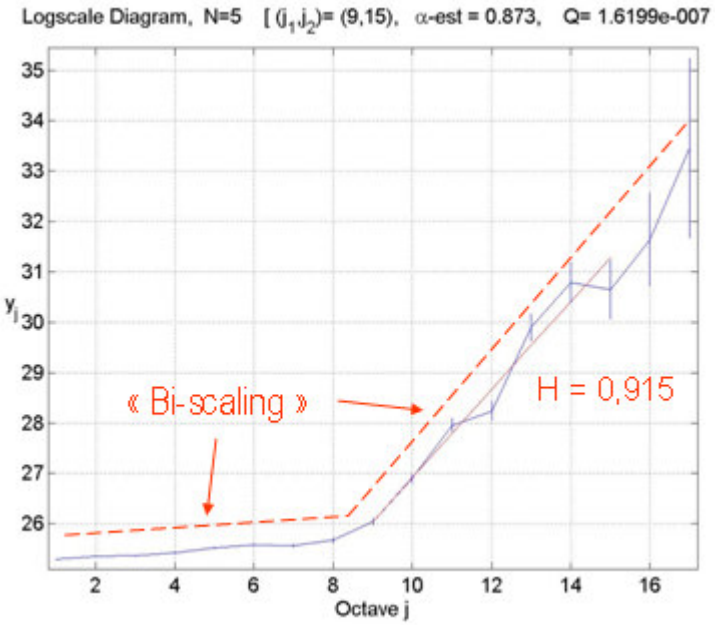

Figure 2.6: Usage of the LDEstimate tool to determine the Hurst parameter. Exhibition of the bi-scaling effect due to the existence of elephant and mouse flows in Internet traffic. [Larrieu03]

Furthermore, in Figure 2.6 it is possible to distinguish different behaviours at two different time ranges (called a bi-scaling phenomenon). Such behaviour is not particular to one trace, but is reproduced with different traces, from different places, independently of the method chosen to evaluate the Hurst parameter.

In this particular case, the border between these two levels of self-similarity/LRD is around octave 8, and highlights two levels of LRD for longer and smaller time scales, which can be translated in two power laws. For smaller time scales (octave < 8), i.e., for packets closer from each others, dependence is smooth. In the other hand, for higher time scales (octaves > 8), i.e., packets that belong to different congestion windows, dependency is much more important [Owezarski04].

Thus, the presence in traffic of very long flows introduces a LRD phenomenon, which is visible at the higher octaves (higher than 12). This level of LRD in traffic represents an important drawback since an oscillation that occurs in time t may be reproduced at any time t_0, which is dependent of t (due to the LRD that exists between the packets exchanged using traditional protocols such as TCP). The bending point corresponds to the average size of flows, as it was shown by Owezarski *et al.* [Owezarski04]. The right part of the plot corresponds to the impact of the elephant flows.

Although self-similarity and LRD are two major concepts in network traffic studies, both are difficult to obtain and to work with, at least directly. More, the problems with real-life data are worse than those faced when dealing with artificial data, and because of that just knowing that traffic is variable is not enough. Real life data is likely to have non-stationary

effects in the sense that its behaviour or structure is variable with respect to shifts in time. This may influence certain types of inference and estimation procedures. So, the naive researches taking a data set and running it through an off-the-shelf method for estimating the Hurst parameter is likely to end up with a misleading answer or possibly several different misleading answers.

2.1.3. Self-Similarity, LRD and Networking

Traffic variability is a characteristic of network traffic that cannot be removed, and that is partially explained with self-similarity and LRD. The knowledge of such variability forced the redefinition of traffic models used until then. Since most of them were inherited from the telephony domain, main resources were considered to be limited by upper and lower bounds, which is unsuitable for currently variable patterns. More, traffic variations do not have consequences only in traffic modelling, but other domains are also affected. According to Park *et al.* [Park00] four different domains may be distinguished, based network functions: physical modelling, queuing analysis, traffic control and resource provisioning, and measurement-based traffic modelling. All four domains are independent, all four domains relate with each other.

The research on physical modelling involves work that explains the causes of traffic self-similarity and LRD, and that provide new insights into dynamic nature of traffic. Most of the explanations use theories based on network mechanisms and properties of distributed systems established empirically, since altogether, they collude to induce self-similar burstiness at multiplexing points in the network layer. In this domain, models are built to try to reproduce exactly how self-similar traffic behaves. More elementary properties are used to perform such reproduction, and most of the times such explanations are based on the nature of traffic itself. So, while the simplest cause for self-similarity and long range dependence is the arrival pattern of packets, more elaborate propositions explain variability in terms of heavy-tailed distribution of files and objects size.

Queuing analysis work provides mathematical models of LRD traffic focusing in facilitating performance analysis in the queuing theory sense. Such kind of work is important in order to establish basic performance boundaries by investigating queuing behaviour with LRD input. Such inputs exhibit performance characteristics fundamentally different from the ones of systems with Markovian inputs.

Traffic control and resource provisioning relates with the control of self-similar network traffic through the correct provision of resources and the correct dimensioning of network elements. An important aspect is to quantitatively estimate the marginal utility of a unit of additional resource such as bandwidth or buffer capacity. A principal lesson learned in the resource provisioning side is the ineffectiveness of allocating buffer space vis-a-vis bandwidth for self-similar traffic [Ryu96] [Grossglauser96]. Another lesson is that traffic oscillations are very damaging for the global use of network resources as the capacity freed

by a flow after a loss for example cannot be immediately used by other flows. This corresponds to some resource waste, and of course a decrease in the global QoS of the traffic and network: the higher the oscillations amplitude, the lower the global network performance [Owezarski05].

Finally, measurement-based traffic modelling is concerned with aspects related with the collection of traffic traces from physical networks and their analysis to detect, identify and quantify pertinent characteristics. Some work showed that traffic characteristics are largely unknown and deviate significantly from standard suppositions. Moreover nowadays regular traffic is not the unique element of flows: anomalous traffic is also present – an object of interest for this work.

The first study that related self-similarity and LRD to network traffic is due to Leland *et al.* [Leland3]. There, time-scale invariant properties of Internet traffic were exposed and the kick-off for hundreds of related studies was settled. Some of those studies were concerned with the effects of traffic variability on regular traffic, and directed their research to domains such as traffic characterization and performance evaluation [Erramilli96] [Owezarski04]. Others such as [Park96] [Willinger97] [Crovella96] focused their work on explaining the existence of traffic variability.

Several causes were pointed then, as promoting self-similarity and LRD in current networks. However, an in deep analysis of LRD and its causes, presented in [Owezarski04], evidenced that it is mainly due to TCP and its congestion control mechanisms – slow start and congestion avoidance. The closed control loop of TCP congestion control mechanisms is defined in such a way that sending one packet in a congestion control window depends on receiving an acknowledgement packet from the previous congestion control window. And, such phenomenon besides existing for consecutive congestion window, also exist for all congestion windows of a flow. Indeed, it was noticed that such behaviour promoted the synchronization of connections to each other when they were crossing the same link or router. And, in case of congestion, all connections passing through the same link or router were impacted at the same time and then started decreasing and increasing their sending rates pretty much at the same time. This means that the global traffic consists of high activity periods during which all connections try to send a maximum amount of data, and idle periods during which all connections reduce their sending rates.

This synchronization characteristic, often analysed as the correlation of traffic, explains the high traffic variability, even when there are no disruption and related performance issues. Furthermore, synchronization is directly affected by traffic profile, which is changing since a few years from small flows mainly associated to web traffic (generically named as mouse flow) to longer and larger flows associated to P2P connections (generically named as elephant flow). One main consequence of such evolution in terms of applications and usages is related to flow size distribution changes [Owezarski04]. The proportion of very long flows has increased in an important way, and current flow size distribution is now very heavy

tailed. In fact, the transmission of elephants creates in the traffic the arrival of a large wave of data that has the particularity of lasting for a long time, and for instance to increase the correlation between different connections behaviours. The side-effects of correlation between flows are particularly exacerbated in the occurrence of packet loss, since the loss of one packet at time t implies the loss of several other packets at time $t+1$. In extreme cases this situation can lead to congestion collapse. This means that the presence in traffic of very long flows introduces very long scale dependence phenomenon and TCP mechanisms are not suited for transmitting long flows on high speed networks [Owezarski04].

A major consequence of traffic variability is related with the complexity of providing stable and guaranteed Quality of Service (QoS) which is one of the main goals of researches lead during the last decade. Actually, guaranteeing QoS means providing the requested QoS under all circumstances, including the most difficult ones. Among the most difficult circumstances, Internet QoS is highly sensitive to traffic variability, and to a wide variety of disruptions, often designated as unexpected traffic, be they induced by failures, by the *byzantine* behaviours of some network elements, or more simply by the significant, though not abnormal, increase in traffic levels related for instance to the live diffusion of some popular event.

Traffic disruptions more generally include all events that provoke a large change in network traffic characteristics, and that can badly impact the QoS provided by the network. In particular, such disruptions make Internet traffic non stationary. Recent Internet traffic monitoring projects showed such issue with the very versatile traffic, whose characteristics are very different from one link to the other, and rapidly changing during time. It is important to mention that with a non-stationary traffic, for which the throughput average often changes during time, guaranteeing a stable QoS is even more difficult.

2.2. Irregular Traffic

As stated before, variability cannot be dissociated of network traffic, and until now several efforts were accomplished in order to develop models able of quantifying such variability and to integrate it in networking tools. However, the amplitude of oscillations of network traffic is not only due to intrinsic causes. It also may be due to external factors. One of particular interest is anomalous traffic which impacts regular traffic transforming it into irregular traffic. Traffic anomalies may be perceived as unusual events – some of which being malicious – that affect the regular flow of traffic, and could increase or decrease traffic parameters (for example, number of packets, number of bytes, number of flows, etc.).

While many efforts related with normal network traffic variability have been accomplished until now, the same is not true for irregular traffic. Anomalous traffic, being an unstable element that changes regular patterns, still remains unpredictable, hardening the task of dealing with it.

2.2.1. Network Anomalies

Network traffic is very sensitive to disruptions independently of their intensity, their source and their type (legitimate or illegitimate). Indeed, most of the times traffic disruptions are responsible for changes in the QoS and network performance perceived by network users, breaking the Service Level Agreement (SLA) committed by the Internet Service Provider (ISP). Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are classical examples of network anomalies, which may be responsible for significant losses to ISPs. But it is in fact the same problem with any kind of disruption that can arise such as crashes, byzantine behaviours of network components, or even legitimate disruptions, such as the broadcast of a very popular event (concert, sport event, etc.). All of these events are identified as being able of impacting the good working of a network, impeding it of providing the services it committed to provide.

ISPs need to detect these anomalies as they occur and then need to classify them in order to select the appropriate response. For example, after a link disruption it would be possible for an ISP to automatically change the routing and the traffic matrix, in order to better balance the increase of traffic among all possible paths. The major challenge in automatically detecting and classifying anomalies is that anomalies can span a vast range of events: from network abuse (e.g., DOS attacks, scans, worms), to equipment failures (e.g., outages), to unusual customer behaviour (e.g., sudden changes in demand, Flash Crowds, high volume flows), and even to new and previously unknown events. A general anomaly diagnosis system should therefore be able to detect a range of anomalies with different structures, distinguish between different types of anomalies and group similar anomalies. This is obviously a very ambitious goal.

But, automatic detection and identification of network anomalies is laborious since anomalies are a moving target. It is difficult to precisely and permanently define the set of network anomalies, especially in the case of malicious anomalies. More, new network anomalies are continuously appearing, and will continue to arise over time; so an anomaly detection system should avoid being restricted to any predefined set of anomalies. This is becoming the main concern of most recent approaches related with anomaly detection.

2.2.2. Detection and Classification of Anomalies

The research on anomaly detection and classification was fostered by the perception that network traffic could be irregular with its own patterns, and previous models based on regular traffic were unsuitable to deal with anomalous traffic. Because detection and anomaly classification are both prolific domains of research, the diversity of existing proposals makes the attempt of grouping them very hard.

Hence, detection processes may follow an offline or an online approach. The post-mortem analysis permits a more complete study of data and the extraction of information that

otherwise would be impossible to obtain. On the other hand, online approaches reveal themselves more desirable since they do not require saving large amounts of data. Moreover, the detection process can be accomplished on the fly. However, online approaches may require special and expensive equipment, able to process data promptly, as it is the case when capturing network backbone traffic.

Most of the times, performing online or offline analysis constrains how the data to be analyzed is obtained, the latter having a larger choice of options. One of the eligible sources for post-mortem analysis is Simple Network Management Protocol (SNMP) data, which is available at almost all network equipments. Other data sources are NetFlow and routing protocols tables, as BGP tables. One common point between these sources is that they are usually configured to present sampled data, and not all data packets reaching a monitoring point. If this reduces significantly the amount of saved data, on the other hand sampled data may not be an accurate representation of data flowing since sampling strategies may fail to capture the underlying distribution of the data. In particular, sampling is influenced by heavy traffic from Flash Crowds and DoS/DDoS attacks. In such cases, data collected reveals little information about the other smaller traffic patterns which may contain interesting yet important information about the traffic, as more anomalous flows.

As stated above, the use of online data usually requires dedicated hardware (as DAG cards [Endace08]) and/or software, which are usually expensive. In both cases, due to the significant quantity of data captured per unit of time, a good processing unit is also required. More, since special hardware/software is necessary, it may be difficult to capture information at some network points, as the backbone of current network operators. There the flow of data is significant and difficult to manage.

Another important sorting criterion when detecting traffic anomalies relates with the methodology used to perform traffic analysis. Depending on the background technique used by each detection methodology, they can be loosely classed as: threshold-based, profile-based, wavelet-based or subspace-based. Threshold-based detection algorithms are also known as volume-parameter algorithms. Such algorithms require the definition of thresholds, above or below which, traffic is marked as anomalous. The values assigned to each threshold depend of the traffic parameters under analysis: usually the number of packets and bytes per unit of time.

Denning's model [Denning87] was the first proposal for a detection algorithm, and it was volume-parameter based. Since then several other models have been presented, making threshold-based algorithms one of the most studied and available methods for detection of anomalies. One innovative work is the one proposed by Cormode *et al.* [Cormode04], which detects anomalies by looking for significant large differences in traffic. But, such differences in traffic may be screened over time, between interfaces or between routers, and require the use of deltoids. Deltoids are used as items that have a large difference that is absolute, relative or variational. An absolute difference corresponds to a large difference between the

number of packets sent in one hour and the next. A relative difference is a large ratio of the number of packets sent in one hour and the next. A variational difference is a large variance of the number of packets taken over multiple time periods.

Most of the work in the recent research and commercial literature (e.g., [Arbor08] [Lakhina04] [Lakhina04b]) has treated anomalies as deviations in the overall traffic volume (number of bytes or packets). Since volume parameters are available at almost all data sources such methods are easily deployed, and proved to be effective in isolating large traffic changes (such as bandwidth flooding attacks). However, a significant volume parameter variation is not an intrinsic characteristic of traffic anomalies. This makes volume based methods unsuitable for a large number of new traffic disruptions. Furthermore, volume parameter methods require fixing volume thresholds beyond which the anomalies are detected. Such thresholds may become burdensome since it is not an obvious task to define a line that separates regular traffic from irregular one. This is why such methods are error prone with a significant number of false positives. Such drawbacks may be overcome by "upgrading" the simplest forms of volume parameter algorithms (i.e., data time series plus threshold), or by using a different methodology. Those methodologies perceived that straight threshold-based algorithms are not sufficiently accurate to deal with an all-group of new anomalies that are not "visible" most of the times. So, to outperform the classical approach they claim the use of statistical elements and signal analysis.

Profile-based methodology, as the name implies, requires the definition of a normal profile of network activity. Such profile is defined with statistical elements captured from network data time series. Based on such profiles, the algorithm is able of defining what will look like future network activity, i.e. to predict network activity. Then, when analyzing current traffic, the algorithm compares it with the expected values. If the predicted activity is different from the current activity, then it is assumed that an anomaly is occurring. Profile-based methods use algorithms such as Holt-Winters, Auto-Regressive Integrated Moving Average (ARIMA), and Auto-Regressive Moving Average (ARMA) to perform predictions. The works developed by Brutlag [Brutlag00] and Krishnamurthy *et al.* [Krishnamurthy04] are typical examples of profile-based algorithms. As with threshold-based approaches, profile based also requires the specification of how much a current value may deviate from its expected value. An incorrect specification may rise up significantly the number of false positives.

Wavelet-based methodology uses wavelet technique to divide a given network signal in its frequency components. Then, each component is used to look for anomalies, which duration matches with its scale. So, at low frequency domains, information is mostly sparse, and if detected an anomaly is of long duration. On the other hand, at high frequencies spontaneous changes are detected (e.g., noise), as well as short term anomalies [Barford02]. Such approaches are quite efficient in detecting anomalies. However the requirement of working in the frequency domain may limit the usability of wavelets in such tasks.

The main goal of the subspace-based approach, also known as Principal Components Analysis (PCA), is to identify typical variations in a set of correlated metrics, and detect unusual conditions based on deviations from those typical variations [Dunia98]. In the context of anomaly detection, subspace-based methods separate normal from anomalous network-wide traffic using Origin-Destination (OD) flows. Such OD flows have a set of designated temporal patterns that become characteristic of normal traffic, if they are common to most of all OD flows. The remaining temporal patterns form the anomalous patterns. Anomalies are detected by statistical thresholds on those anomalous patterns. Currently, several approaches for anomaly detection are based on the subspace method. This is the case of the work developed by Lakhina *et al.* [Lakhina05] and by Zhang *et al.* [Zhang05].

As remarked above the classification of detection algorithms in four categories is only a loose attempt. More and more, new coming algorithms are finding their foundations in other domains. This is the case of detection algorithms based on artificial intelligence [Baldi05], on image processing techniques [Kim05], or in heavy-hitters as a mean of detecting high-volume anomalies [Zhang04].

In spite of the increasing diversity and accuracy of algorithms for anomaly detection, current trends are focused on simultaneous detection and classification of network anomalies. This led to the development of more sophisticated approaches that still use part of the background acquired with anomaly detection, but combined with other methods and techniques.

Much of the work in anomaly detection and classification has been restricted to point-solutions for specific types of anomalies, e.g., Port Scans [Jung04], worms [Kim04] [Schechter04], DOS attacks [Hussain03], and Flash Crowds [Jung02]. General anomaly diagnosis methods remain elusive, although some notable instances of anomaly classification are the ones presented by Thottan *et al.* [Thottan03] or Kim *et al.* [Kim04c]. The authors of [Thottan03] seek to classify anomalies by exploiting correlation patterns between different SNMP MIB (Management Information Base) variables. The authors of [Kim04c] proposed rule-based heuristics to distinguish specific types of anomalies in sampled flow traffic volume instead, but no evaluation on real data is provided. However, both of these attempts for anomaly classification rely on specific characteristics of specific anomalies due to specific applications/tools. Such modus operandi is inadequate when considering the everyday new forms of network anomalies. Particularly, those about which nothing is previously known, as it is the case with some network attacks.

Based on the observations that anomalies do not affect all traffic parameters evenly and that more than one type of information is required to perform a good analysis of anomalies, other types of proposals were developed. Moreover, a significant part of such approaches consider IP addresses and port information, generically named as IP features, in their algorithms. Such algorithms are part of what is the trend in anomaly detection and

classification. The authors of [Kim04b] use address correlation properties in packet headers to detect and classify anomalies, while the work in [Kohler02] show that IP address distributions change during worm outbreaks and Zhang *et al.*

The work of Lakhina *et al.* [Lakhina04b] uses the subspace method to detect traffic anomalies, enhanced with entropy and clustering features to classify those anomalies. The classification of anomalies is accomplished by the systematic analysis of anomalies IP features distributions, followed by the assignment of traffic anomalies to distinct categories. Because subspace-based approaches work on OD flows, the detection of anomalies is over network-wide traffic, which enables the detection of anomalies that may be dwarfed in individual link traffic. In spite of the good results being presented by the subspace-based methods, all the mathematical background required has some impact on the detection rate, which is usually significant. Moreover, the definition of correct OD flows is not always possible when performing online analysis of anomalies.

Instead of using only IP features distributions, the work of Abry *et al.* [Abry07] [Dewaele07] detects and classifies traffic anomalies using a Gamma-FARIMA model coupled with a sketch-based approach.

Beside the lack of algorithms that are able of detecting and classifying simultaneously network anomalies, most of these algorithms also require a significant mathematical background, many times coupled to very specific scenarios of application. As expected, such specificity may have impact on the detection rates or difficult the easiness of implementation of the algorithms.

The state-of-the-art of anomaly detection and classification algorithms somehow directed this study to former volume-based methods. Usually, such approaches allow good results in anomaly detection with a minimum overhead, and are easily transported to network administration context – which is one of the goals of this work. However, as it was stated above, the vector (time-series, threshold) needs to be improved in order to provide accurate results. Such improvements need to take into consideration several factors, such as that each anomaly impacts traffic parameters in its own way; an anomaly may affect only one flow or an aggregate of flows; and there is no fixed time scale to detect an anomaly.

Hence, the work developed and presented in this thesis proposes an approach that is completely generic (not restricted to a particular type of anomaly), and that can work on any kind of data time series issued from incoming traffic or packet traces. Also, it allows the simultaneous use of different time series during detection, classification and identification of anomalies, since each type of anomaly acts differently over each traffic parameter. In addition, since one main goal is to keep the tool as simple as possible, it works on network traffic representative features, such as bytes, packets and flows, and it is based on simple statistics. When needed, new data time series are easily integrated in the approach, as for example, time series related with the number of TCP SYN or RST packets. At last, coupling

the classification with the detection of “candidate” anomalies decreases the number of false alarms and enhances the correlation between traffic behaviours and traffic anomalies.

2.3. Intrusion Detection Systems

Since the seminal work of Denning in 1987 [Denning87], which defined profiles of users based upon their activities – some profiles had a candidate-anomaly associated – many Intrusion Detection System (IDS) prototypes have been created. Of particular interest are those IDS that operate on the basis of packet header contents. These ones provide a way to define specific patterns not only on the basis of textual data in the reconstructed TCP sessions, but also on packet fields. These approaches are particularly interesting in the detection of certain classes of attacks (especially, probing attacks) that do not result in valid TCP sessions.

Many researchers have proposed and implemented different models for IDS, which define different measures of system behaviour, with an ad hoc presumption that normalcy and anomaly (or illegitimacy) will be accurately manifested in the chosen set of system features that are modelled and measured. Examples of commonly found IDS in the literature are MADAM ID [Lee99], Bro [Paxson99], NATE [Taylor01], EMERALD [Neumann99] and SNORT [Roesch99] [Snort08].

MADAM ID was one of the first studies to use data mining techniques for intrusion detection. In this context, data mining approaches look for network features such as the number of successful TCP connection, connection rejection, wrong size rate, bytes sent in each direction, or failure to send all data packets to infer about normalcy and anomaly. BRO functions as a high-speed passive network monitor that filters traffic for specific applications.

NATE or Network Analysis of Anomalous Traffic Events system uses statistical clustering techniques to learn normal behaviour patterns in data networks. Training data is used in the formation of clusters, or groups, of similar data. During detection, data points that do not fall into some cluster are seen as anomalous. Attack detection is based on identifying anomalous data values in individual packets.

EMERALD is a large hierarchical system that can respond to threats on local targets and coordinate its monitors to form an analysis hierarchy for network-wide threats. EMERALD IDS contains a statistical component that maintains short and long-term distribution information for several types of “measures”, using a decay mechanism to age out less recent events. It also has a component that combines signatures and anomaly-based approaches. Such component uses previously learned knowledge to determine from a number of features whether the values of those features fits with some normal behaviour (http, ftp, etc.), some predefined bad behaviour (mailbomb, ipsweep, etc.), or neither of these.

Finally SNORT is an open source network intrusion prevention and detection system that utilizes a rule-driven language. It is capable of performing real-time traffic analysis and packet logging on IP networks, as well protocol analysis and content searching/matching. It can be used to detect a variety of attacks and probes, such as buffer overflows, stealth Port Scans, etc.

The diversity of IDS systems led to the necessity of classifying them. Currently, several taxonomies have been proposed [Almgren03] [Debar99] [Axelsson00] [Alessandri01] [Halme95] [Ko97] [Jackson99], and hence, if they are all different in respect to the nomenclature, they are all similar in the way IDS grouping was accomplished. Attending the work developed in this thesis, two of such classifications are presented: according to the scope of protection of an IDS, and according to the detection method used by the IDS.

2.3.1. IDS Classification: Scope of Protection

The rule for classification based on the scope of protection is related with the extent of IDS functionalities. According to that, an IDS may act over an host, a network, an application or a target [Alessandri01].

Host-Based IDS

An IDS is said to be host-based (HIDS) when it runs on individual hosts or devices on the network. Usually, HIDS are software agents that secure critical network servers and desktops that contain sensitive information. In typical implementations agents are loaded on each protected asset. These agents use system resources such as: disk space, RAM, CPU time to analyze the operation system, and application and system audit trails. The collected information is compared to a set of rules to determine if a security breach has taken place. These agents are tailored to detect host-related activity and can track these types of events with a fine degree of granularity (for example, which user accessed which file at what time).

Network-Based IDS

A Network-based IDS (NIDS) monitors activity on a specific network segment. Unlike host-based agents, network-based systems are usually dedicated platforms with two components: a sensor that passively analyzes network traffic and a management system that displays alarm information from the sensor. The sensors may be configured manually according to the policies defined by the security system. Implementations for NIDS can be appliance-based, including sensor and management platforms, or software-based.

The sensors in a NIDS capture traffic in a monitored segment and perform rules-based or expert-system analysis of the traffic using configured parameters. The sensors analyze packet headers to determine source and destination addresses and type of data being transmitted, and analyze the packet payload to learn about the data being transmitted. When the sensor detects misuse, it can perform various security-related actions: log the

event, send an alarm to the management console, reset the data connection, or instruct a device to deny future traffic from that host or network. On such cases, when the IDS is able to both detecting intrusion activities and managing responsive actions throughout the network, it is said to be an Intrusion Prevention System (IPS).

Application-Based IDS

An application-based IDS collects data from running applications. The sources of data may include the event logs from the applications, or other data stores internal to the application. Such type of IDS is like host-based IDS, but designed for monitoring a specific application. It is extremely accurate in detecting malicious activity for the applications it protects. However, this type of specialized IDS may fail to detect attacks not specifically targeting the application.

Target-Based IDS

Target-based IDS is a special case of IDS, also known as an integrity verification system. It uses its own data to detect integrity flaws. It uses checksums or cryptographic hash functions to detect alterations to system objects and then compare these alterations with the current policy. Because the state of the target object is monitored and compared to the activity taking place on the system hosting the object, this monitoring strategy can be efficient for some systems that cannot be monitored using other approaches. For example, monitoring of some commercial applications, over which different users have different privileges and access to different levels of interaction.

2.3.2. IDS Classification: Detection Method

According to [Axelsson00] [Halme95] [Ko97], IDS detection methods fall into two general categories: signature based and anomaly based detection. Signature-based detection typically is done with an expert system by filtering activity according to a predefined set of rules. Signature-based methods match intrusions to exact patterns of stored misuse behaviour. Anomaly based methods seek to characterize normal system behaviour and detect deviations from normal. The trade-off between anomaly based and rule based methods is that rule based methods can't detect new or novel attacks but their false-positive rate is lower. Anomaly-based detection methods have a potential higher false-positive rate due to inexact methods of intrusion identification. It is the inexactness of these methods that allow the detection of new attacks, yet.

Signature-Based IDS

A signature could be defined as a collection of distinctive characteristics that is able of identifying a specific occurrence. Particularly, in an IDS context a signature is a set of patterns that are used to identify a traffic anomaly.

Signature-based IDS are directly connected to a particular type of anomaly through the identification specific patterns. Those can be a piece of code, a sequence of bits in a packet, or a sequence of function calls that are malicious. A key advantage of using such IDS is that they are easy to develop and understand, and there is an earlier knowledge of what network behaviour they will try to identify – the signature clearly copes with a specific attack through a rule set. For example, a pattern-signature may look for particular strings within an exploit payload to detect attacks that are attempting to exploit particular buffer-overflow vulnerability. More, because of the limited rule set used, detection of such signatures can be performed very quickly on modern systems so that the CPU-power needed to perform checks is minimal. The use of regular expressions and string matching in pattern-based signatures makes them prone to false positives.

The association between a signature and a specific pattern means that signatures are strongly dependent on the integrity of the patterns that are used. Indeed, this frequently troubles the recognition of new types of attacks or even older ones in which known code strings have been altered somewhat. Such *modus operandi* is commonly used by hackers, and it is threatening because of the multitude of attack patterns, that may be created by a human or, a worm with self-modifying behavioural characteristics. The overall ability of a signature engine to scale against these changes is hamstrung by the fact that a new signature must be created for each variation, and as the rule set grows, the engine performance inevitably slows down. Moreover, the requirement of previous knowledge about all the anomalies that should be detected by a system, limits the time response of that system. So, the number of false alarms rise up, as the number of new undetected anomalies increases.

Essentially, signature-based IDS boils down to an arms race between attackers and IDS signature's developers, where the delta is the speed at which new signatures can be written and applied to the IDS engine.

Anomaly-Based IDS

Anomaly-based IDS are also known as profile-based IDS, since anomaly detection involves establishing profiles of normal user behaviours, comparing actual user behaviours to those profiles, and flagging deviations from normality. Profiles are defined as sets of metrics – measures of particular aspects of user behaviour, being each metric associated with a threshold or a range of values. Abnormal behaviour patterns indicate intrusion.

Anomaly-based IDS are not fully anomaly-dependent, as signature-based IDS. They learn the normal behaviour of traffic and systems, and then continually examine them for behaviours that frequently accompany incidents and are associated to potentially harmful anomalies. This approach recognizes attacks based on what they do, rather than whether their code matches strings used in a specific past incident. Because of that, they may be used with a broader set of anomalies.

Anomaly-based IDS are characterized by two phases: learning and detection phases. In the training phase, the behaviour of the system is observed in the absence of attacks, and learning techniques are used to create a profile of such normal behaviour. In the detection phase, the profile is compared with the current behaviour of the system, and any deviations are flagged as potential attacks. Detection of possible network intrusions, denial-of-service attacks, and other threats can be accomplished in three different ways: threshold, statistical and learning-based.

When using threshold detection, users set volume thresholds for different types of network traffic. If the volume for a type of traffic, such as e-mail, rises above the threshold the system triggers an alarm. In statistical detection, the distribution of regular traffic in a network is learned, based on data types and connections to other systems. When unexpected accesses occur, alarms may be raised. For example, the statistical approach would determine how much traffic comes from the Web and how much is e-mail. When these distributions change significantly, for example, when more traffic starts going to sites previously not contacted frequently, an alarm may be raised up, signalling an attack. Such behaviour doesn't adapt quickly to rapid, yet, harmless traffic changes. Despite their limitations, both approaches are the most used with anomaly-based IDS. Learning-based systems examine a network over time, and use neural-network or other data mining approaches to learn which specific traffic and system behaviours are potentially harmful. Learning-based systems are not as common because implementing the learning process can be complex and difficult. At this point, a classification system may be used in order to identify the detected anomalies.

The incorrect selection of the metrics that need to be checked in order to distinguish an anomaly from a normal behaviour is one of the major drawbacks of anomaly-based IDS. Such occurrence reflects in an increase of false negatives. Another potential drawback of anomaly-based systems is that when misbehaviours are detected and alerts are generated, it might be very difficult to correlate those alerts back to a specific attack. This is the case whenever an alert only indicates that non-normal traffic has been detected. More analysis is required to determine whether the traffic represents an actual attack and what the attack actually accomplishes. Or, it is possible to associate some sort of pattern-matching to the detection capabilities of anomaly-based IDS, which is the case with the work developed in this thesis.

Because profile-based IDS do not look for specific attacks, they can be used to detect previously unpublished attacks. This is a major advantage for anomaly-based detection. Instead of having to define a large number of signatures for various attack scenarios, it is enough to define a profile for normal activity. Any activity that deviates from this profile is then abnormal and triggers a signature action.

2.4. Conclusion

Network anomalies are unexpected events that in some way disturb the correct flowing of information in a network. Such disturbance may have an instantaneous impact over Internet traffic, and be clearly noticed by the network administrators/users. Or, instead of it, got unnoticed (i.e., do not change in a visible way how traffic is behaving) but affecting traffic's internal structure, and then strike the target harmfully. The negative impact that such occurrences have on network performance is of major concern, either for network administrators or final users. Because of that, so much work has been developed at the detection anomaly domain, and still is.

Dealing with traffic anomalies revealed itself an arduous task, because of intrinsic characteristics of regular traffic. As it was seen above, even regular traffic presents some variability, and distinguishing variability due to normal LRD/self-similarity from the one due to an anomalous event is not always an easy task. A lot of work has already been accomplished, but most of the developments have associated constraints. Or they are specific for a certain type of anomaly, or they are too complex to be implemented, or they have a limited extent (e.g., only detection or classification of anomalies).

The lack of a complete solution that could distinguish accurately between normal and anomalous traffic, that could be generic enough to detect even new types of anomalies, and that could be easily implemented and integrated in an IDS/IPS infrastructure were the main reasons for the work being presented in this thesis. The analysis of all previous work, related with regular traffic characterization, irregular traffic detection and intrusion detection systems contributed to define the major lines for our proposal: an accurate anomaly detection approach that is also able of classifying and identifying anomalies at an early stage.

The main characteristics of the proposal define it as a threshold-based approach that requires the establishment of some limits to distinguish between regular and irregular traffic. As stated above, threshold-based approaches are of simpler implementation (one of the requirements), but require additional features to assure accuracy. This is guaranteed with the simultaneous use by the algorithm of multi-criteria, multi-aggregate and multi-scale characteristics, which are, as far as we know, unique.

The multi-scale characteristic is related with our conviction that some anomalies appear more clearly at some scales, than others. This is the case of anomalies associated to large flows – if only small time scales are used, the long time behaviour of the flow might be truncated, and not recognized. However, the use of different time scales when looking for traffic anomalies is also an advantage in other cases. For example, in routing context it would be appropriate to modify routing behaviour after the identification of some traffic anomalies, in order to maintain performance. In those cases larger time scales are preferable – only anomalies that stay longer than routing updates times are meaningful. In

security context, small- and long-lived anomalies are of interest. So using just one time scale during traffic analysis would, quite surely, let some anomalies pass.

The multi-criteria and multi-aggregate characteristics, contribute to the precise identification and classification of anomalies. To do so the IP-space is scanned totally or partially for the criteria under analysis.

Finally, due to its characteristics the proposal of this thesis is suitable with IDS/IPS functionalities: monitoring of network and/or system activities (detection) for malicious or unwanted behaviour (classification) and can react in real-time (identification), to block or prevent those activities. As stated above, such frameworks may be classed as signature-based or anomaly-based, but our approach does not fully suit any of them. It suits some parts of both categories.

In this proposal, signatures are defined to allow the classification of anomalies being detected, but they are not anomaly dependent. Instead, they are class of anomaly dependent. This means that for each class of anomaly (DDoS attacks, Port Scans, Network Scans) a signature is defined, i.e., the signature is related with how the anomaly affects traffic profile, but is independent of the tool used to generate the anomaly itself. Such signatures are then used at the detection process, where a behaviour profile is searched for. With this approach, behaviours that are suspicious are defined based on historical analysis. For example, when a consecutive range of network addresses receive a small amount of data it normally indicates something unusual such as a Network Scan attempt. The contribution from different domains as traffic characterization, traffic measurement and traffic detection, allowed the definition, in our point of view, of an innovative algorithm – Network Anomaly Detection Algorithm (NADA).

Chapter 3

Network Anomaly Detection Algorithm – NADA

NADA – Network Anomaly Detection Algorithm follows a threefold approach that allows the detection, classification and identification of network traffic anomalies. To reach its goal NADA analyzes traffic using a multi-scale, multi-criteria and multi-aggregation approach, looking for significant variations, which may correspond to a traffic anomaly. The threefold approach allows the detection of anomalies that otherwise would be unnoticed, such as low intensity attacks, distributed attacks, etc.. In order to increase its accuracy, NADA search for anomalies simultaneously in several directions.

The reminder of this chapter is devoted to presenting NADA fundaments, the algorithm details, and some aspects related with its implementation. Also in this chapter, traffic anomalies signatures obtained with the execution of NADA are presented.

3.1. NADA Fundaments

In a lot of systems, network traffic anomalies are perceived by final users when they lack their access to the Internet. This means that the macroscopic impact of an anomaly occurrence is still measurable as a lack of service or a network slow down. However, more and more, the microscopic effects of an anomaly over a network system are also carelessness. More and more, real-time or at least almost real time actions urge to constrain such occurrences, or at least to restrict their negative effect. This is NADA's main goal. For instance, the algorithm being presented looks for significant variations in traffic parameters that may denote an anomaly. Significant variations are signalled by using NADA's core algorithm, expressed in Equation 3.1, below. There, X is a data time series directly obtained from traffic traces, and P is a data series that is obtained from X, in which each value is the

difference between two consecutive values of *X*. Δ represents the time granularity, i.e., the amount of time considered when gathering time series information. Common values of Δ are 10, 30 or 60 seconds. Others may be used, depending on the traffic trace.

$$X = \{x_1, x_2, ..., x_n\},\ x_i = \{\#\ packets \mid \#bytes \mid \#\ flows\}/\Delta$$
$$P = \{p_1, p_2, ..., p_{n-1}\},\ p_i = x_{i+1} - x_i$$
$$\begin{cases} p_i \geq E(p) + k\sigma(p),\ select \\ p_i < E(p) + k\sigma(p),\ reject \end{cases}$$

Equation 3.1

Considering *P* instead of *X* is important because NADA's equation considers that significant anomalies are not those that produce a variation in traffic flow, but those that are responsible for unexpected significant variations, that might disturb network resources. In a practical point of view this means that significant anomalies must be understood as significant variations between consecutive intervals, and not a constant high value in just one interval. Significant variations were introduced by Cormode *et al*. [Cormode04], who formerly named them as *deltoids* and used them to detect significant traffic changes.

The mean and the standard deviation, E(p) and σ(p) respectively, of each data time series are calculated and used to define a threshold. Each value of the data time series that exceeds the threshold might point a traffic anomaly. Since NADA is intended to be executed offline and online, the data time series used to calculate E(p) and σ(p) are different. When considering offline mode, the data time series is obtained directly from the all traffic trace, previously captured. At online mode, new data time series are continuously obtained from traffic trace captures with a minimum size of three time windows Δ (NADA uses the difference between time intervals to look for significant variations, so when calculating *deltoids*, at the first execution a minimum of three consecutive time intervals is required). Hence, at the beginning, a capture with duration 3Δ is performed, and the corresponding data time series obtained. Subsequent data time series are obtained by removing data corresponding to the first time window Δ, and by inserting data of the next new time window Δ (like a sliding window mechanism). Then, at each new time unit Δ, a new window is added, new data time series are calculated, as new E(p) and σ(p).

The type of filtering being performed can be more or less coarse grained depending on the value of the adjustment parameter *k* in Equation 3.1, where smaller values of *k* fine-grain the search. Currently, the value of *k* is assigned manually, ranging from 0.5 to 2.5, being the value 2.0 the most used (and also the per-default value). These values were obtained empirically, after successive executions of NADA from where it was seen that for values of *k* greater than 3.0 no significant variations are detected, while for values of *k* smaller than 0.5 the formula is not effective because $E(p) + k\sigma(p) \approx E(p)$. The automatic attribution of *k* is a task to be accomplished. By NADA's equation, it is intuitive that the value of *k* is intimately related with the traffic trace under analysis. When a traffic trace exhibits larger oscillations, higher values of *k* are suitable, since *deltoids* have also higher values and are still detected

when considering greater thresholds. At the opposite, whenever a traffic trace is smoother, the value of *k* must be smaller. An over dimensioned value of *k* would not detect any *deltoid*. So, the intrinsic relationship between the value of *k* and the traffic trace behaviour is an orientation that should be followed when implementing the automatic assignment of *k*.

3.2. Algorithm Details

When using NADA detection, classification and identification of anomalies is possible due to the execution of the algorithm over three different dimensions, which make this algorithm unique. The multi-criteria dimension states the use of more than one criterion during analysis. The multi-scale dimension states the use of different time scales during analysis. Finally, the multi-level dimension sates the use of different levels of flow aggregation during the analysis. The conjugation of these dimensions is important since, as it was seen by other approaches as [Zhang04][Estan03], traffic anomalies have different impacts on different traffic parameters. They can be detected at any time scale and may affect IP flows differently. When analyzing each potential anomaly over these three perspectives we have found that NADA conduces to a very low False Positive rate, as well a low False Negative rate.

3.2.1. Multi-Criteria

Each network anomaly acts differently over traffic parameters. For instance, a Distributed Denial of Service (DDoS) TCP flooding attack usually increases more significantly the number of packets being transmitted and the number of flows than the number of bytes, while a Flash Crowd anomaly is usually related with the increase of volume of packets being transmitted. This diversity of behaviours suggested that when working with anomalies considering a unique parameter to signal its existence would promote a large number of false negative detections. This is why NADA requires a minimum of two parameters to work properly: number of packets and number of bytes. The current implementation of NADA also considers the number of new flows. However, and because NADA aims at being completely generic, it can work on any kind of time series issued from incoming traffic or packet traces.

The need of working on several traffic parameters to allow the correct detection, classification and identification of anomalies is also very important, as it is intended the tool based on NADA to be efficiently used by network operators. Therefore it has to work on network and traffic representative features, such as bytes, packets and flows, and simple statistics. Parameter related with the number of TCP SYN or TCP RST packets could also be easily added in the algorithm if they are meaningful for the operators. In any case, by using only simple mathematics, it is intended to make NADA easily and efficiently exploitable and configurable by any network technicians.

The evolution along time of each criterion considered when executing NADA is kept in a data time series that is used as input. Hence, each new criterion to be considered during NADA analysis relates with a new time series to be considered. This feature is particularly interesting when working online with NADA. Since NADA is modular, the inclusion of a new data time series only requires slight adjustments in NADA source's code, in order to include the new parameter in anomaly detection and classification. For example, NADA's version at the OSCAR project [OSCAR08] includes the computing of SYN packets time series, in addition to bytes, packets and flow time series.

3.2.2. Multi-Scale

Traffic anomalies do not have the same intensity and consequently the same impact over the network (i.e., the remaining traffic, the final user services, etc.). This is why the choice of the time scale at which anomalies are screened is an important topic. If previous work [Papagiannaki03][Karagiannis04] has established that the use of smaller time scales is appropriate to detect the most common anomalies, as denial of services for example, other work have showed that anomalies do not have a specific time scale for being detected [Mao06] [Kompela07]. According to these observations, and being the main intention to develop a generic approach, it is not appropriated to be restricted to just one time-scale. This is one of the reasons why NADA performs a multi-scale analysis of time series. A more important point for considering different time granularities is the fact that each anomaly has intrinsic characteristics that may be unperceived, if considering just one time scale. For instance, if analyzing the number of flow data series for a DDoS attack, if this one is low intensity, using a small time scale may not be enough to detect significant variations with NADA. In such case, a higher time scale is recommended. Also, if NADA is being used in routing context, considering anomalies that are detected at small time scales is not of interest. It is not affordable for routing algorithms to change at very second!

The different time-scales chosen to screen traffic traces can range from very low values as some milliseconds to several hours. The values used are a function of the traffic trace duration, but also of the type of anomalies that are being searched.

The consideration of higher values, such as 600 seconds, is important when looking for "heavier" anomalies, i.e., anomalies that can only be detected when considering a significant amount of data, since otherwise they remain hidden. One common example of such anomalies is a Flash Crowd that is only detected when counting packets over a considerable amount of time. Otherwise, the small quantities of packets sent per unit of time are not enough to originate a significant variation in the number of packets that may indicate an anomaly.

The smaller values (between 10 seconds and 60 seconds) are the time scales at which more common anomalies occur, as DDoS, Network Scans, Port Scans, etc. Also, some minor

anomalies with respect to their impact on the network are more easily detected when considering even smaller time windows.

3.2.3. Multi-Aggregation

It was showed by previous researches [Lakhina04][Kim06] that just considering volume parameters is not enough to detect accurately traffic anomalies, especially more recent ones, which only affect volume in a very discrete way. This is the main concern solved by the multi-aggregation dimension, which introduces IP information in the analysis process.

In IPv4 environment, the maximum theoretical number of distinct IP addresses is 2^{32}. Working with such a diversity of IP addresses is not affordable, and this is why associated to each IP address there is an IP mask. IP masks range from 1 to 32, and allow the splitting of IP address space in IP sub-spaces that are manageable. Considering higher values for the IP mask, split the IP space in more IP subspaces with fewer IP addresses each. Hence, for an IP mask = n, 2^n subspace will be created, each with 2^{32-n} addresses.

In NADA's context, the aggregation of IP addresses is accomplished using IP masks. Hence, for an IP mask, the IP space will be splitted in several IP subspaces of the same size: some will contain IP flows[1] from the traffic trace, others will be empty. So, the aggregation of IP flows with the same IP address reduces the number of different IP addresses that NADA needs to search for an anomaly: some groups may be empty, and therefore will not be screened.

In a first stage, NADA acts as a simple volume parameter tool that screens for significant variations over all the packets in a trace, i.e., over all IP flows – this corresponds to IP mask 0.0.0.0 (/0). Then, when considering any other IP mask (ranging from /1 to /32) the traffic parameters are still measured but this time, over specific flows. The algorithm presented here has a broader definition and considers a flow as a sequence of packets from any source to a destination identified by the tuple (IP network, mask), and a timeout limit of 64 seconds to the inter-arrival time between two packets having the same five-tuple. Nonetheless, if this flow definition is not considering all the parameters of the Claffy *et al.*'s five-tuple, it is prepared to include them. So, by increasing the IP mask from /0 to /32, and by applying NADA's algorithm at each resulting collection of IP flows, it is possible at the end to point the flows that were responsible for the significant variations, and that may be associated to the network anomaly. More, the introduction of the IP level allows the complete identification of the sources that are sending faulty packets.

The use of different IP masks is also important in the detection of certain attacks, as stated by Mahajan *et al.* [Majahan02]. For instance some attacks are only visible when traffic is still

[1] A flow, accordingly to Claffy *et al.* [CLAFFY85] is a set of packets moving from one source to a destination point, and that is identified through a five-tuple masking (Source Address, Destination Address, Protocol, Source Port, Destination Port) and an idle timeout value for delimiting them.

very aggregated, as it is the case with certain Network Scans. On the other hand, when flows are completely disaggregated, some DDoS attacks are detected.

3.3. Implementation

At a high level, NADA proceeds in three stages: detection, classification and identification of traffic anomalies. The detection capability of NADA by screening a trace of duration *T* allows spotting time intervals p_i, of duration Δ where a significant traffic variation has occurred. The classification capability permits among a set of possible traffic anomalies to signal which is occurring. This point is particularly important, since a variation previously spotted may not correspond to a traffic anomaly, but only to a normal traffic variation. The information required to perform the classification of anomalies is produced by NADA through the simultaneous analysis of different data time series, at different time scales, and different IP levels of aggregation. Finally, the identification feature of NADA gives information about the entities involved in the anomaly, namely the source and destination addresses and ports. Annex A fully describes the implementation and how to use NADA.

3.3.1. Detection Stage

The first stage of NADA is a recursive process in which the lower recurrence is a low cost anomaly detection mechanism that provides information about the existence or not of an anomaly. It also reduces the searching space for further traffic analysis. Successive recurrences operate on data streams with progressively decreasing granularities, and perform more fine-grained analysis.

Each recurrence of the detection stage is associated to a new IP mask (which reduces the IP search space) and requires the computation of new set of data time series (with only information about the possible anomalous flows). At the beginning, the data time series are obtained considering all the packets from the trace, i.e., an IP mask /0 (highest level of traffic aggregation). At the second recurrence, IP mask /1 is considered, which splits the IP address space in two ranges [0.0.0.0 to 126.255.255.255] and [127.0.0.0 to 255.255.255.255], as it can be seen in Figure 3.1. At recurrence *n*, IP mask */n-1* is used, behaving-like a binary search tree approach. From the second recurrence, the new data time series are calculated considering the packets of the flows which have, as IP destination, an address belonging to the IP range under consideration. Due its similarity with the medical tomography, which opens successive windows over a point of observation in order to obtain more accurate information, this approach is also called IP tomography.

In practice, to reduce the execution time, and because of the redundancy between two consecutive levels of aggregation, only some levels of aggregation are considered during the

analysis (e.g. /8, /16, /24, /32, and some intermediate values, if needed, during the different algorithm iterations).

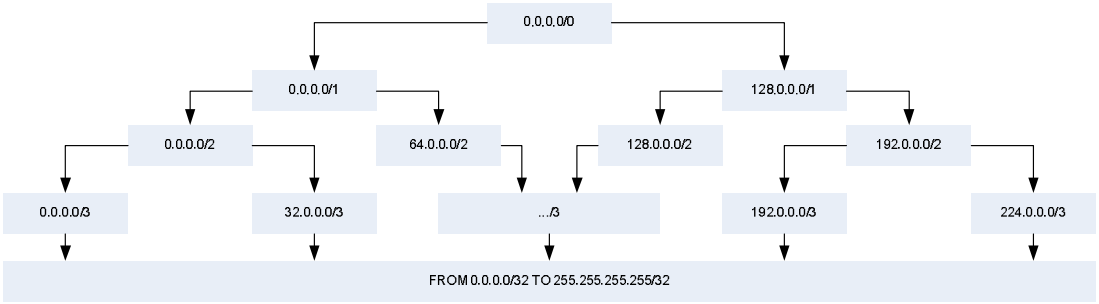

Figure 3.1: Representation of the IP address space decomposition used during the tomography process.

The pseudo-code for the detection stage, in offline and online modes, is shown in Figure 3.2. As it can be seen the input for this stage are the traffic trace (*Trace*), a slot identifier (*S*), the adjustment parameter *k*, the maximum aggregation (*d*) and the time granularity (*Δ*). Since NADA may be executed offline or online, the traffic trace may be a static file containing all the packets captured during a period of time, in the first case. Or, the traffic trace may be a sequence of packets captured during a minimum of three time slots (minimum required by NADA to initiate the analysis), when operating online. In online mode, the sequence of time slots is continuously refreshed.

The function Tomography() is the core function of the analysis process. It allows searching the all-IP space address through a recursive process, considering different levels of flow aggregation. Hence, by ranging the IP mask from 1 to *d*, all possible addresses are analyzed for significant variations, from more to less aggregated IP addresses. For each group of addresses being analyzed, a data time series, per criteria under evaluation, is created according to the packet flows that belong to the group (function Create_DataTimeSeries()). Then, the function Detect_Variation() analyzes each data time series for significant variations. If a significant variation, associated to a flow or aggregation of flows, is detected the type of variation and information about the flows involved are saved. Otherwise, all the intermediate files created during the analysis process are removed (function Remove_Files()).

In Figure 3.1, the all-IP space, and any of its subsets, may be represented as a binary tree. This feature is partly due to reduce the time of analysis required by the function Tomography() to search for variations. When no data packets are associated to a root branch (point of aggregation of flows), the analysis stops – no packets will be found further. Otherwise, the function Tomography() ends up when the maximum level of aggregation defined is reached or when packet flows stop exhibiting significant variations. When a flow stops exhibiting significant variations at aggregation level *n*, at any further level of aggregation (*n+x*, $x \in [1, 32\text{-}n]$), the flow will be more disaggregated, and will not present significant variations.

```
INPUT VARIABLES
Trace ← Name of Trace
S ← Slot to Search
K ← Parameter of Adjustment
d ← Maximum Aggregation
Δ ← Time Granularity

FUNCTION Tomography (ip_space, mask_in, mask_out, S, d)
    Create_DataTimeSeries (*ip_space, mask_in, S, d),
    variation ← Detect_Variation (k);
    IF (variation AND mask_out < d) THEN
        Tomography (ip_space, mask_in+8, mask_out+8, S, d)
    ELSE
        Remove_Files ()
    ENDIF
END

BEGIN
    OPEN Trace
    mask_in ← 0
    mask_out ← 8
    WHILE (packets in Trace) DO
        Tomography (0.0.0.0, mask_in+8, mask_out+8, S, d)
    ENDWHILE
END
```

Figure 3.2: Pseudo-Code of NADA's detection stage.

Currently, NADA accepts two types of traffic trace formats: the ERF format which is created by DAG cards [EndaceProduct08] and OSCARFIX format which was developed in the context of the OSCAR project [OSCAR08]. The slot identifier, *S*, presented above, was introduced in order to reduce the time of execution of NADA – a previous search is made in the trace to locate possible anomalies in time slots of duration Δ. The adjustment parameter *k*, allows NADA to be more or less coarse grained. Currently, the value of *k* is assigned manually, ranging from 0.5 to 2.5, the value 2.0 being the most used. These values were obtained empirically, after successive executions of NADA from where it was seen that for values of *k* greater than 3.0 no significant variations are detected, while for values of *k* smaller than 0.5 the formula is not effective because $E(p) + k\sigma \approx E(p)$.

Finally, the parameter maximum aggregation, *d*, establishes a limit to the number of times the recursive function will be called. It allows stopping the detection stage for values of IP

mask different from /32, which is useful since most of the times there is enough information to classify and identify an anomaly when considering IP masks equal to /24.

The result of this stage is a collection of time slots in which there are flows that have experienced some significant variations, and because of that are possibly anomalous.

3.3.2. Classification Stage

Given a time slot, with a collection of one or more flows to a destination that may be related to a traffic anomaly, the classification stage is responsible for naming the anomalies in that period of time. Because this stage requires some IP information, parts of the anomaly identification stage are also accomplished.

Features	Description
Time Hierarchy	Time windows at which the anomaly is detected. It could range from some milliseconds to several hours.
Network Hierarchy	Information about the source(s) and destination (s) IP address(es) and port(s). Networks can be considered at different levels of IP flow aggregations: /0 (the entire IPv4 address space); /8; /16; /24 subnets; or /32 (single IP address).
Data Time Series	Data time series is a general term to identify any type of time series being considered. At this point the number of packets, number of bytes and number of new flows are considered, by default.
k	Level of filtering.

Table 3.1: Features considered by NADA when defining an anomaly signature.

The classification of anomalies uses signatures, i.e., a set of features that occur whenever an anomaly is present. It was observed, that each type of anomaly acts differently over the set of features presented in Table 3.1: the time scale at which they are detected, the level of IP flow aggregation, the data time series at which a significant variation is signalled, and even the value of the adjustment parameter that was used. The combination of these five features allows the identification of different anomalies, with a very low rate of false alarms (see Section 3.4).

In practice, information about time hierarchy, network hierarchy and data time series is kept through a set of flags that are settled when running NADA. The set of flags is related with NADA's three axis: multi-criteria, multi-scale and multi-aggregation, as it can be seen in Table 3.2.

Hence, if a significant variation in one or more criteria are detected, for a given aggregate of flows and a given time scale, a set of flags is settled to represent such state. Then, as NADA is a recursive approach, more flags are settled as the flows continue exhibiting significant variations at other levels of aggregation and time scales. Tracking the evolution of a

destination aggregate of flows and the correlation of all flags assigned, allow the identification of the anomaly, accordingly to a previously defined anomaly signature.

Axis	Flags	Description
Multi-Criteria	One flag per data time series considered when executing NADA.	Each flag is set to 1 if the anomaly caused a significant variation in the corresponding data time series (by default, FLAG_PACKET, FLAG_BYTE and FLAG_NEW). For example, if the anomaly under analysis was responsible for a significant variation in the number of packets FLAG_PACKET=1, otherwise FLAG_PACKET=0.
Multi-Scale	One flag per time-scale considered when executing NADA.	Per each time-scale at which a flow is analysed, a flag is created. If the flow exhibits a significant variation at that time scale FLAG_TIME=1, otherwise FLAG_TIME=0.
Multi-Aggregation	One flag per level of aggregation considered when executing NADA.	Per each level of aggregation at which a flow is analysed, a flag is created. If the flow exhibits a significant variation at that level of aggregation FLAG_LEVEL=1, otherwise FLAG_LEVEL=0.

Table 3.2: Generic set of flags used by NADA to identify traffic anomalies and how they affect each axis.

Currently, NADA is able of classifying five types of anomalies (alpha flow, DoS, Flash Crowd, Network Scan and Port Scan) described in Table 3.3, that affect differently each of three axis defined by NADA. The first two columns identify and describe the anomalies recognized by NADA. The rightmost column shows how each of the features of NADA (presented in Table 3.1) are affected by each kind of anomaly. Because such features are stable for each type of anomaly, they can be used accurately by NADA to classify those anomalies. Moreover, such characteristics allowed the definition of unique signatures for each type of anomaly.

About the anomalies classified by NADA it is important to notice that usually DoS attacks leave of being detected when traffic flows are highly disaggregated, which occurs most of the times for IP masks values greater than /27. And also, when targeting Scans, it is very difficult to define the appropriate threshold for the time scale and level of flow aggregation at which the anomaly is “visible” or “invisible”. This is because such thresholds change significantly with the anomaly intensity.

Anomaly	Description	Features of NADA Affected
Alpha Flow	Point to point flow with an unusually large volume of data.	▪ Time Scale: any ▪ Aggregation: any ▪ Data Time Series: packet and byte
DoS DDoS	Denial of Service Attack: single-source or distributed-sources.	▪ Time Scale: ≤ 60 sec ▪ Aggregation: ≥ /8 ▪ Data Time Series: packet, byte and flow
Flash Crowd	Unusual burst of traffic to a single destination IP address or network, from a "typical" distribution of sources.	▪ Time Scale: ≥30 sec and ≤600 sec ▪ Aggregation: ≥/8 ▪ Data Time Series: packet
Network Scan	Probes to many destination IP addresses on a small set of destination ports.	▪ Time Scale: depends on the anomaly intensity ▪ Aggregation: depends on the anomaly intensity ▪ Data Time Series: flow
Port Scan	Probes to many destination ports on a small set of IP destination addresses.	▪ Time Scale: depends on anomaly the intensity. ▪ Aggregation: depends on anomaly the intensity. ▪ Data Time Series: packet

Table 3.3: Classification of traffic anomalies by NADA.

3.3.3. Identification Stage

The identification stage initiates after the full completion of the detection part. However, it initiates while the classification stage is still running, since both parts share the IP information of the flows under analysis. As input, it receives a time slot, a collection of flows with significant variations and IP information, such as source and destination addresses and ports.

Anomaly identification is particularly important, since it is the stage that provides information to act over the traffic anomaly. The knowledge of the class of anomaly and IP information about the entities involved are a requirement to select the more appropriate countermeasures. This stage then includes an exhaustive description of the anomaly, using all the information previously collected. Particularly, this anomaly identification is accomplished at the IP level, but could also be at a different point of view level. So, by executing the identification module and using the information obtained from the previous stages it is possible to provide the features indicated in Table 3.4 about any anomaly being classified.

Parameter	Description
Time window	Time window at which the anomaly is detected.
Source identification	Collection of IP addresses and ports responsible for the anomaly.
Target identification	Destination IP address and ports which receive suspect packets

Table 3.4: Parameters used by NADA during anomaly identification.

All the information collected during the identification stage is then saved in a file, which could be sent via TCP sockets to a repository server. Figure 3.3 shows how such file looks like.

It is noteworthy, that the identification of the anomalies requires their previous classification, through the anomaly signatures. The identification part is only an enhancement to NADA, and because it was out of the scope at the beginning, the algorithm employed to perform such analysis is rudimentary, and may be further upgraded.

Besides the creation of a specific file with a complete description of each anomaly being detected and identified by NADA, the tool also produces a set of four plots that altogether allow the classification of anomalies. Each set of plots define a signature anomaly, and may be used by network administrators as a shortcut when looking for network misbehaviours.

```
Time Slot: 2
Time : 30 seconds
-------------------------
Type: DDoS
Destination: 140.93.192.174
Source(s):
193.55.221.1
193.55.221.2
140.77.241.8
129.199.223.67
-------------------------
Type: Network Scan
Destination: 195.78.43.0
Source(s):
189.65.8.45
67.63.145.89
```

Figure 3.3: Example of a file, created by NADA, containing the network traffic anomalies detected during a time slot.

It is important to notice that the classification and identification stages are only possible, if NADA already has some previous knowledge about traffic anomalies. Such learning occurs in an earlier phase, during which NADA is executed against a set of special traces, with specific anomalies, at specific time slots. Such learning phase is responsible for the definition of the database of signatures of traffic anomalies.

3.4. Signatures for Traffic Anomalies

As stated in Section 2.3 signatures are a unique representation of an event. In NADA's context an event is an anomalous network behaviour that may be legitimate or illegitimate. Currently, this work has defined signatures for the following anomalies: Flash Crowd, DDoS attacks, Port Scans and Network Scans, which will be described. Altogether they constitute NADA's database of anomaly signatures.

Each anomaly signature produced by NADA has two parts: four distributions of IP feature information related with anomalous flows and a collection of flags. Both parts are required to allow the correct identification of the anomaly, since some anomalies of different classes present the same IP features distributions. By using a set of flags related with how anomalies affect traffic flows, NADA assures the one-to-one relationship between the signature and a class of anomalies. The flags are the ones presented in Table 3.2.

The set of IP features distributions (addresses and ports) considered in NADA's signatures allow somehow looking inside the data flows that experienced significant variations, and were spotted at the detection phase. Such distributions permit apprehending the number of different sources and targets involved in the flow (and that are responsible for the variability), as well the involved ports. Because each anomaly constrains the relationship between an anomalous flow and the related IP information, these distributions are a powerful tool for classifying anomalies, and defining a unique signature for each of them.

For easiness, such distributions assume a graphical form. At the moment four plots/distributions are enough to classify all the anomalies considered by NADA:

- Distribution of source IP addresses versus destination IP addresses;
- Distribution of source IP addresses/ports versus destination IP addresses;
- Distribution of source IP addresses/ports versus destination IP addresses/ports;
- Distribution of source IP addresses/ports versus destination IP ports.

The use of these particular four distributions is the final results from several attempts. One main attribute of NADA is the use of IP features to classify/identify traffic anomalies. However, using only the simplest features (source and destination IP addresses) was inefficient, since some anomalies have identical distributions for such features. Hence, it was

necessary to increase specificity, which was accomplished by introducing port information. Each pair address/port allows that by associating to each IP source and destination addresses, the corresponding port information. Like this, for example and as it will be seen, a Network Scan may be distinguished from some types of DDoS. Since an anomaly acts differently over each IP traffic feature considered, and it is unknown a priori, it is mandatory to consider the four plots simultaneously, when analyzing traffic.

One set of plots is obtained for each candidate anomalous flow, i.e, a flow that has experienced a significant variation in one or more criterion, at several time scales, and at different levels of IP aggregation. A given flow may be aggregated according to IP mask *m*, $m \in [1, 32]$. For instance, when considering mask *m* = 24, IP address aaa.bbb.ccc.0 aggregates 256 flows, while mask *m* = 32 aggregates only one flow aaa.bbb.ccc.ddd.

Hence, distribution of source IP address versus destination IP address is the simpler distribution being considered. Given an aggregation of destination IP addresses which experienced a significant variation in one or more parameters, the distribution relates the individual IP destination addresses of the aggregation, with the source IP addresses that had sent packets to those destinations.

As stated above, sometimes IP address information is not enough to distinguish between two anomalies (e.g, a DDoS and a Network Scan), or between two types of the same anomaly. In those cases information about the ports involved in the anomaly may be important, hence, distributions 2 and 3 include that information. Distribution of source IP address/port versus destination IP address, relates each source IP address/port that had sent packets to a particular destination IP address from the aggregate that composes the candidate anomalous flow under analysis. Distribution of source IP address/port versus destination IP address/port relates each IP source address/port with the IP destination address/port from the flow aggregate. The more addresses an IP flow aggregates, or the more different ports are being used, the more points are depicted in distributions. Information involved in these distributions is useful in distinguishing different types of DDoS attacks, for example.

Finally, distribution of source IP address/port versus destination IP port was necessary to classify and differentiate scanning procedures, since it relates destination port information with source information.

3.4.1. Flash Crowd

"Flash Crowd" is an English expression introduced in 1973 during a science fiction novella, authored by Larry Niven [Niven73]. The novella related the social consequence of inventing an instantaneous, practically free transfer booth that could take anyone, anywhere on Earth in milliseconds. One consequence not predicted by the builders of the system was that with the almost instantaneous reporting of newsworthy events, tens of thousands of people

worldwide — along with criminals — would flock to the scene of anything interesting, hoping to experience or exploit the instant disorder and confusion so created. In various other books it is suggested that easy transportation might be disruptive to traditional behaviour and open the way for new forms of parties, spontaneous congregations, or shopping trips around the world. A network Flash Crowd has all of this!

On the World Wide Web (WWW), a Flash Crowd can occur when a web site catches the attention of a large number of people, and gets an unexpected and overloading surge of traffic. John Pettitt of Beyond.com first coined this usage in 1996 [Wikipedia08b]. Since then, several notorious examples occurred, including the Slashdot effect, and the access to new Linux releases.

Unlike of other anomalous behaviours that follow, Flash Crowds are most of the time legitimate anomalies, since users are not being purposely harmful – the surge of traffic results from independent decisions among a set of users.

Figure 3.4 illustrates the Flash Crowd signature. As expected, and accordingly to the description of how a Flash Crowd occurs, all graphs exhibit a straight line: a set of sources sends packets to the same destination IP address and port. Notice that the sources use the same port with each packet. Since the targeted destination port is identified, it is possible to know which service is crowded. Though, it is most of the times the HTTP service.

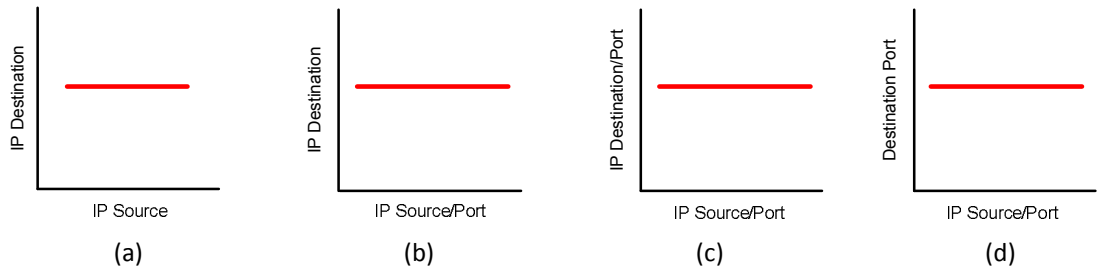

Figure 3.4: Signature for a Flash Crowd anomaly.

In this collection of plots, and for all signatures that will be presented, x-axis and y-axis are IP prefixes of any length *m*, $m \in [1, 32]$. So, if an anomaly appears at different IP prefixes, a set of plots with the same sequence of shapes will be obtained.

3.4.2. Distributed Denial of Service Attacks

The sole purpose of DDoS attacks is to disrupt the services offered by the victim. While the attack is in place, and no action has been taken to fix the problem, the victim would not be able to provide its services on the Internet. DDoS attacks are really a form of vandalism against Internet services. DDoS attacks take advantage of weaknesses in the IP protocol stack in order to disrupt Internet services. The most common DDoS attack, or at least the

one most heard, is the flooding attack. As its name suggests, such attacks are responsible for an abnormal increase in the number of packets, flows or bytes.

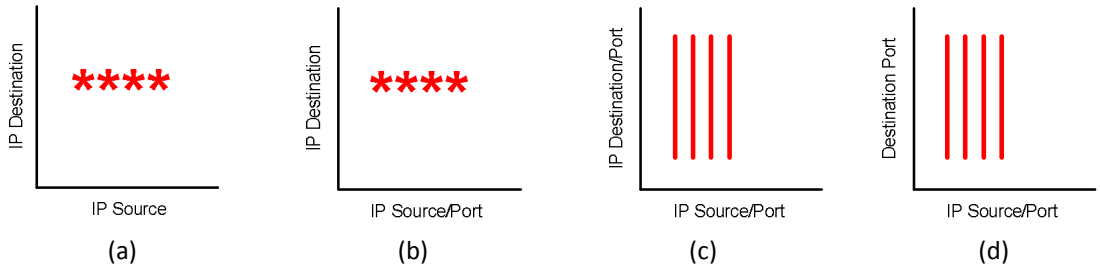

Figure 3.5: Signature for a DDoS anomaly of type: *n* IP source addresses x 1 IP source port → 1 IP destination address x *n* IP destination ports.

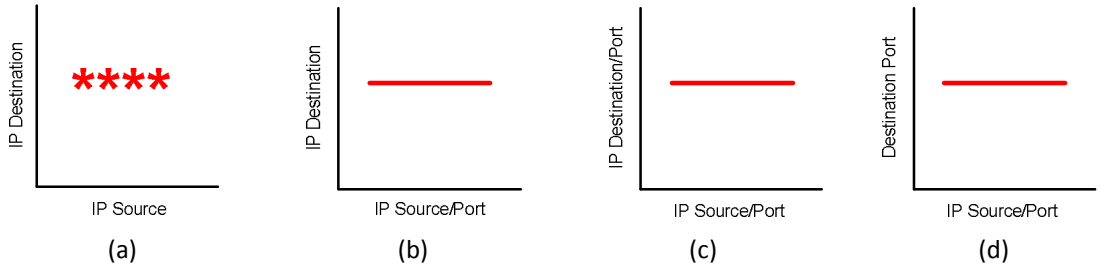

Figure 3.6: Signature for a DDoS anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x 1 IP destination port.

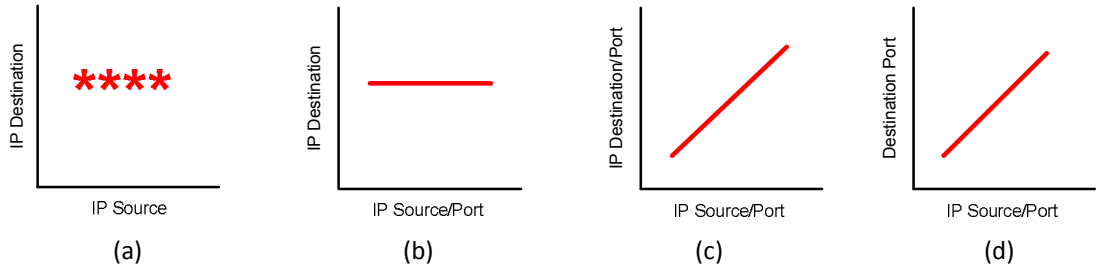

Figure 3.7: Signature for a DDoS anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports.

As expected, a DDoS signature always exhibits a collection of IP source addresses that target a single destination IP address (straight line at plots (a) and (b) of Figures 3.5, 3.6, and 3.7). The usage of stars at plot 3.5 (b) instead of a regular line is because some DDoS sources use the same port number when sending its anomalous packets. Both plots (a) and (b) are enough to signal a DDoS occurrence, plots (c) and (d) being useful to increment the knowledge about the way the attack is being performed: against the same port number, or a collection of port numbers.

3.4.3. Network Scanning

Network Scanning is a procedure for identifying active hosts on a network, either for the purpose of attacking them or for network security assessment. Such procedure is characterized by sending small probes to many destination addresses, on a restricted set of destination ports. In some contexts, Network Scanning is perceived as a generalization of any scanning procedure, as Port Scanning or port sweep.

For an attacker, scanning is one of the three components of intelligent gathering. In the foot printing phase, the attacker creates a profile of the target organization, with information such as its Domain Name System (DNS) and e-mail servers, and its IP address range. Most of this information is available online. In the scanning phase, the attacker finds information about the specific IP addresses that can be accessed over the Internet, their operating systems, the system architecture, and the services running on each computer. In the enumeration phase, the attacker gathers information such as network user and group names, routing tables, and Simple Network Management Protocol (SNMP) data.

Figures 3.8 to 3.10 show the three signatures obtained with NADA for Network Scanning anomalies. From plots (a) and (b) of all figures, it is noticeable that one or more sources, that may use or not different port numbers can perpetrate such anomalies. However, and as expected, more than one IP destination address is implicated in Network Scan occurrences (as showed in plots (c)).

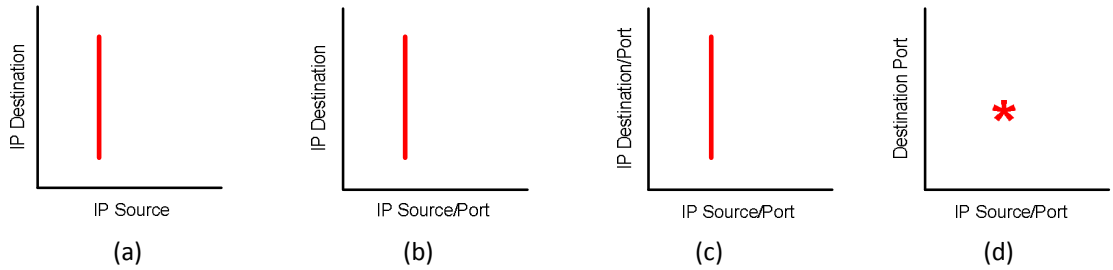

Figure 3.8: Signature for a Network Scanning anomaly of type: 1 IP source address x 1 IP source port → *n* IP destination addresses x 1 IP destination port.

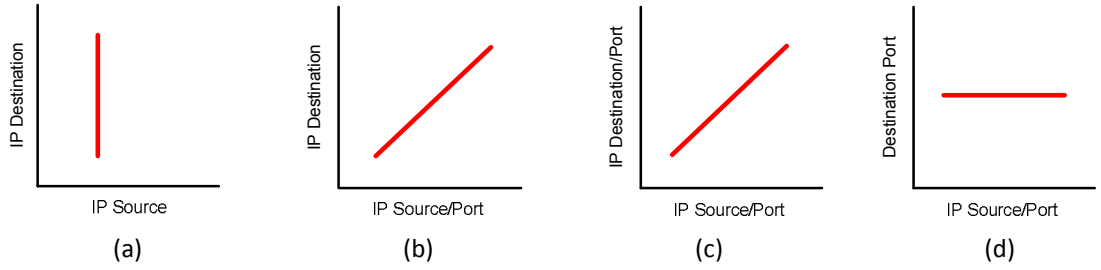

Figure 3.9: Signature for a Network Scanning anomaly of type: 1 IP source address x *n* IP source ports → 1 IP destination address x 1 IP destination port.

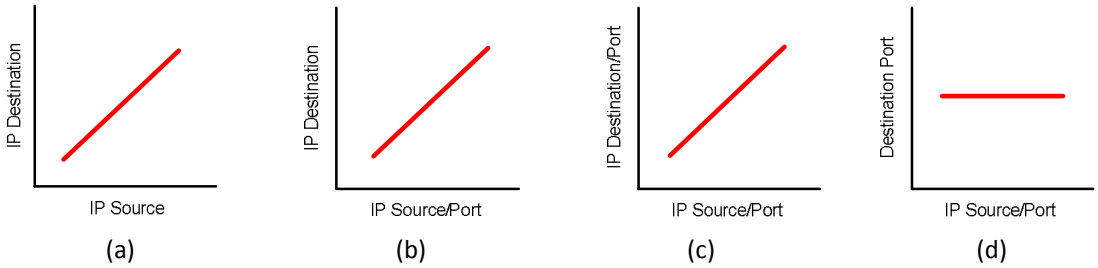

Figure 3.10: Signature for a Network Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → *n* IP destination addresses x 1 IP destination port.

3.4.4. Port Scanning

Port Scanning is a procedure in which a host, or a limited number of hosts, screens for multiple listening ports. Such scanning procedures return information about what services a live host on the Internet offers, and thereby an idea where to probe for weaknesses. Essentially, a Port Scan consists of sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can therefore be probed. Sometimes, in literature Port Scanning is considered as a particular case of Network Scanning.

Currently, TCP/IP is the protocol stack that is the most common on the Internet. In this system, hosts and host services are referenced using two components: an address and a port number, having 65535 distinct and usable port numbers. While most services use a limited range of numbers; others use port numbers in an “as needed” manner.

A diversity of applications is available for scanning (for example, Nmap, Superscan, Scanmetender, etc.): while some directs their efforts to the most common ports or vulnerable ones, others are not so specific, and look for any port. According to [WikipediA08], the result of a scan on a port is usually generalized into one of the following categories:

- Open: when the host send a reply indicating that a service is listening on the port;
- Closed: when the host send a reply indicating that connections will be denied to the port:
- Filtered, Dropped or Blocked: when there is no reply from the host.

Open and closed ports are an open access to vulnerabilities of which administrators must be wary, since in both cases there is a response from the host (at the application or operating system level), which may be exploited.

The information gathered by a Port Scan has many legitimate usages, including the ability to verify the security of a network. Port Scanning can however also be used by those who intend to compromise security. Many exploits rely upon Port Scans to find open ports and then send large quantities of data in an attempt to trigger abnormal conditions. Such

behaviour can compromise the security of a network and the computers therein, resulting in the loss or exposure of sensitive information and the ability to do work.

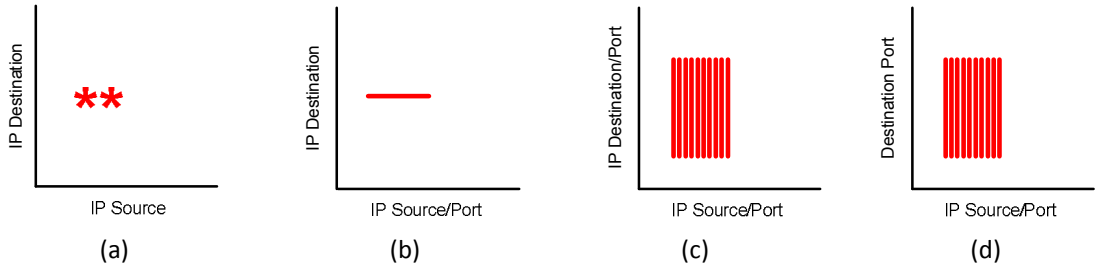

Figure 3.11: Signature for a Port Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports.

The main characteristic of any Port Scanning is that more than one destination IP port number is reached, which is compliant with plots (c) and (d) of Figure 3.11, where each vertical line corresponds to the set of scanning packets sent by each source. In this signature, such plots must be exactly the same.

Furthermore, the exhibitions in Figure 3.11 are generic in respect to the number of IP sources that could be associated to the anomaly. So, it is also possible to have a Port Scan in which only one source is involved, in which case plots (a) and (b) will exhibit a single point, and plots (c) and (d) will exhibit a single line.

3.5. Conclusion

Being anomalies a structural part of traffic, it is important to completely detect, classify and identify them in order to act adequately. Actions over the detected anomalies will be different if they are legitimate, as Flash Crowds, or illegitimate, as DoS attacks. Such differences between anomalies pushed the different teams of researchers to present and develop different approaches.

However, as it was showed, some of those approaches are very performing, but have associated sets of restrictions, that limit their usage in day-by-day network management and security operations.

The need to develop a tool, that could be easily used, integrated and understood, somehow conditioned the development of NADA. The choice of the parameters to be used, how they should be analysed and how results should be prompted, were requisites taken into account. One aspect of particular importance was the one related with the classification and identification of anomalies. First, their inclusion in the same tool that performs anomaly detection is not usual, but more than this, classification and identification of anomalies should be affordable with the simple information available. The usage of signatures to identify anomalies was the option.

The usage of signatures has always been subject to some criticism, because of the strong dependence between the signature itself and the event it represents. So, one main concern when developing NADA was to decouple both. A relationship between a class of anomalies and a signature had to be established, but that relationship could not be founded on specific characteristics of the anomaly itself. It was possible to overpass such limitation by classifying the anomalies by their effect over traffic flows, instead of how they are specifically. For instance, it was a concern to know that a DDoS affected the number of new connections directed to a target IP among others, but completely useless to know specificities of the tool used to produce the attack. Such approach allowed NADA, in a first phase of training to recognize several anomalies, and then use such knowledge when analyzing random traffic traces.

Another important issue when developing NADA was the IP features information related with traffic anomalies. It was noticed that none of the approaches developed by other authors completely exploited its richness. This is in particular the fault of the complex mathematics, statistics and signal processing methods which make difficult to come back from the frequency or entropy spaces to network features easily understandable by network technicians operating networks. So, this work tries to overcome such limitations through a correct collection of traffic data time series. That permits not only the detection and classification of different kinds of anomalies, but also to identify the intervening parties, in an easy way for both configuring the tool and analyzing its outputs. This aspect is particularly important when one of the main goals is to limit the negative effects of an anomaly occurrence in actual networks.

Chapter 4

Experimental Results – Validation of Classification Mechanism

Attacks pose a silent threat to networks, as they can circumvent traditional security measures before a software patch is applied or an anti-virus solution can be updated. So, since the beginning, one major concern of network administrators was the development of countermeasures that could act before an attack becomes harmful to the network. One of such countermeasures uses a set of attack signatures that along with Intrusion Detection Systems (IDS) detect the occurrence of harmful events. Limitations exhibited by such systems along with new types of anomalies, that are continuously appearing, motivated some improvements to signature based systems since they are largely deployed in current networks.

Definition of traffic signatures, from regular or irregular traffic, requires a learning stage. The learning stage of Network Anomaly Detection Algorithm – NADA comprehended the construction of an anomaly signatures database from a set of traffic traces with previously known anomalies. Such traces were obtained in the context of the MetroSec project, which maintains a repository of documented traces.

The remainder of this Chapter presents the repository of traffic traces used, their advantages and drawbacks. It also presents NADA's results, when those traces were used to evaluate it.

4.1. Repository of Traffic Traces

Any detection system that uses traffic signatures requires training time-like. During such period, the system is trained to learn the set of traffic anomalies that it is intended to recognize lately. In NADA environment, such learning period was accomplished by running NADA over traffic traces previously captured, with documented traffic anomalies. This means, that anomalies were purposely created. Non-documented traces include traces were others had already detected anomalies but none was intentionally created. Those traces are used in a later phase to infer about the approach efficiency.

4.1.1. Documented Traces

The validation of NADA could have been done with two distinct databases of documented traffic traces: the MestroSec and the KDD'99. However, as it will be seen, the latter one is unsuited for the current patterns of evaluation, and at the end, only the MetroSec database was considered.

MetroSec Repository

Documented traces used during NADA testing and validation were obtained from MetroSec project [MetroSec08]. Though artificial or simplistic this approach of creating a database of anomalies may seem, such reference database production methodology is a mandatory step for reliable development and validation of applications. Moreover, it allowed the overcome of one of MetroSec major drawbacks. Since its early beginning, MetroSec project faced the lack of a complete and accurate repository of anomalous traffic traces. It is very difficult to obtain sets of traces with well-documented anomalies, in order to assess the relevance and performance of data modelling and anomaly detection procedures. Of course, it would be possible to use the DARPA dataset [DARPAset], which is referred as a standard benchmark for intrusion detection system, or KDD'99 Cup [KDD'99set] based on DARPA 1998 and 1999. KDD'99 provides a connection-based data, and has four attack categories: PROBE, Denial of Service (DoS), User to Root (U2R) and Remote to Local (R2L), like DARPA dataset. However, such databases are made of intrusions simulated in controlled environments, which clearly is not a trustworthy representation of Internet traffic, or even traditional LAN traffic. Moreover, the anomalies considered at the elaboration of the database are somewhat out of range, if current attacks are taken into account.

Hence, one important contribution of MetroSec, besides its core study, was the creation of an actual traffic trace repository containing labelled and documented anomalies that was used for NADA's signatures validation process.

The Framework used to collect the traffic information is depicted in Figure 4.1, where five participants, with two or more machines, acted as attackers: LIP6 and LIAFA at Paris, ISS at Nice, ENS at Lyon and LIUPPA at Mont-de-Marsan, and one as target: LAAS at Toulouse. The

communication between all sites was assured by the RENATER[2] network. All packets sent were collected at the Internet Access Point of LAAS, through a 100 Mbps connection, by a DAG device [EndaceProduct08]. The information about each captured packet was saved accordingly to the ERF format [ERF08], which keeps timing information and packet IP features, such as IP addresses and ports. For some types of attacks, that required a point of control (for instance, DDoS attacks that require a master), machines located at Mont-de-Marsan had such function.

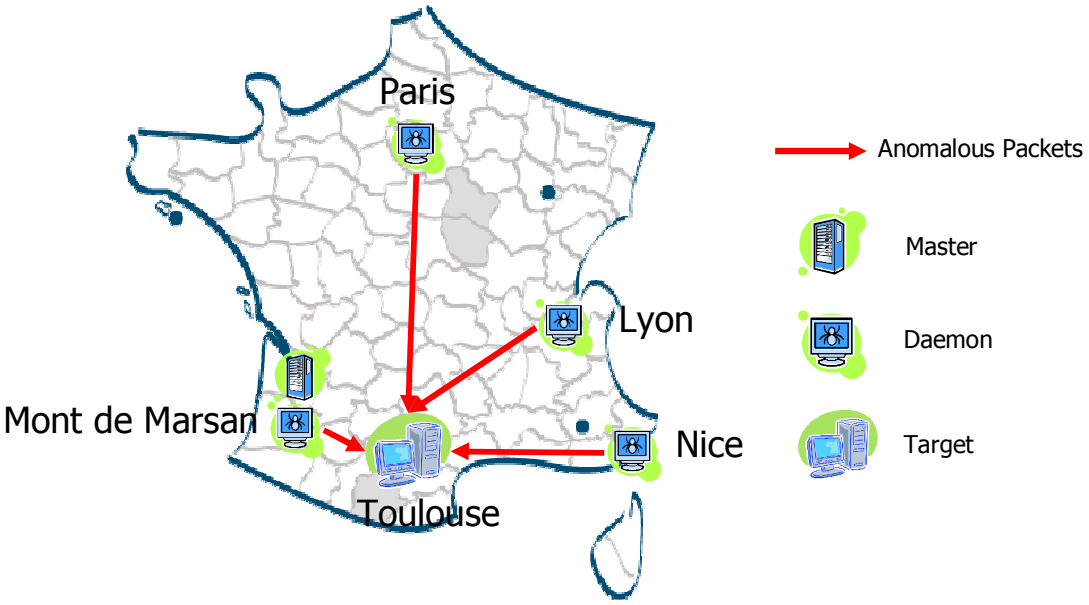

Figure 4.1: Representation of the MetroSec attacking framework.

The MetroSec repository spans different types of anomalies, legitimate and illegitimate ones, with different levels of intensity. Also, distinct types of anomaly generators were used in order to improve the quality of the database. Currently, the database has 46 entries distributed as showed in Table 4.1.

DoS/DDoS were accomplished by flooding a specific target with unexpected UDP or TCP packets. Four different tools were used, all of them freeware and accessible without restrictions from the Internet. IPerf [IPerf08] is not a pure DoS traffic generator. It is intended to tune various parameters and UDP characteristics of a connection. Among others, Iperf reports bandwidth, datagram loss, delay jitter. However, when "incorrectly" parameterized, it may be used to generate and overload, with UDP packets, a defined target. Hping [Hping08] is a network tool able to send customized TCP/IP packets and to display target replies like ping program does with ICMP replies. As with IPerf, specific configurations of this tool may change its default behaviour, leading Hping to act as an UDP flooding tool, as it was the case at the aim of the MetroSec project.

[2] RENATER is the French network for education and research that interconnects academics and some industrial partners, see http://www.renater.fr/ for topology and further information.

Trinoo and TFN2k are different from the previous tools since they were developed with the purpose of perpetrate DDoS attacks. Trinoo [Trinoo08] appeared in 1999 and performs DDoS attacks using a hierarchy of attack, with at least a master and several soldiers. TFN2k [TFN2k08], is a program used by hackers to issue DDoS attacks against Internet FTP and/or HTTP servers. This program has been published in websites as source code making this attack tool potentially dynamic with regard to static code content.

Anomaly	Quantity	Tool	Intensity
Flash Crowd	4	Real accesses at webpage www.laas.fr. Minimum of 30 simultaneous accesses.	34% – 71%
DDoS	10	Iperf – UDP flooding	15% – 58%
	3	Hping – UDP flooding	28% – 99%
	5	Hping – TCP SYN flooding	45% – 91%
	3	Trinoo – UDP flooding	7% – 87%
	3	TFN2K – UDP flooding	4% – 92%
	2	TFN2K – TCP SYN flooding	12% – 33%
	2	TFN2K – ICMP flooding	8% – 10%
	1	TFN2K – Mixed flooding	27%
	1	TFN2K – Smurf	4%
	3	TFN2K Modified – Flat burst with variable intensity	1% – 4%
	3	TFN2K Modified – Bursts with variable throughput and variable intensity	
	3	TFN2K Modified – Ramp Bursts	
	3	TFN2K Modified – Asynchronous bursts	

Table 4.1: The MetroSec repository of traffic traces: summary of anomalies.

From Table 4.1, it is clear that TFN2k was the most used generator, mainly due to its versatility. Of particular interest are the attacks of type Modified TFN2K, which required some changes in the generator's core. The motivation for these new DDoS was to enrich the MetroSec repository information with anomalies with different behaviours – in this case the packets are sent in three bursts instead of just one burst, which is the by default pattern of attack generators.

KDD 99 Repository

In 1999, MIT Lincoln Lab's launched the basements for the development of training datasets voted to the evaluation of Intrusion Detection Systems (IDS). Because such datasets contained traffic traces with documented anomalies, they became quickly the must-use benchmark to analyse IDS or even to train signature-based IDS. Because of this, when

developing NADA, KDD'99 has to be taken into consideration, despite all the criticism on its capabilities.

The KDD'99 intrusion detection datasets are based on the 1998 DARPA initiative, and were obtained through a simulation of a factitious military network consisting of three "target" machines running various operating systems and services. Additional three machines were then used to spoof different IP addresses and to generate traffic. Finally, there was a sniffer that recorded all network traffic using the TCP dump format. The whole simulation covered a period of seven weeks, during which normal connections were created and released, profiling what would be expected in a military network. Attacks on such environment fallen into one of four categories: User to Root (U2R), Remote to Local (R2L), Denial of Service (DoS) and Probe (see Table 4.2), according to the actions and goals of the attacker.

Class of Anomaly	**Description**
Denial of Service	Such attacks have the goal of limiting or denying services provided to a final user, computer or network. A common tactic is to severely overload the targeted system (e.g., smurf, Ping of Death, mailbomb, udpstorm, SYNflood, ...).
Probing	Such attacks intend to gain knowledge of the existence or configuration of a computer system or network. Port Scans or sweeping of a given IP-address range typically fall in this category (e.g., saint, portsweep, mscan, nmap, ...).
User-to-Root (U2R)	Such attacks have the goal of gaining root or super-user access on a particular computer or system on which the attacker previously had user level access. These are attempts by a non-privileged user to gain administrative privileges.
Remote-to-Local(R2L)	In such attacks, a user sends packets to a machine that it does not have access to, in order to expose the machine vulnerabilities and exploit privileges that a local user would have on the computer. It includes for example, dictionary attacks and guest password attacks.

Table 4.2: Distribution of network traffic anomalies at the aim of the KDD'99 project.

Despite its availability and the lack of other data sets to train algorithms, such as NADA, several points come out, against the use of KDD'99:

- The datasets are almost 10 years old, and some of the attacks are passed out, such as DoS;

- The tools used to perpetrate the attacks, particularly those related with Denial of Service, are also passed out. Most of these tools exploited weaknesses on operative systems, that currently do not make sense;
- DoS attacks reported by KDD'99 are of high intensity. Such attacks are of reduced interest for NADA, which intends to be efficient with low intensity attacks, i.e., when they are still harmless.
- U2R category do not have interest in NADA's evaluation, since NADA aims at protecting a network, not a specific host;
- The KDD'99 dataset was created by processing the tcpdump portions of the 1998 DARPA Intrusion Detection System Evaluation dataset, created by Lincoln Lab under contract to DARPA. Since one cannot know the intention (benign or malicious) of every connection on a real world network, the artificial data was generated using a closed network, some proprietary network traffic generators, and hand-injected attacks;
- McHugh [McHugh01] rose that no validation was ever performed to show that the DARPA dataset actually looked like real network traffic. Indeed, even a cursory examination of the data showed that the data rates were far below what will be experienced in a real medium sized network.

Because of all these reasons, KDD'99 dataset was not used when training NADA for traffic anomaly signatures. No value-added was pictured.

4.1.2. Non-Documented Traces

Non-documented traces cannot be used at the learning phase of NADA, since no information is provided about the anomalies that they may contain, or their timing. However, they have a key role when evaluating the behaviour of detection algorithms when in front of independent traces, that were used during the learning phase.

OSCAR Repository

Overlay Networks are virtual networks that are established over physical ones. This allows Overlay Networks to be virtually dimensionless, since theoretically they may use any node and link of the physical network. This feature is particularly important in Peer-to-Peer (P2P) environments in which any node may access resources in any other node of the network, during limited periods of time. If physically implemented, this all-to-all communication would require an extensive communication infra-structure which is not affordable. By using the Internet as an Overlay Network, P2P communications are possible allowing the communication and sharing of resources (files, CPU, multimedia contents) in any-to-any perspective.

An experimental network was setup to conduct P2P experiments, between some of the project partners: LAAS-CNRS, ENS Lyon, INRIA Sophia, ENST Bretagne and France Telecom R&D, and as in MetroSec, RENATER was used as the core network, to which the local networks of each participant are connected via 100 Mbps links. Also, each participant has a DAG card to capture all in-out experimental traffic. P2P traffic is generated through e-Mule clients, or at some experiments traffic is generated through an IXIA generator/analyzer located in the France Telecom R&D local network, which besides regular traffic, is also able to inject anomalous traffic and attacking traffic. Currently, several traces are available at the aim of the project. Some wit purposely inserted DDoS anomalies.

The innovative aspect brought by OSCAR (Overlay networks Security: Characterization, Analysis and Recovery) project consisted in the definition of a prototype able of detecting and limiting attacks in Overlay Networks, in order to preserve performance and restore normal functioning after such attacks. Most of the attacks noticed in Overlay Networks are common ones, such as DoS and DDoS, the compromising of nodes via worms, viruses or Trojan and also the embitterment of network applications and infrastructure by a deterioration of the QoS. Particularly, DDoS attacks that target Overlay Networks look like similar attacks, in which the principal servers (for example, tracker on BitTorrent) are flooded by many requests of non-existent objects, that end up by immobilizing the resources of the network such as bandwidth, memory and CPU.

Another particularity of the OSCAR project was the creation of an internal standard format to export traffic flow information: the OSCARFIX format [OSCAR08], which is recognized by NADA. As its official counterpart, the IPFIX format [IPFIX07], OSCARFIX is intended to unify data formats in the algorithms being developed at the OSCAR project, and to facilitate the transfer of information between applications.

NLANR Repository

The NLANR – National Laboratory for Applied Network Research [NLANR08] project was ended in July 2006, but has still available a full repository of traffic traces that were used during this work. NLANR had several goals, but one of the most important was the characterization of behaviour of High Performance Connection (HPC) networks.

To meet their research goals, NLANR has created a Network Analysis Infrastructure (NAI), which included not only a growing collection of measurement data and multiple analyses, but also tools and methods, numerous avenues for sharing information, and several collaborations with other research groups, both within and outside of the HPC network measurement community. As far as it is known, the NLANR/NAI was the largest project of its kind that made all data publicly available, as well as all analyses, tools, and methods, for use by other network researchers, systems administrators, engineers, and students.

NLANR acted over three different perspectives, being of particular interest to this work the sub-project Passive Measurement and Analysis (PMA), which contributed significantly for

the development of a traffic database. The goal of the PMA project was to deliver new insights into the operation, behaviour, and health of the Internet, for the benefit of network users and operations. Passive header trace data provided the means to study workload profiles for a number of strategically located measurement points in high speed environments. Most of the measurements were taken from OC3 through OC48 speeds, but also a few were taken from OC192mon/10GigE links and OC192c instrumentation on the Abilene backbone.

Of particular interest to this work is the PMA Daily Traces Archive, which contains long contiguous traces. These include Abilene-I, the first publicly available OC48c backbone trace data, Auckland-IV, a 45 days continuous trace, Bell Labs-I, a one week contiguous Internet access IP header trace, and Auckland-VIII a two weeks continuous GPS-synchronized IP header trace collected at the University of Auckland Internet access link in December 2003 – to name a few. All these traces are available at link http://pma.nlanr.net/Special/.

4.2. Anomaly Signatures

As explained in Chapter 3, NADA is a tool that detects and classifies traffic anomalies, and produces according to it, a set of plots that allow a quick identification of anomalies, if any. Such sets of plots are anomaly signatures, a unique representation of an event, i.e., an anomalous network behaviour that may be legitimate or illegitimate.

Each anomaly signature is unique and composed of four characteristic plots. However, generically each plot results of the distribution of the number of sources and destinations, relative to an aggregate of flows that are experiencing a significant variation. So, when looking for such kind of signature or plots, for instance Figure 4.2, from up to down and left to right, the first plot shows the distribution of different source IP addresses, of the aggregation of flows versus their destination IP addresses. Notice that each plot axis does not have specific addresses, in the sense of values such as aaa.bbb.ccc.ddd. Instead they have the number of different IP addresses, involved in the flow being analysed. The same is true for all the other plots. The second plot (up-right) shows the distribution of IP source pairs (address/ port) versus the number of destination IP addresses. The third plot (down-left) shows the distribution of the IP source pairs (address, port) versus the number of IP destination pairs (address, port). Finally, the fourth plot exhibits the relationship between the IP source pairs (address, port) and the destination ports involved.

The database of anomaly signatures obtained in this work contains entries for the following anomalies: Flash Crowd, DDoS attacks, Port Scans and Network Scans, which will be presented. Their presentation also intends to be an informal validation of NADA's classification method.

4.2.1. Flash Crowd

Most of the Flash Crowds might be classified as legitimate anomalies, since some of them are not intentionally perpetrated, but are the result of adverse circumstances. Flash Crowds arise when a resource is accessed by a significant number of connections in a small piece of time, ant it is not prepared for that. The visible consequence is a significant slow down in resource access time, or worst the impossibility of accessing the resource at all. Usually, such anomalies might be overcome by reducing the number of accesses or by increasing network parameters, such as bandwidth or the resource availability (creation on mirrors).

Traffic traces with Flash Crowd anomalies were analyzed with NADA, and all of them exhibited a characteristic set of plots – a significant number of IP sources connecting an IP address/port. If most of these anomalies are related with HTTP accesses, it is also possible to have a Flash Crowd even when the resource is not an HTTP object, but a database repository, for example. However, even in such cases the set of plots is identical. The sequence of plots in Figure 4.2 constitutes the signature obtained from a Flash Crowd on an HTTP server located at LAAS. As expected, all four plots exhibit a straight line, with the same number of IP sources sending packets to the same HTTP resource.

Also, as it can be seen from the sequence of plots in Figure 4.3, the shape of the anomaly signature does not change when changing the number of aggregated flows under analysis, since the one of interest continues in the group of study. For instance, the same shape is obtained with more aggregated (network mask equal /16) or less aggregated flows (network mask equal /24). Such behaviour is not surprising, since when considering higher values for network masks, NADA is just isolating the true responsible packets for the anomaly. Because a Flash Crowd is the sum of independent contributions, increasing the network mask just removes packets from non-related connections.

One important consideration when using NADA to detect anomalies is to adjust the time granularity used when performing detections. Such adjustments may depend on the type of anomaly that is being searched or on the main actions to be taken whenever an anomaly is detected. Particularly, when considering Flash Crowds due to their intrinsic characteristics (i.e., it takes some amount of time before a server become congested) time granularities such as 60 seconds or higher, are acceptably suited to allow the detection and classification of Flash Crowds.

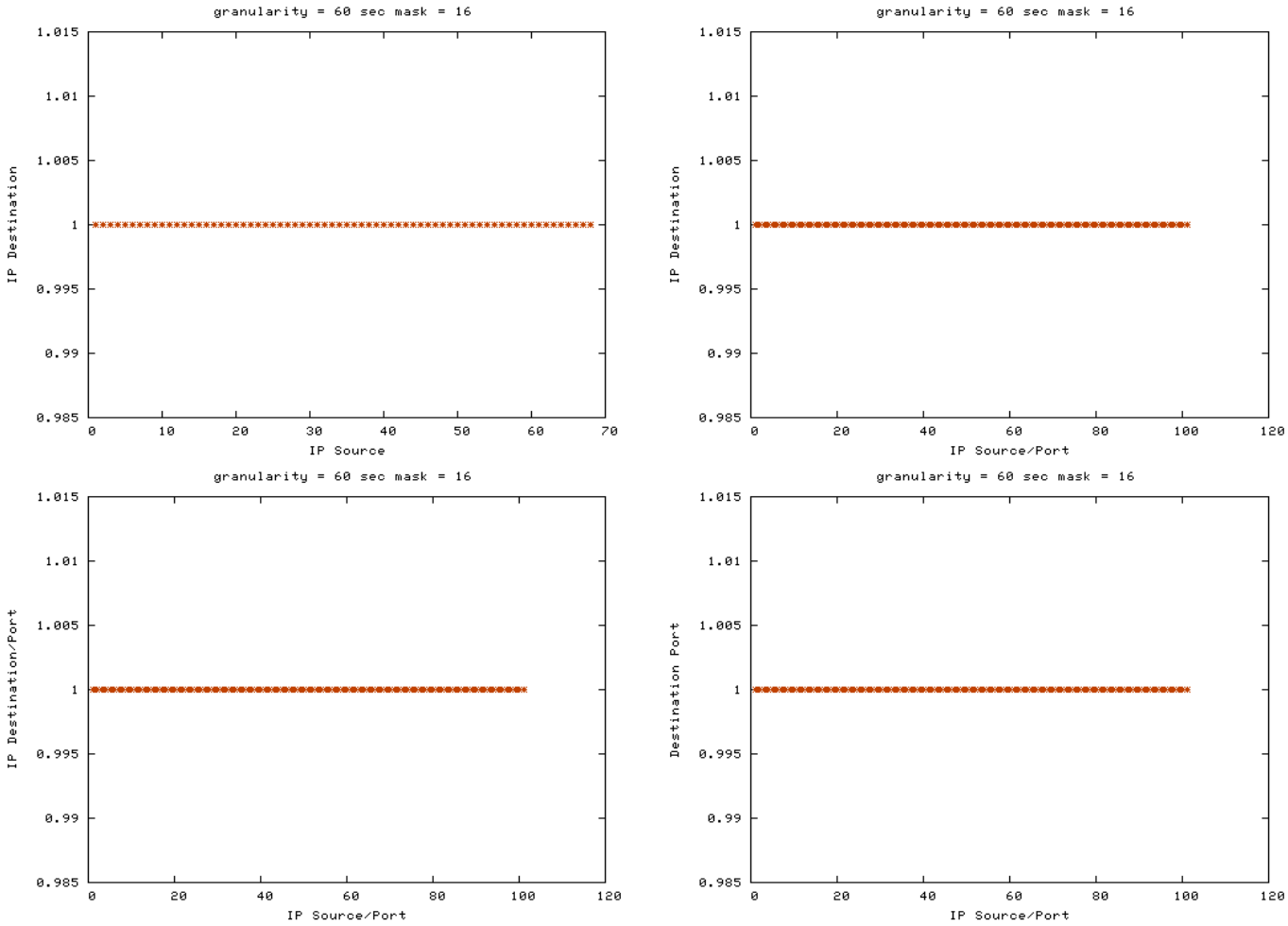

Figure 4.2: Detection of Flash Crowd anomalies using higher levels of network aggregation (mask = 16).

However, time scales of 60 seconds, or even higher, may also increase the number of attacks that get unperceived when running NADA. This is particularly true for other types of attacks than Flash Crowds, which have more irregular signature patterns. So, time granularities in the order of 30 seconds may be more realistic when using NADA as a generic application that detects and classifies more than one type of anomaly. Usually, such anomalies are not as high intensity as Flash Crowds usually are or could be. And, as stated above, network administrators want to detect anomalies at earlier stages, when network resources are still available.

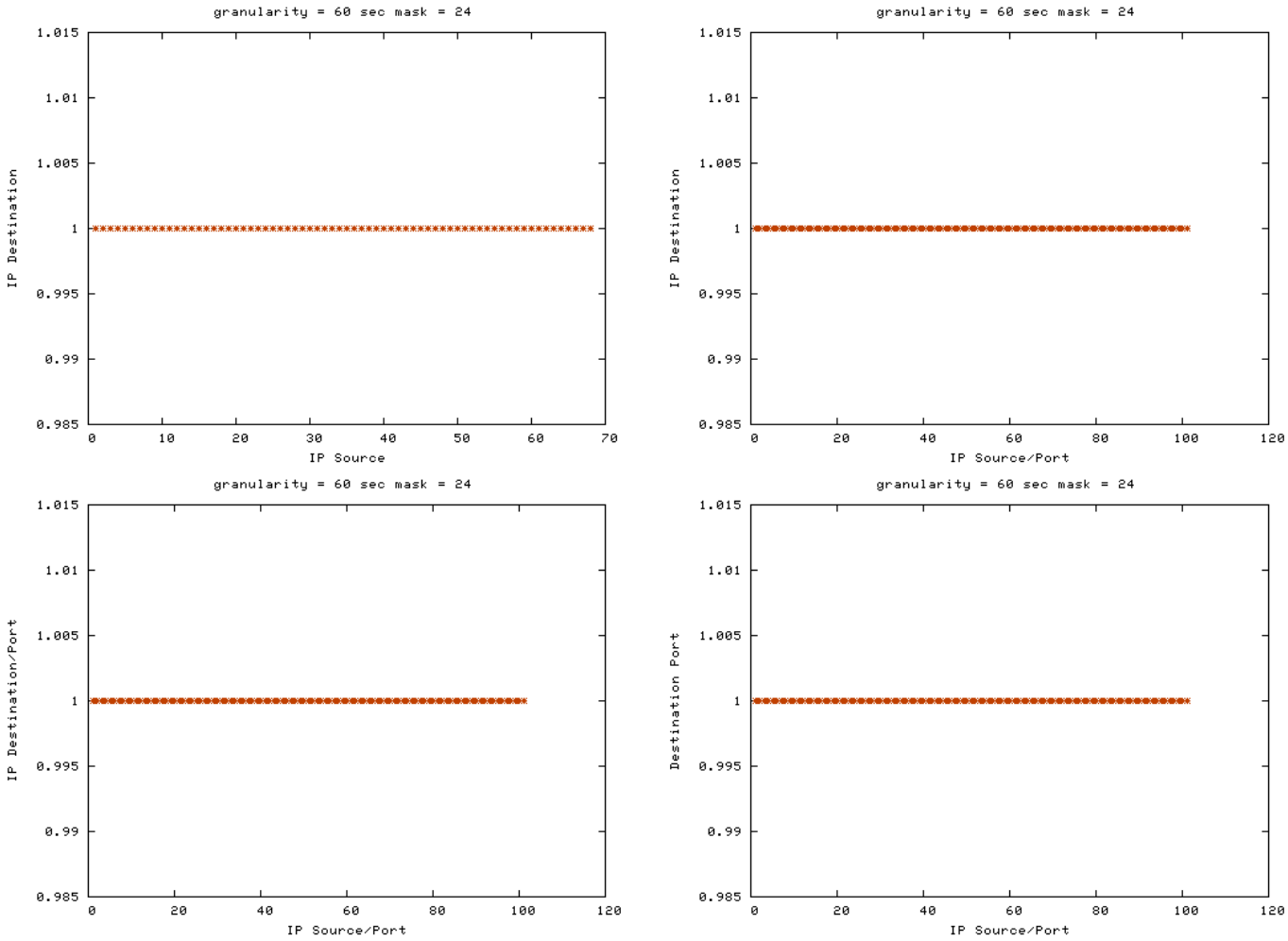

Figure 4.3: Detection of Flash Crowd anomalies using lower levels of network aggregation (mask = 24).

The sequence of plots in Figure 4.4 proves what has been stated before. The detection of Flash Crowds is still possible when lowering the time granularities considered and the level of flow aggregation. Hence, a more refined result is obtained. These last four plots were achieved from the Flash Crowd anomaly example, being presented along this section. Particularly, this sequence was obtained considering a network mask equal 32 (less aggregated flows) and time granularity of 30 seconds. The analysis of Figure 4.4 shows that the Flash Crowd under study is in fact, the result of four low intensity Flash Crowds perpetrated against the same server, but using different IP source addresses and ports. Of course, such result only appears whenever small Flash Crowd anomalies are of moderate to high intensity type. Otherwise, they would not be detected. Relationship between Flash Crowd intensity and its detection is a subject for further studies.

Nonetheless, from all the sequences of plots being presented, it can be noticed that the format of the Flash Crowd signature is the same at all levels of flow aggregation and time scaling. Such behaviour may be considered when configuring NADA for anomaly detection – parameters should be adjusted to more sensitive signatures, such as DDoS ones.

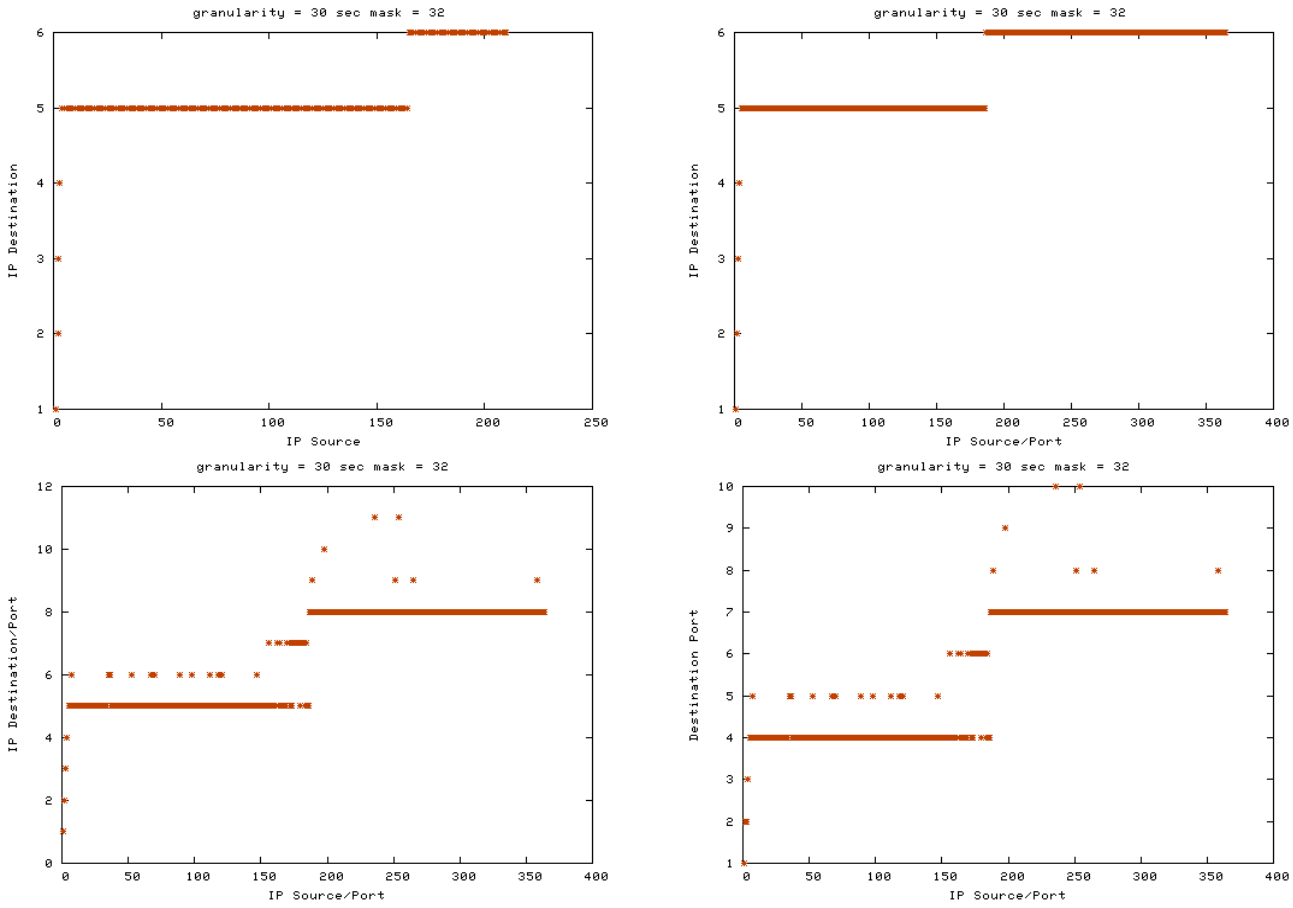

Figure 4.4: Impact of lowering the time granularity (30 seconds) and the level of network flow aggregation (mask = 32) in Flash Crowd detection and classification

Another concern that will require further study is the legitimacy of Flash Crowds. Legitimacy may be considered as the line of separation between a Flash Crowd and another type of anomaly – and it is not mathematically measurable. Intentionally harmful Flash Crowds are becoming frequent, and such events are being classified as DDoS. In such cases, Flash Crowds are defined as attacks that engage many bots, distributed all over the Internet, to send small-rate, legitimate service requests to a target. Aggregated traffic then overwhelms the target, but it is legitimate and often critical to its business, such as Web page requests. The target cannot filter all Web traffic, and bot behaviour is so non-aggressive that they blend in with legitimate clients. Jung *et al.* in their work [Jung02] identify some features that could be used to differentiate between legitimate and malicious (DDoS) Flash Crowd events. However, such type of differentiation was not possible during this thesis by a lack of time.

One last remark about the plots of Figure 4.4 is necessary. When observing that set, one can see that two of them have noise (points beside the ones directly related with the signature). Those points correspond to other flows destined to the same IP destination under evaluation. Still, since they are not organized following one of the signature patterns, they are not considered as anomalous flows. The use of filtering would remove such points. This was not accomplished since it is a time consuming task, which requires algorithms that are

out of the scope of this work. Such kind of noise is particularly evident when analysing traffic traces with high density of packets directed to the destination IP address under analysis.

4.2.2. Distributed Denial of Service Attacks

The worst distributed DDoS attack is the one that uses a large number of well-distributed bots, in which each bot sends traffic at a very small rate, but the aggregated volume is enough to overwhelm a server. In such cases, traffic consists of not abnormal service requests, e.g., for a Web page. So, there is no anomaly in traffic and no aggressiveness in individual bot behaviour that can help identifying and filter the attack.

Most known DDoS attacks are accomplished through a set of sources that floods a destination or limited set of destinations with small packets. Most of high intensity DDoS attacks are easily detected, just by looking selected parameters, such as the number of packets. Figure 4.5 exhibits such kind of attack, where a simple analysis of the number of packets is enough to recognize the occurrence of an anomaly. Particularly in this case, the attack was accomplished using only three different sources, sending IP packets to a single IP destination. When sending a packet, each source address used different source and destination port numbers, increasing like this the number of new IP flows.

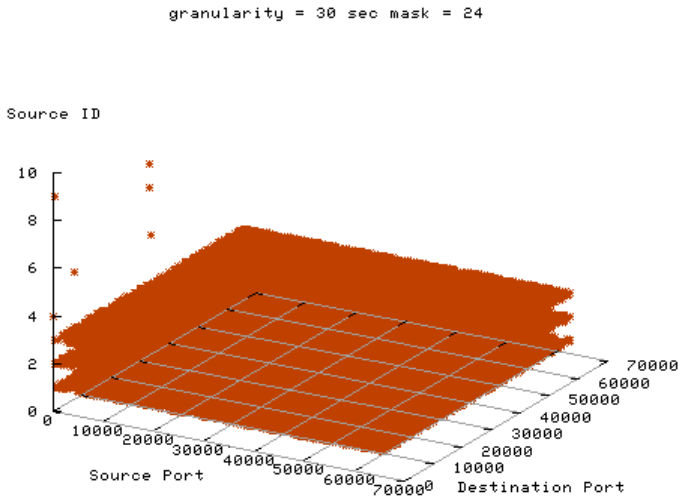

Figure 4.5: Example of a high intensity DDoS attack. High intensity attacks can be recognized just by looking at the number of packets sent.

Such usage, of continuously changing the number of the ports involved, is a confounding element, which requires deeper analyses of the flooded packets.

However, most of the current and successful attacks, currently perpetrated, are not so obvious. Those that succeed are the ones that get unperceived, until they start disturbing the access to network services, and then the application of countermeasures is ineffective. Of course, the detection of such attacks cannot rely on a simple inspection in the number of packets sent to a destination, because they would pass unperceived. Beside the detection of

low intensity DDoS attacks, NADA also differentiates three types of such attacks, according to the number of IP addresses and port involved. As their name suggests, DoS/DDoS attacks may involve more than one source, and have as target a single IP address or a restricted set of network addresses. Particularly, NADA allows the detection of attacks directed to a single IP address, or a set of addresses of the same network. Such characteristics define at some extent how the set of four plots will look like.

The next sequences of plots (Figures 4.6 and 4.7) are related with a DDoS in which a set of sources sent packets to a target address, using different destination port numbers (but the same source port number). Because this is a low intensity attack, the time granularity in the detection phase cannot exceed 30 seconds, otherwise the attack would be unperceived by NADA. However, such attack may be detected and partly classified even when considering a network mask equal /8, as it is showed by plots of Figure 4.6 – vertical line at source number 580 at plot (a) and 1200 at the other three plots. The source numbers 580 and 1200, on plot 4.6. (a) and plots 4.6 (b)(c)(d), respectively, correspond to a specific source IP address, obtained from the aggregate flow under evaluation. Particularly, source 1200 is obtained from source 580 in plot 4.6 (a), by adding the source port number information, reason why it has a higher value. Such representation is necessary to increase the clearness of the graphs, and to highlight the anomaly visual characteristics.

The consideration of flows so aggregated, as the ones with a network mask /8, may not always provide enough information to classify anomalies. Usually, at such levels of aggregation it is usual to have significant noise – collection of points that are sent to the same destination network, but from “regular” flows. When analysing larger aggregates of traffic flows, as it is the case when considering lower values for the network mask, it is frequent to detect more than one traffic anomaly. Particularly, for the set of plots presented, the DDoS under analysis is not the unique anomaly. In fact, it is possible to recognize a Network Scan at the lower IP source numbers, as well as a Port Scan, also associated to the first IP source numbers.

As stated before, the DDoS under analysis is first spotted around IP source number 580 in Figure 4.6 (a) (such classification is only possible after the observation of the other three plots). The points associated to the anomaly are interleaved with other points that do not represent an anomaly at all, and the result of such mixing is a vertical line. Such line indicates all the different destination IP addresses with which the sources communicate. When looking plots (b), (c) and (d) of Figure 4.6 it is possible to see that the same vertical line stands up, including the same set of IP sources and IP destinations. Moreover, when considering port numbers, it is evidenced that the anomaly is perpetrated using packets towards a common IP destination address, but a different destination port number. Such behaviour is a typical strategy to difficult anomaly detection – if attending the classical definition of flow by Claffy *et al.* [Claffy85] only very small flows exists. According to such definition a flow is a unidirectional traffic stream with a uniquely identifying tuple [source-IP-

address, source-port, destination-IP-address, destination-port, IP-protocol]. So, in such DDoS, whenever a packet is sent by a source, the destination port number field is changed, and each packet sent is considered as belonging to a new flow. Since, only a few packets are sent exactly with the same fields, the attack is composed of very small "inoffensive" flows.

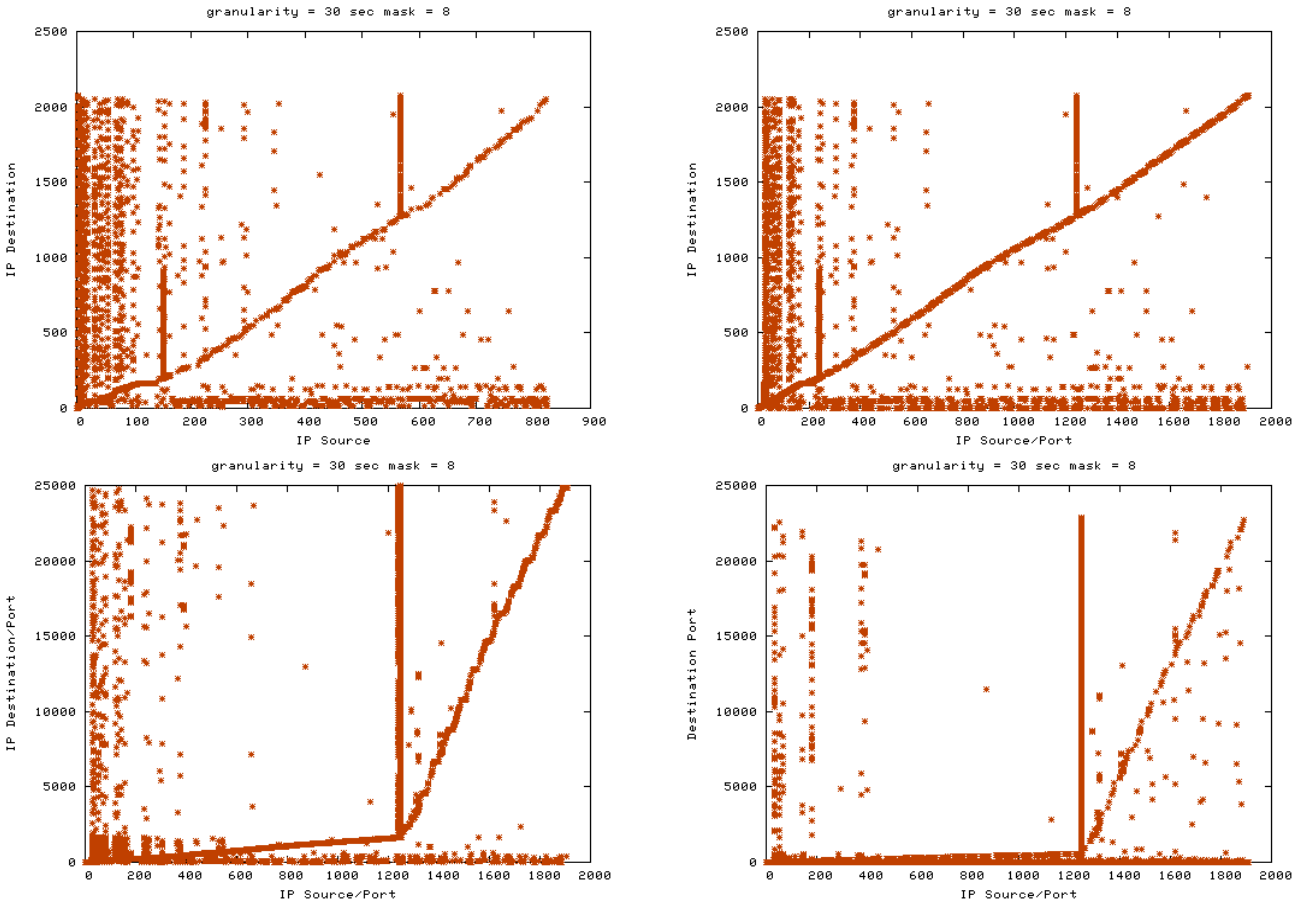

Figure 4.6: Low intensity DDoS attack of type: *n* IP source addresses x 1 IP source port → 1 IP destination address x *n* IP destination ports.

The excess of information obtained when considering lower network masks is overcome by conducting a fine-grained analysis, using higher network masks. Figure 4.7 is a zoom of the previous DDoS candidate anomaly. The analysis of the plots shows that the DDoS involves sources number 10, 11 and 12 against destination number 9. These four plots were obtained considering a network mask equal to /24. Increasing this value allows the reduction of the number of packets being analysed (lesser flows are being considered), and more accurate results are obtained. Hence, the observation of the four plots allow the recognition of one of the three patterns presented in Chapter 3 for DDoS: a restricted set of sources (in this case three) use the same source port number to flood a destination IP address, but using different destination port numbers. It is worth noticing the number of different port numbers used by attackers.

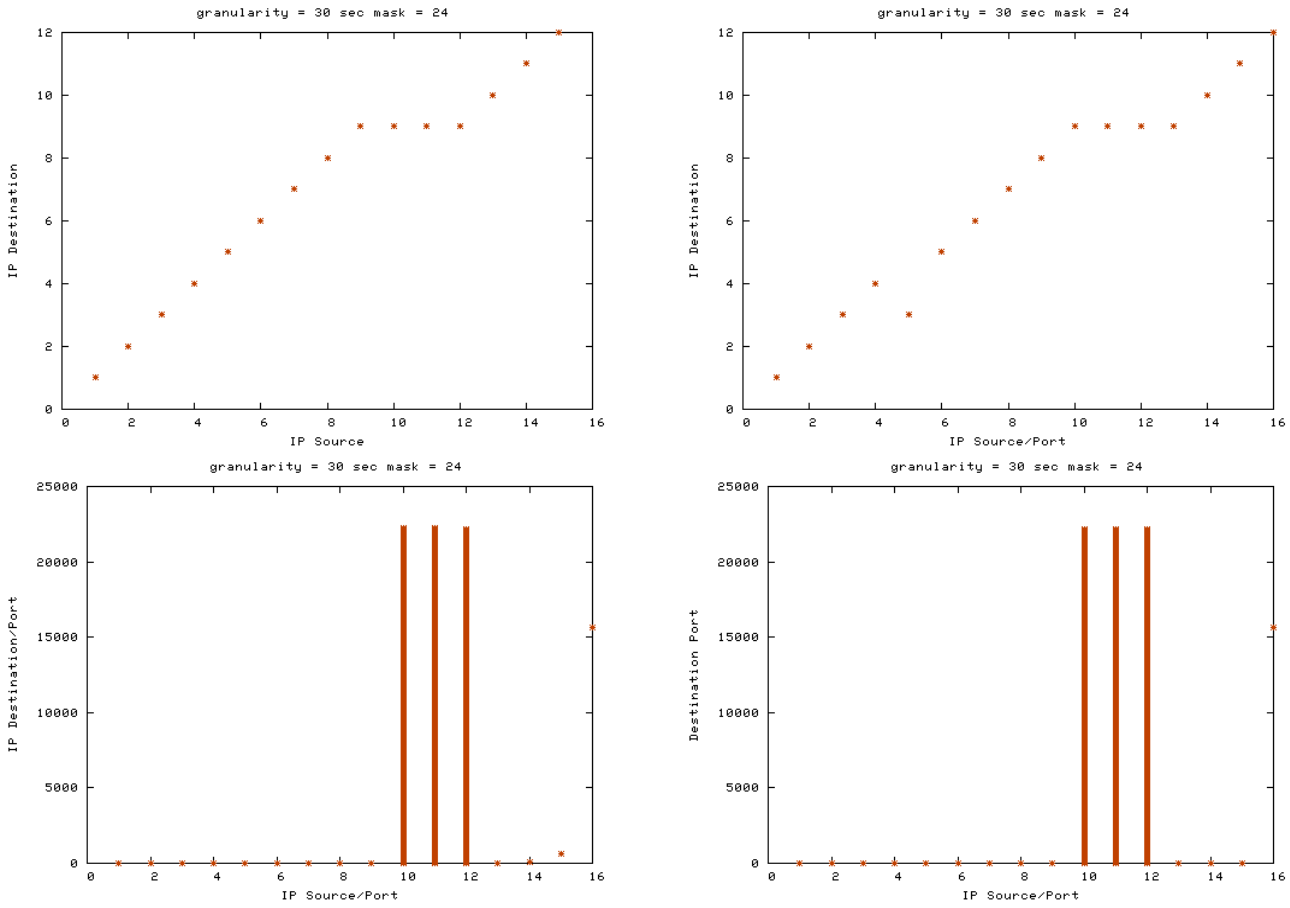

Figure 4.7: Zoom on the low intensity DDoS attack of type: *n* IP source addresses x 1 IP source port → 1 IP destination address x *n* IP destination ports. Analysis of straight flows.

The example being presented evidences the importance of the correct choice for the level of flow aggregation, when detecting certain types of anomalies. Hence, whenever looking for DDoS attacks a network mask of size /24 seems to be the most appropriate. Moreover, considering network masks higher than /24 does not bring any significant improvement to the detection and classification processes of anomalies, but increases significantly NADA's time of execution.

Following, the set of four plots of Figure 4.8 exhibits the signature pattern for a DDoS attack, in which a set of IP sources using different port numbers flood a destination IP address/port. Since this was also a low intensity attack (UDP flooding with 7% intensity), all plots were obtained considering a 30 seconds time granularity and a network mask of /24 bits. Such type of attack is easily identified by the horizontal line focused at the destination IP address that extends between the same numbers of IP sources. So, at plot (a) around 350, different IP sources (sources [50, 420]) are flooding a single destination IP address. Plots (b), (c) and (d) highlight that such sources use different port numbers when flooding the target (sources [2800, 5000]), by increasing the number of source/ports related with the attack. Also, the DDoS is not the only attack being perpetrated, in the time window being analyzed. A Network Scanning signature pattern is also identified, along with some noise.

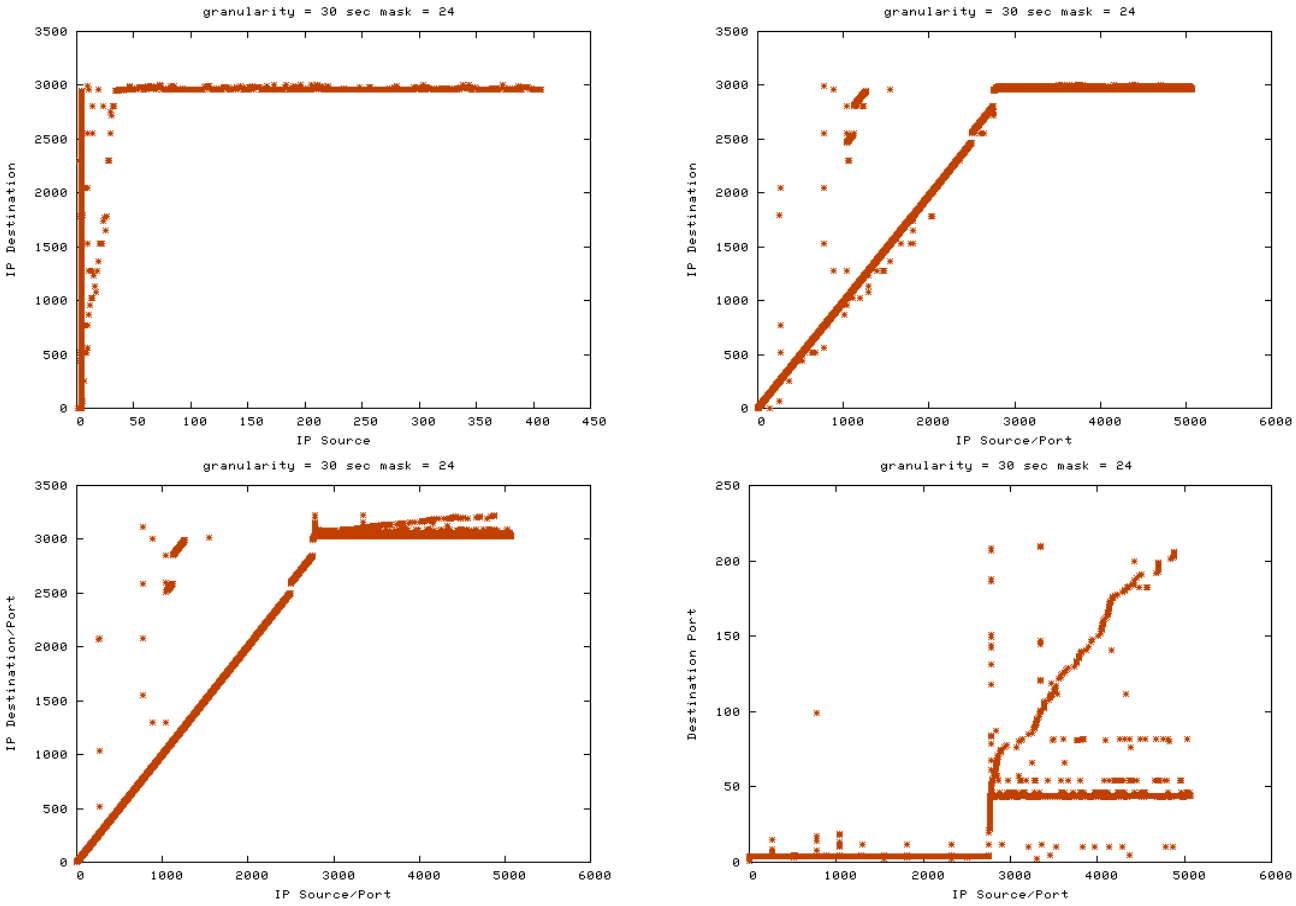

Figure 4.8: Low intensity DDoS attack of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x 1 IP destination port.

The simultaneous exhibition of two or more traffic anomalies on the same set of plots clearly reflects the requirements of security in current networks, and enhances NADA's ability of detecting and classifying simultaneously more than one anomaly.

The last example, exposed in Figure 4.9 is related with a DDoS attack in which a single destination IP address was flooded by packets from a set of sources. Flooded packets had different source and destination ports. As before, this is a low intensity attack, which involves TCP, UDP and ICMP packets, detected when the network mask was /24 and the time granularity set to 30 seconds. The usage of other parameters also allowed the detection of the attack, but hardened its classification, since the pattern was not so evident (when considering more aggregated flows). From the sequence of plots it is clear that three IP sources were involved in the attacks (points 3 to 5 in x-axis). When considering the information about the port numbers, that number rose up to almost 150, i.e., almost 150 new flows attacked the destination IP address.

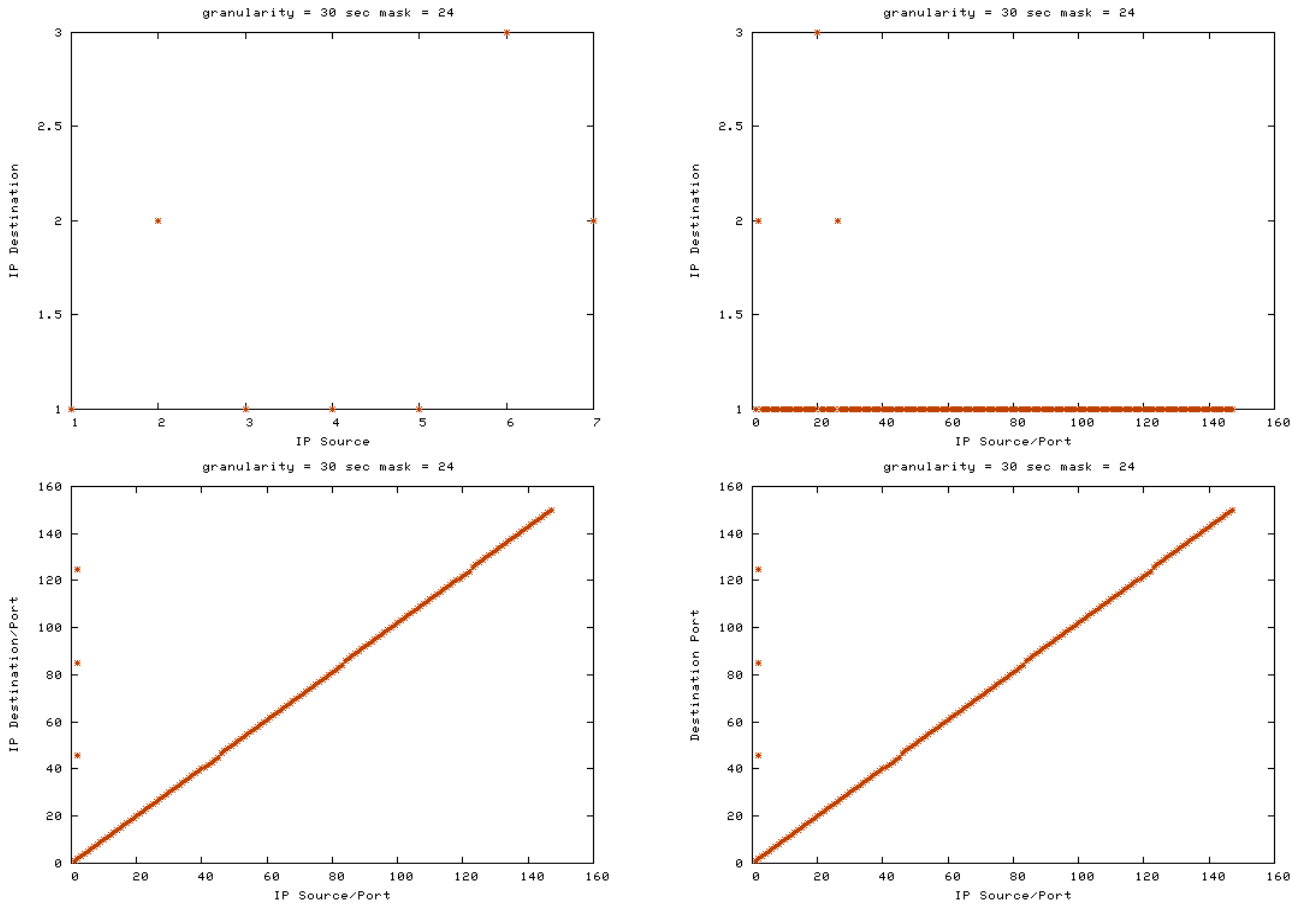

Figure 4.9: Low intensity DDoS attack of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports.

In opposition to what happens with high intensity DDoS attack, which can be easily detected and identified, by just looking at the number of packets sent to a destination, the same is not true for low intensity attacks. In such cases, the analysis must be directed to the specific IP destination, and only then some information about the DDoS attack may come out. Figure 4.10 was obtained after the classification of the DDOS anomaly, and directing the analysis to the destination IP address being attacked. Such kinds of graphs are able of revealing anomalies because they aggregate all the packets sent to a specific destination IP address, and do not deal with isolated contributions of a flow, defined by the following information (source address, source port, destination address, destination port).

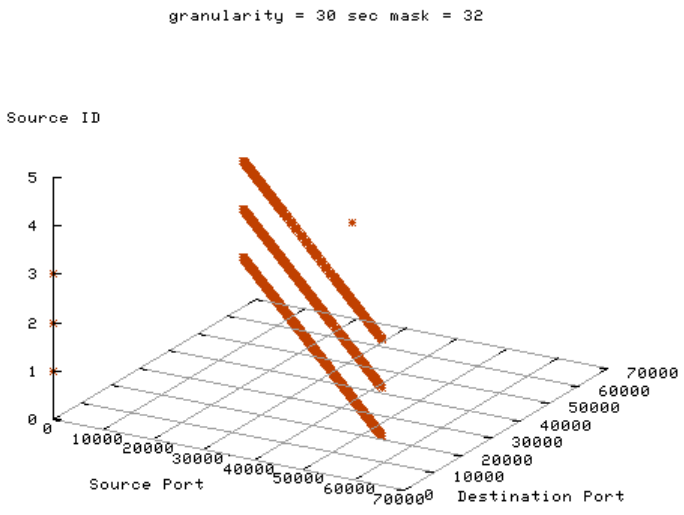

Figure 4.10: Identification of a low intensity DDoS attack of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports, counting the number of packets sent to a target destination.

4.2.3. Network Scanning

In Section 3.4.3 Network Scanning was presented as a procedure that sends small probes to many destination addresses, on a restricted set of destination ports. Such procedures *per se*, are usually harmless, since they do not involve large amounts of packets. However, they may be the first step for harmful attacks, since they may act as "spies". Experiments with scanning procedures were not explicitly conduced at the aim of the MetoSec project. However, when analyzing the anomalous traces (and even the non-documented ones) network or Port Scans were always present, motivating their inclusion and study by NADA. From the analysis of several traces, three Network Scan patterns were identified by NADA. The main difference between them is due to the port numbers used during the scanning procedure.

Plots of Figure 4.11 exhibit the pattern for the simplest Network Scan: an IP source address/port (approximately, 90 in plot (a), and 100 in the other plots) sends probes to a range of destination IP addresses. The destination port number is always de same, in this case 135. This explains why in plot (d) the anomaly representation is reduced to just one point, at IP source address/port number 100, approximately.

Figure 4.12 increases the complexity of such anomaly by sending probes from a single source, but using different port numbers. In graph (a), anomalous packets are responsible for the vertical line at IP source address number 5. On graphs (b) and (c), the anomaly is identified by the oblique line at range of IP source address/port numbers [5, 200], approximately. The oblique line then becomes horizontal in plot (d), ranging from IP source address/port number 5 to 200, approximately, since all probes were sent to the same port number. Notice, that both sets of plots allow the classification of the Network Scan considering a time scale of 60 seconds.

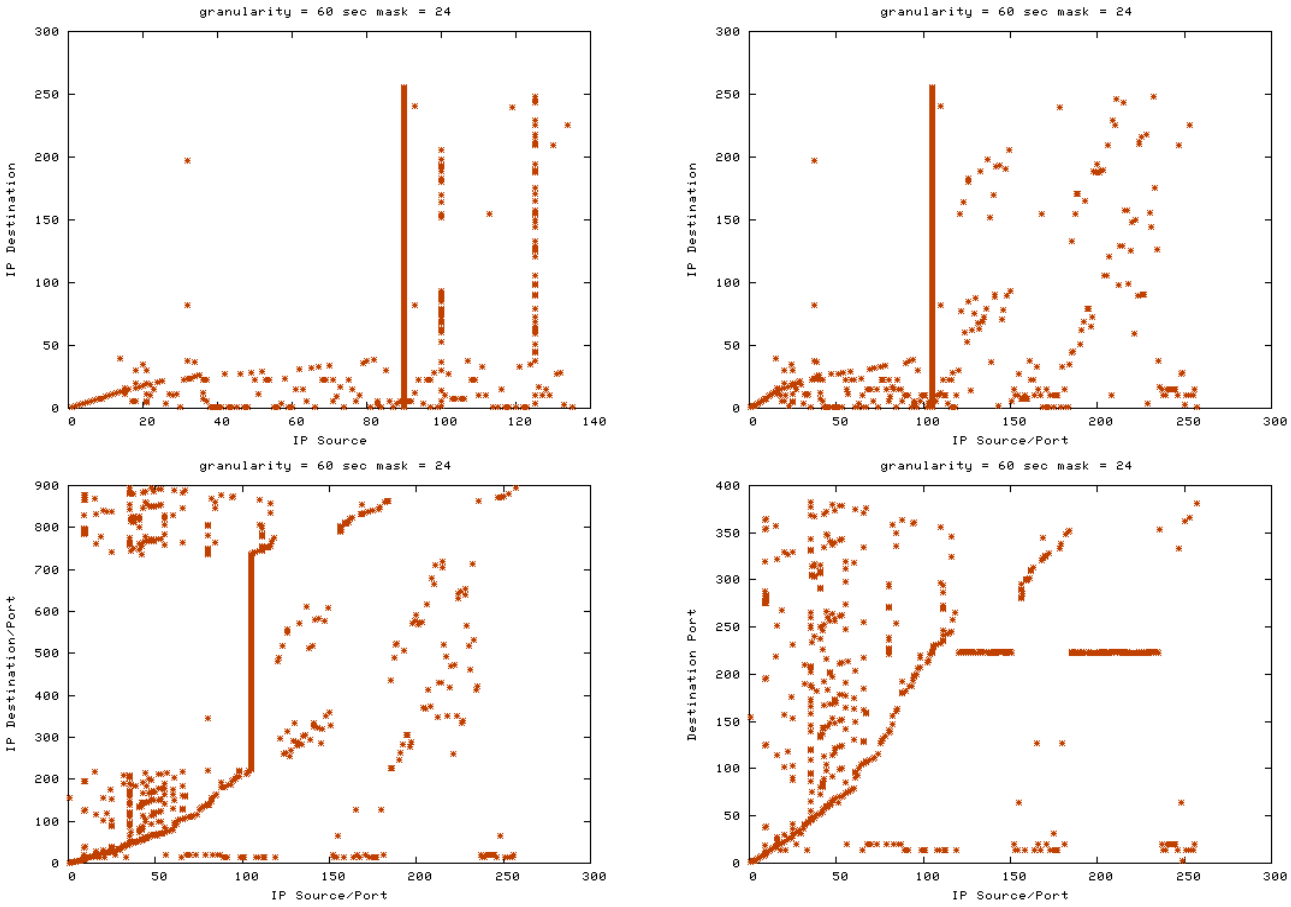

Figure 4.11: Network Scanning anomaly of type: 1 IP source address x 1 IP source port → *n* IP destination addresses x 1 IP destination port.

Plots of Figure 4.13 exhibit the third pattern of Network Scanning in which more than one IP source sends probes to a set of IP destinations. As before, probes are sent to different port numbers. So, in graph (a), the anomalous packets are responsible for the oblique line, going from IP source 0 to 40, approximately – different IP sources were used to send probes. At graphs (b) and (c), the insertion of information related with source and destination ports, respectively, do not change the profile of the graph. However, the number of points in each x- and y-axis slightly increases, meaning that the Network Scan was accomplished using a limited set of different source and destination ports. Finally, at graph (d), the Network Scan is reduced to a horizontal line at the same range of IP source numbers [0, 40], meaning that different IP sources were used to send probes to different IP destination ports.

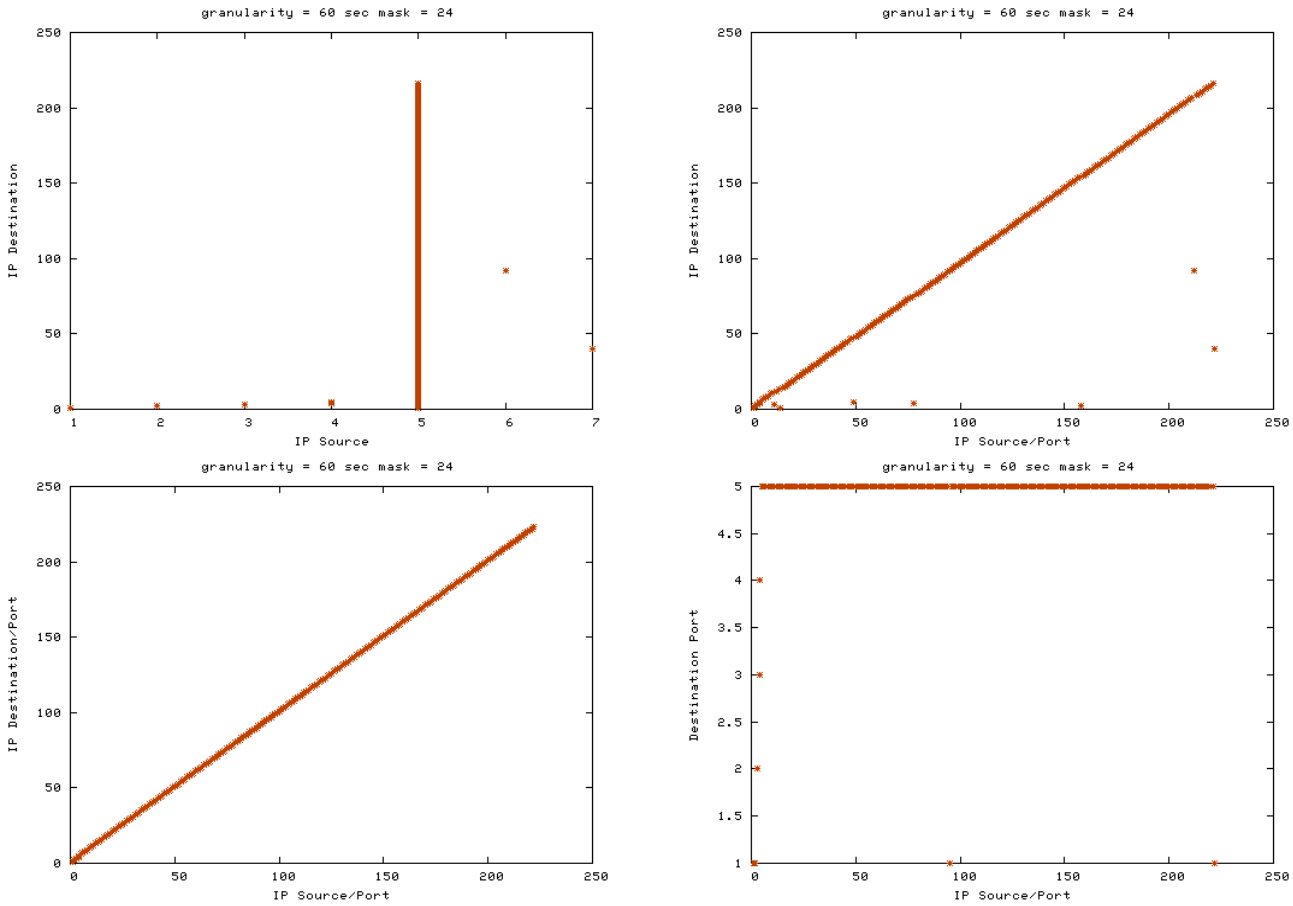

Figure 4.12: Network Scanning anomaly of type: 1 IP source address x *n* IP source ports → 1 IP destination address x 1 IP destination port.

When the main goal is the detection of Network Scanning procedures, it is very difficult to define the best time scale to detect them. The sets of plots presented above show that Network Scanning can be spotted at different time scales, and there is not enough information to relate it with the procedure intensity, as it happened with DDoS anomalies. About the appropriated flow aggregate to detect Network Scanning anomalies, it is also difficult to define a best value for the flow mask. However, due to the way that such attacks are accomplished, it is obvious that analyzing them at lower levels of aggregation (i.e., considering higher network mask) is not a good option. There would be too many small flows.

Once again, it is important to remember that Scans were not obtained through controlled experiments, as it was the case for DDoS and Flash Crowds. Network Scanning is a procedure that is permanently occurring without being noticed, which increases its threatening potential.

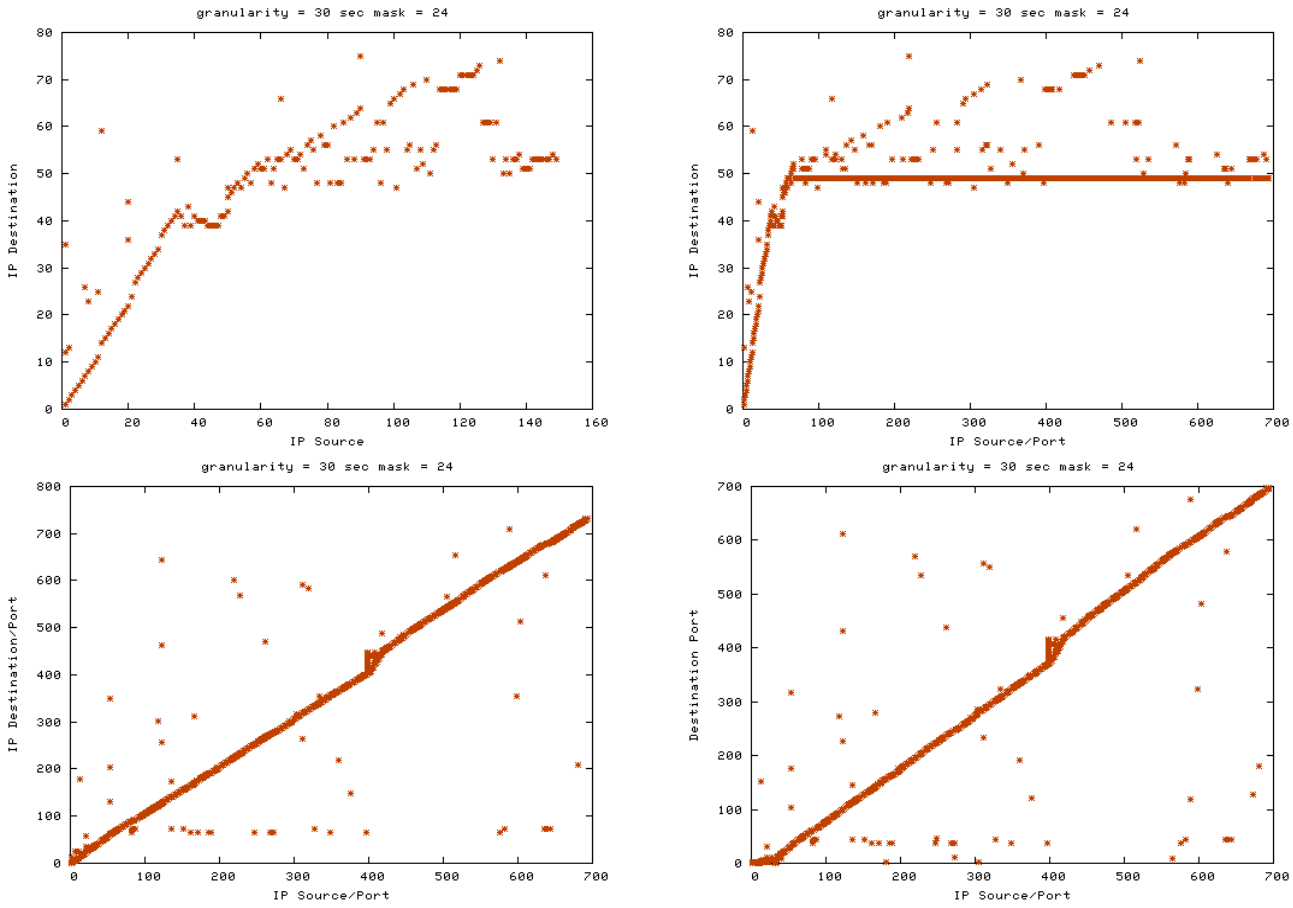

Figure 4.13: Network Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → *n* IP destination addresses x 1 IP destination port.

4.2.4. Port Scanning

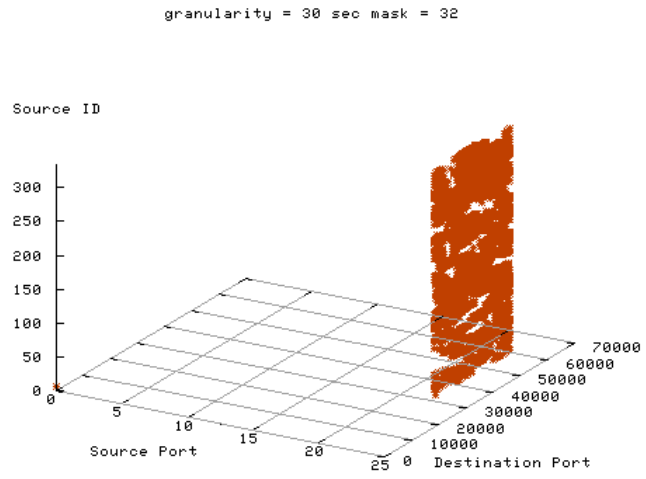

Figure 4.14: Port Scanning anomaly. The high number of destination port numbers used during the probing period is signalled by the vertical plane. Particularly in this example, several sources are being used.

Port Scanning is a searching procedure, which final goal deals with granting an illegal access to a network. As any scanning procedure, one party probes one or a reduced set of

destination IP address, for different destination port numbers. This feature differentiates a Port Scanning from other types of scanning.

So, when such type of attack is perpetrated against an IP address, the packet received by the targeted IP destination IP leads to a plot similar to the one in Figure 4.14. Particularly in this Figure, beside the change of port number at the destination side, the source also uses different port numbers with its probes (which may difficult the recognition of the Port Scan). This Figure was obtained from data collected during a brute force attack to a system.

The set of Figures 4.15 and 4.16 exhibit the same Port Scanning anomaly pattern, obtained with two different levels of aggregation: /24 and /32, respectively. Due to the significant number of different sources involved in both Port Scanning attacks, plot (a) of both Figures is not the expected one from formal signatures. Instead of a few points (that at low intensity attacks are sometimes unperceivable), a horizontal line is exhibited.

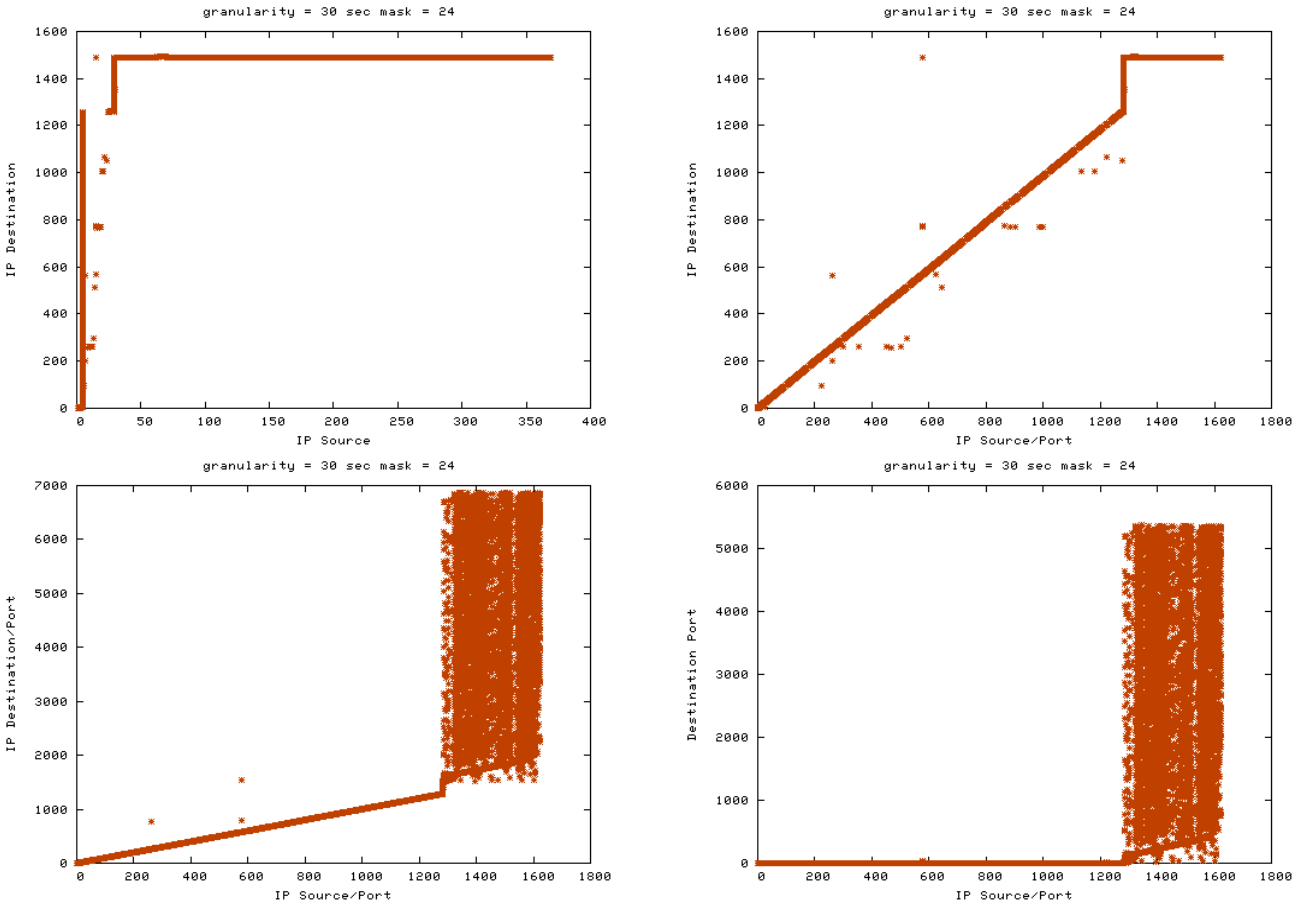

Figure 4.15: Port Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports.

Particularly, Figure 4.15 (a) shows that almost 300 sources were involved in the attack (source numbers between 20 and 350). When considering source port information, the number of IP source addresses does not change, which is corroborated by the corresponding horizontal line, that extend between source/port numbers 1300 to 1600. Taking into account the destination port numbers led to the characteristic shapes present in plots (c) and (d) – a

large number of probes are being sent to different port numbers. Figure 4.16 represents the same Port Scanning of Figure 4.15, however in this case only the targeted destination IP address is analyzed.

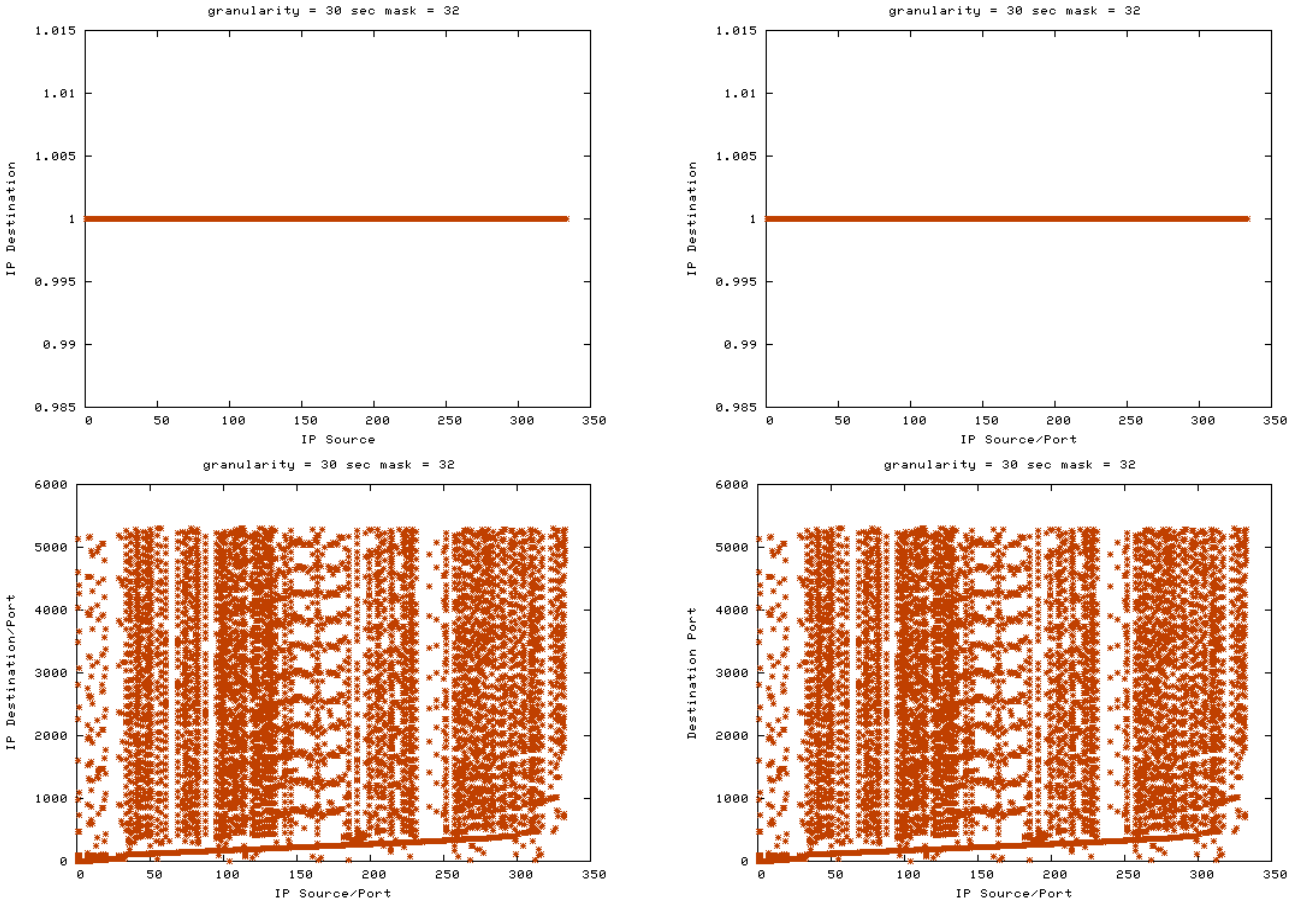

Figure 4.16: Zoom in the Port Scanning anomaly of type: *n* IP source addresses x *n* IP source ports → 1 IP destination address x *n* IP destination ports.

Port Scanning anomalies are easily identified by their characteristic vertical planes in the last two plots of the signature – those that consider destination port information. Moreover, only Port Scanning attacks led two to identical graphs (c) and (d), reason why the last plot is considered. The pattern of this type of anomaly is perfectly identifiable at different levels of aggregation, even at lower ones, as it is the case of network mask /32.

As it can be seen from the collection of plots in Figures 4.15 and 4.16, the anomaly signatures obtained with a network mask of /32 bits do not add new information to the one obtained when considering more aggregated traffic. And this is true at most of the cases. It is important to notice that the use of network mask of /32 with NADA means that the tool will look for all the sources that are sending packets to a specific destination IP address. The result will be a collection of all packets sent to that destination during a period of time. If this would not be the case, considering network mask /32 would only include one flow from one source to one destination, and no anomaly would be detected. The contribution of each individual flow is harmless, since anomalous flows have only a few packets. Only when aggregated, all the small flows become harmful!

It is interesting to notice that a closer look to the pattern signatures for DDoS and Port Scanning shows up some similarities. This is not surprising, since in some way it is correct to state that a Port Scanning is only a particular case of a DDoS, in which destination port number is the feature exploited to conduct the attack. Moreover, the distinction between these two types of anomalies is related to their subjacent motivation: attacking directly an IP destination, or analyzing the destination flaws in order to exploit them latter with specific tools.

4.2.5. Multiple Anomalies

Most of the examples being presented until now have the particularity of exhibiting one or more anomalies, beside the one under analysis. The occurrence of simultaneous anomalies is not surprising, since being connected to the Internet is more or less the same as an agreement to be attacked. This also means that NADA is then able to detect more than one anomaly at the same time. This characteristic of NADA of dealing with multiple anomalies simultaneously is a clear advantage. However, there are some cases in which the simultaneous detection of two or more anomalies is not possible. This is particularly true when the detection of specific anomalies depends on specific configuration parameters of NADA, such as the level of aggregation or the time granularity. For example, if looking only for high intensity attacks, considering higher values for time granularity seems to be a good option. However, this may limit the number of low intensity attacks that could be detected simultaneously.

Figure 4.17 exhibits two anomalies that occurred simultaneously (at least during the same time slot under analysis): a Network Scanning and a DDoS attack of high intensity (or if preferred, because of the significant number of different port numbers involved a Port Scanning). In this particular case a NLANR – Auckland trace was used. So, plots of Figure 4.17 exhibit two signature patterns. The Network Scanning is associated with the initial source numbers (leftmost), while the DDoS/Port Scanning is perfectly recognizable in the remaining part at each plot.

Looking more closely Figure 4.17, from left to right and up to down: plot (a) shows a Port Scanning anomaly, in which anomalous packets are responsible for the vertical line around IP source number 6. It also shows a DDoS anomaly, in which anomalous packets are responsible for the horizontal line between sources 30 and 120. In Plot (b), the Port Scanning anomaly is responsible for the oblique line between source/port numbers 6 and 2000, while the DDoS anomaly is responsible for the horizontal line between source/port numbers 2500 and 40000.

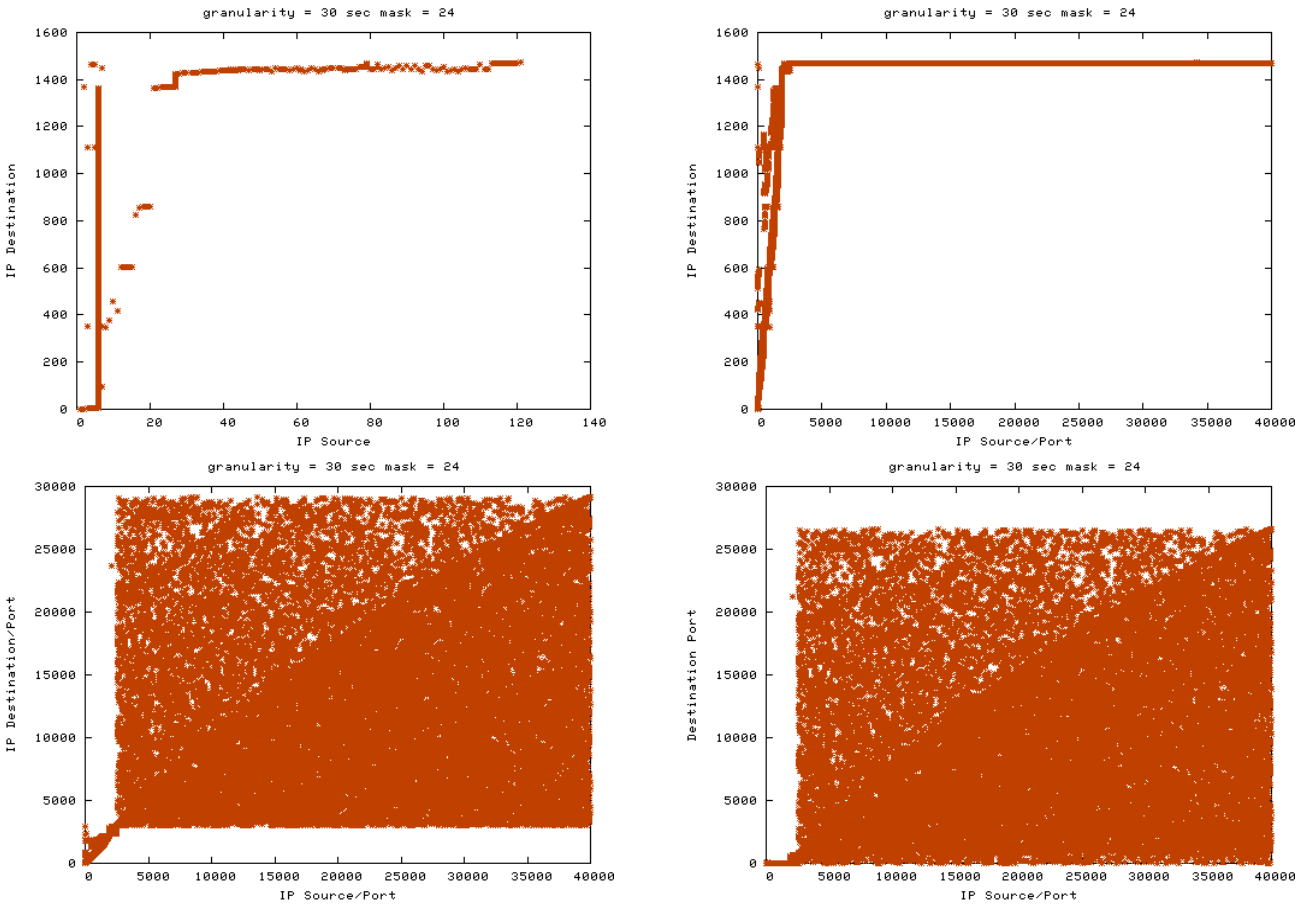

Figure 4.17: Port Scanning anomaly of type: 1 IP source address x n IP source port → 1 IP destination address x n IP destination port. DDoS attack of type: n IP source address x n IP source port → 1 IP destination address x n IP destination port.

In plot (c), Port Scanning anomalous packets are responsible for the oblique line between source/port numbers 6 and 2000. DDoS anomalous packets are responsible for the oblique line between source/port numbers 2500 and 40000. Finally, at plot (d) the Port Scanning anomaly is responsible for the horizontal line between source/port numbers 6 and 2000, while the DDoS anomaly is responsible for the horizontal line between sources numbers 2500 and 40000.

The two lower plots of the signature clearly evidence the high intensity of the DDoS attack, even increasing the difficulty of recognition of the signature pattern. The high concentration of points is due to all the small flows and packets sent to the targeted destination IP address. Almost all destination ports were used during the attack.

4.3. Conclusion

Signatures through the recognition of a pattern allow the classification of anomalies being detected. Particularly, NADA allows the definition of signatures independently of the specific tool used to generate those anomalies. Such feature allowed in a first approach, the identification of four different groups of anomalies, each with specific characteristics. This

means that for each class of anomalies (DDoS attacks, Port Scans, Network Scans) a set of characteristics is identified, related with how the class affects traffic profile. Then, such characteristics may be used in detection processes, where behaviour-profiles are searched for. With this approach, behaviours that are suspicious are defined based on historical analysis. For example, when a consecutive range of network addresses receive a small amount of data, it normally indicates something unusual such as a Network Scanning attempting to do something to a system.

All signatures being presented have a graphical shape, which result from the correlation between four IP features: source and destination addresses and ports. Each class of anomaly affects differently each of the parameters, and consequently the way they relate to each other. Four different classes of anomalies were exploited by NADA. For each class at least one signature was defined. Others have more than one signature, reflecting the diversity of strategies that a hacker may use when attacking a network. For instance, a Port Scanning anomalies rise up the number of different ports used to reach a destination IP address, while Flash Crowd anomalies are associated to the most regular pattern.

Beside the signature pattern, an important related aspect is the time scale at which the pattern is detected. As it was seen, if the class of anomaly is important when choosing appropriate values for time granularity, another important factor is the intensity of the attack. Attacks being studied range from a major impact on global traffic profile (high intensity) to attacks that are completely hidden in the global traffic (low intensity). Usually, later type requires smaller time granularities, in the order of 30 seconds or less. Low intensity attacks remain low in traffic volume, making them difficult to be detected via simple statistics such as sample mean or variance estimates. Low intensity attacks are also meaningful for NADA, since one of its main purposes is the detection of anomalies even and mostly when their intensity level remain low, i.e., before they have a negative impact on the network QoS.

Also important when analyzing the pattern of anomalies is the level of aggregation of the related flows. The four classes of anomalies identified by NADA still exhibit the same pattern when considering a network mask of /32, but reaching such level of disaggregation is time consuming. Moreover, in most of the cases there is no significant increase in the quantity and quality of the information obtained. Therefore, the increment of information that could advent from the analysis of single flows does not compensate the increase in NADA's time of execution. Network masks in the range of /20 to /24 bits seem to be suited for the majority of the anomalies. Moreover, if it is previously known that a certain network do not have a large amount of traffic, so the number of different flows will be small, even masks in the order of /8 to /16 bits may be used. Otherwise, traffic packets from other flows (be they anomalous or not) may at some extant make difficult the interpretation of results.

One last reflection is that more and more DDoS are seen as a main class of attacks, from which it is possible through some particularization/circumstances to obtain other types of

attacks. This is true for all classes of attacks analysed during this study. For example, and as it was commented above, a Port Scanning may result from several sources trying to access a large number of port numbers at one IP destination. In such case, only the basis motivation may decide which of both is occurring, or as it is considered sometimes the port numbers used – some port numbers are more “scan able”. On the other hand, if the definition of DDoS is not restricted to a single destination IP address, but extended to a set of destination IP addresses, be they in the same network or not, we are in front of a Network Scanning.

From all the examples exhibited along this Chapter 4, it was proved that NADA behaves correctly, and is able with some accuracy to classify correctly traffic anomalies. Moreover, such classification procedure is accomplished using simple mathematical approaches, which was one of the main concerns when developing the algorithm. Beside the four classes of anomalies considered along this chapter, there is an expectation that NADA could be easily upgraded in order to classify other types of anomalies or variations of the existing ones – just by analyzing the same set of plots, or by redefining the existing ones or just by introducing new simple plots!

Chapter 5

Statistical Evaluation of NADA

Whenever an algorithm is developed, it is important to know about its effectiveness. One way of doing it is through statistical evaluation. Particularly with Network Anomaly Detection Algorithm (NADA), performance assessment includes determining if traffic anomalies are detected, classified and identified correctly, as well as assessing how configuration parameters affect NADA's behaviour.

Nowadays, there are different approaches to accomplish statistical evaluation of algorithms. Despite their differences, all of them have a common denominator: they require a dataset against which the algorithms are tested. Until now, one of the most used datasets was KDD'99. However it is being widely discredited and its usage has been discouraged due to several flaws that have been pointed on it [McHugh01] [Mahoney03]. So, to guarantee the validity of NADA's statistical evaluation, the database of anomalies obtained from MetroSec project was used. More, Receive Operator Characteristic (ROC) approach was chosen to perform such evaluation.

To complete NADA's evaluation, it was also compared against two similar tools: Gamma-Farima and PHAD.

5.1. Classical Evaluation of an IDS

Intrusion Detection Systems (IDS) are concerned with the recognition of intrusion attempts against a computer or network system. To do so, IDS integrate one or more algorithms that are able of detecting irregular events, and rising up some sort of alarms to indicate that an intrusion is occurring. Such behaviour approaches NADA's behaviour.

However, as with any real system, accuracy is not always one hundred percent, and sometimes alarms are raised up inappropriately. Such occurrences may be due to incorrect parameter configuration, or simply due to limitations of the algorithms being used.

One way of evaluating an IDS performance is running the IDS against a specific dataset, counting the number of True Positives, i.e., when real anomalies are correctly detected and classed, True Negatives, i.e., when events are correctly classed as normal, False Positives, i.e., when normal events are classed as abnormal ones, and False Negatives, i.e., when anomalies are classed as normal. Relationships between those values may be showed up using the Receiver Operating Characteristic technique.

ROC technique was used for the first time, as a tool to evaluate IDS, in 1998 during an effort known as the 1998 DARPA off-line intrusion detection evaluation, at MIT's Lincoln Laboratory [MitDarpa08]. It was the first comprehensive test for multiple IDS using a realistic setting. To accomplish that, a small network was built, intending to simulate an Air Force base connected to the Internet. Background activity was produced with scripts, and attacks were injected at well defined points, and gathered with TCPDUMP. Seven weeks of this data, with a list of when and where the attacks occurred, was used to train IDS systems. Once systems were trained, two weeks of data were used to test the detection performance of the intrusion detection systems under analysis.

Various accounts of this evaluation have been published. Classical references are the work of Durst *et al.* [Durst99], Lippmann *et al.* [Lippmann00] and Lee *et al.* [Lee99]. The 1998 DARPA effort was the first that evaluated several IDS, used a wide variety of intrusions, simulated realistic normal activity, and produced results that could be shared by many researchers. After the evaluation period, researchers involved with the DARPA effort and others in the community, provided feedback on the evaluation. This resulted in changes for the 1999 evaluation, named KDD'99. Main changes comprised the use of more stealthy attacks, including a Windows NT target (and its audit logs), the definition of a security policy for the target network, and testing of more recent attacks. While the 1999 evaluation intended to address some of the concerns voiced in the IDS research community, some serious questions remained regarding its applicability.

McHugh [McHugh01] published an in-depth criticism of the evaluation, based solely on the procedures used when building the KDD'99 dataset and performing the evaluation. McHugh's primary criticism was the failure to verify if the network realistically simulated a real-world network. Another criticism was that, while the evaluation was performed to test the performance of advanced intrusion detection systems, traditional signature-based IDS were not run against the data to see how they would fare. In 2003 Mahoney and Chan [Mahoney03] decided to look more closely at the KDD'99 data itself. Mahoney and Chan's work proved that, in fact, the data did not authentically simulate real world conditions and, that even a simplistic IDS could identify and achieve better performance, than it would never

achieve in a real world environment. Also, they discovered that the dataset included irregularities, such as differences in the TTL for attacks versus normal traffic.

Unfortunately, despite the warnings about the DARPA IDS evaluation dataset, it is still being widely used in performance evaluation studies. The lack of available alternatives and new efforts to provide new datasets for evaluation, are the main cause for this persistence. However, since the publication of criticisms, reviewers tend to scrutinize more closely the use of KDD'99 datasets. And, as Brugger *et al.* [Brugger07] sustain, KDD'99 may still be useful for a first-order IDS evaluation. Given Chan and Mahoney's work, it appeared that if advanced IDS could not perform well on the DARPA dataset, it would not perform acceptably on realistic data. Such distress around KDD'99 motivated the construction of a database of anomalies, to be used when evaluating NADA – the MetroSec database, presented in Chapter 4.

In spite of all the drawbacks pointed to the DARPA initiative, it allowed the introduction of a technique to evaluate the performance of an IDS system that is still commonly used. During the 1998 DARPA evaluation, detection results were combined with the total number of network sessions to give two summary measures of an IDS's performance: detection rate (intrusions detected divided by intrusions attempted) and False Alarm rate (False Alarms divided by total network sessions). These summary measures were taken as an estimate of one point on the IDS's ROC curve.

The ROC space is defined by False Alarm rate and True Positive rate as x- and y-axis respectively, which depicts relative trade-offs between benefits and costs. The best prediction of this method yields a point in the upper left corner or coordinate (0, 1) of the ROC space. Such point signifies that all attacks were found and no False Alarms were detected. The (0, 1) point is also called a perfect classification. A completely random guess yields a point along a diagonal line, from the left bottom to the top right corners, named no-discrimination line. The diagonal line divides the ROC space in areas of good or bad classification/diagnostic. Points above the diagonal line indicate better than random classification results, while points below the line indicate meaningless results.

In the anomaly detection and classification context, a True Positive occurs whenever an anomaly occurs and it is correctly detected by the algorithm. A False Alarm counts whenever an event is marked as anomalous, but it is not one or, in the other hand, it is an anomaly but it was not detected as being one.

5.2. Performance of NADA

The ability of an algorithm to correctly detect and classify anomalies, i.e., to perform accurately, depends on the algorithm itself, but also on external factors. Common external

factors are the traces that contain the anomalies and that are used to test the algorithm, the type of anomalies being detected, their intensity and duration.

NADA's internal factors that may affect its performance include its parameters that require some level of tuning. Particularly, the performance of NADA may be affected by the time granularity and the level of flow aggregation considered, as well as the filtering parameter *k*. When executing NADA, the time granularity and the level of aggregation take different values, as a mean to assure that anomalies are detected independently of their intensity and duration. Such behaviour controls the effect that those parameters, could have on NADA's capacity of detecting anomalies.

The same does not apply for the filtering parameter *k*. Its value is assigned at the beginning of NADA's execution and stands the same along all the execution time. Remember that smaller values of *k* correspond to less significant changes, and in the other hand, higher values of *k* allow the detection of anomalies having a large impact on traffic.

5.2.1. ROC Curves

The problematic of anomaly detection and classification deals with algorithms that claim to be able of marking and recognizing all anomalies without False Alarms. The evaluation of such problematic using documented traces is a simple way of obtaining accurate results. So, for each documented trace, one needs to know exactly the number and type of anomalies on it, as well as the location of such anomalies (for example, the identification of packets involved or the time slots of occurrence). Then, the detection algorithm under evaluation detects the correct number and type of anomalies or not! Therefore, the use of documented traces allows the calculation of True Positive and False Alarm rates, i.e., to plot true ROC curves. To calculate True Positive rates one needs to know the number of True Positives and the number of False Negatives. Documented traces are the only way of counting False Negative occurrences: it is only possible to affirm that an anomaly exists, if there is previous knowledge about it. To calculate the False Alarm rate, one needs to know the number of False Positives and of True Negatives. So, if an algorithm recognizes more anomalies of a certain type than the ones that really exist in the traffic trace, such occurrence must also be signalled.

MetroSec Dataset

Statistical evaluation of NADA was performed against the different documented traces presented in Chapter 4. This gave a total of 42 different traffic traces being analysed with at least one Distributed Denial of Service (DDoS) attack per trace (some traffic traces contained four low intensity attacks). A total of 6 different traffic traces with one Flash Crowd anomaly included, were also analysed. As stated then, the documented traces at the MetroSec database could be grouped according to the tool used to accomplish the attacks/anomalies, and in each group, traces could be differentiated according to the intensities and durations

of attacks. Such differentiation is important, since it allows the measurement of detection algorithms sensitivity – i.e., its capacity to detect anomalies (which would not be accomplished by using the KDD'99 dataset – it only allows binary detection).

Anomalies intensity and duration are two characteristics that have a significant impact on detection algorithms. While the detection of high intensity anomalies is well done by most of algorithms, the same is not true when low intensity attacks are the concern. Many times, such attacks pass unperceived by most of the detection algorithms. So, an appropriate method to evaluate the performance of such detection/classification algorithms is one, that is able of counting how many times the algorithm succeed or fails when executed over such traffic traces, with low intensity anomalies. If the algorithm is able of detecting anomalies, even low intensity ones, it will present a low count of False Alarms.

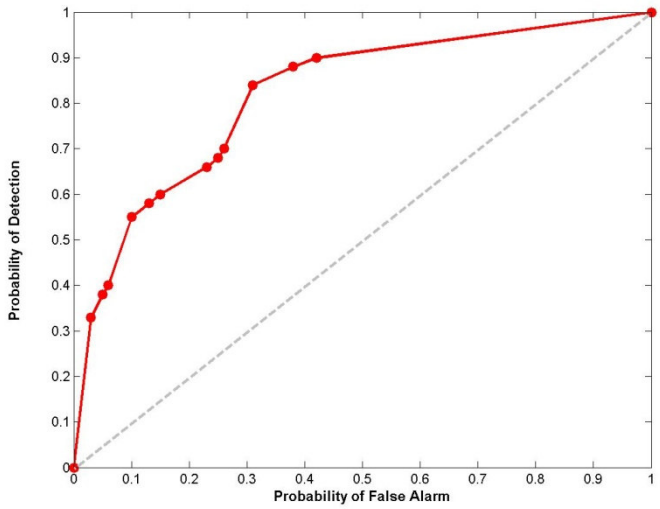

Figure 5.1: Statistical performance of NADA using the MetroSec dataset. Probability of Detection (P_D) vs. Probability of False Alarm (P_F), $P_D = f(P_F)$.

Figure 5.1 exhibits the ROC curve obtained when evaluating NADA over the MetroSec anomaly database. It relates the Probability of Detection (P_D) with the Probability of False Alarm (P_F). Each point in the curve is the mean value of all values obtained when running NADA, over each trace of the MetroSec database and fixing *k*. Otherwise, exhibiting each of the ROC curves obtained when running NADA over each traffic trace would be cumbersome.

The analysis of the curve shows that NADA behaves better than randomness, since all the points of the curve are above the diagonal line. So, even at the worst case, detection is possible with better chance than at random. Also, as the probability of detection increases, the ROC curve shows that NADA still exhibits low levels of False Alarm. For example, a P_D between 60% and 70%, has an associated False Alarm probability around 10% – 20%, which is a good result.

Due to the importance of parameter *k* in NADA's performance, it has been considered useful to expose how the probabilities of detection and False Alarm relate with the filtering

parameter. The plot $P_D = f(k)$ and $P_F = g(k)$ in Figure 5.2 shows that whatever the value of k is, the detection probability is always higher than the rate of False Alarm. Moreover, values of k around 2 (which is the most commonly used value), still have a significant probability of detection associated, while the False Alarm level remains below 20%. In some way, this increases the confidence in the assignment of value 2 to k, as a starting value when using NADA.

The values obtained are coherent with the assignments of parameter k. As k approaches 0, all the anomalies are detected, but also the number of erroneous misclassification. On the other hand, as k approaches the other side of the range, it is more difficult to detect an anomaly (at least those that are not responsible for significant variations). But, in these cases, misclassifications are almost inexistent.

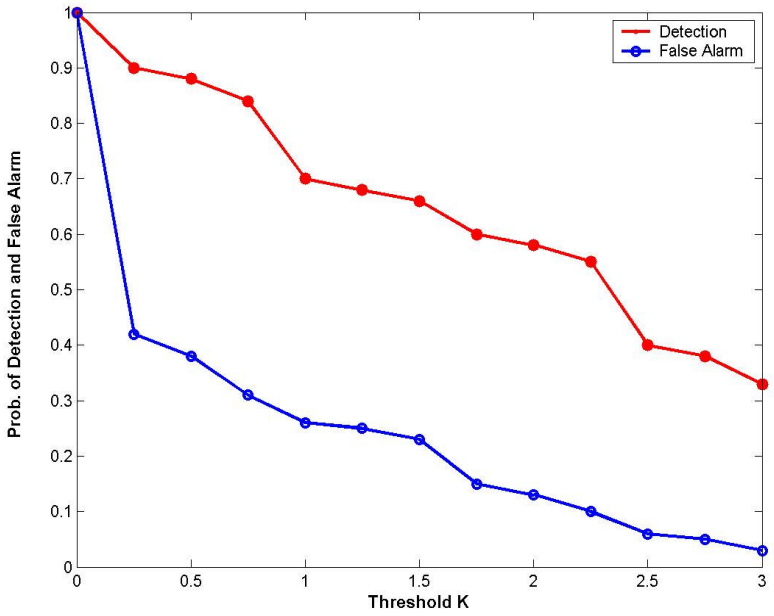

Figure 5.2: Probability of Detection (P_D) and Probability of False Alarm (P_F) vs. threshold parameter k, $P_D = f(k)$ and $P_F = g(k)$.

KDD'99 Dataset

Despite all the criticism, the KDD'99 datasets were considered when evaluating the performance of NADA, at least with the purpose of showing their ineffectiveness.

The KDD'99 intrusion detection benchmark consists of a group of datasets, which are detailed in Table 5.1. In the International Knowledge Discovery and Data Mining Tools Competition [KDD'99set], only the "10% KDD" dataset was employed for the purpose of training [Hettich99]. This dataset contains 22 types of attacks and is a more concise version of the "Whole KDD" dataset. This one contains more examples of attacks than normal connections, and the attack types are not represented equally. Because of their nature, Denial of Service attacks account for the majority of the dataset. On the other hand the

"Corrected KDD" dataset provides a dataset with different statistical distributions than either "10% KDD" or "Whole KDD", and contains 14 additional types of attacks.

Dataset	DoS	Probe	U2R	R2L	Normal
10% KDD	391458	4107	52	1126	97277
Corrected KDD	229853	4166	70	16347	60593
Whole KDD	3883370	41102	52	1126	972780

Table 5.1: Basic characteristics of the KDD'99 datasets in terms of the number of samples.

To evaluate NADA, the 10% KDD dataset was used. Several reasons favoured such choice. First, although it is the simplest dataset, it is also the most used. Second, the intention subjacent to the use of a KDD'99 dataset was to prove that it is not suited to evaluate current detection algorithms. And, finally, it could be a good first test to observe NADA's behaviour in presence of high intensity Denial of Service attacks and probes, two kinds of anomalies that NADA must deal with.

Figure 5.3 exhibits the ROC curve obtained when evaluating NADA using the 10% KDD anomaly database. Such curve relates the Probability of Detection (P_D) with the Probability of False Alarm (P_F), and its analysis shows that NADA achieved very interesting results. When using the KDD'99 dataset NADA exhibits a probability of detection near 90%, coupled with a false alarm probability of approximately 2%. Such results are really good, but unfortunately unrealistic!

The reported results indicate that denial of service attacks and probes were detected accurately, however this may be because the dataset is unrealistically simple: about 98% of the attacks in the trace were denial of service attacks of the same type, and high-intensity ones. Despite all the warnings about the DARPA IDS evaluation dataset, it is thought that it may still be useful for a first-order IDS evaluation. Given Chan and Mahoney's work [Mahoney03], it appeared that if an advanced IDS could not perform well on the DARPA dataset, it would not perform acceptably on realistic data!

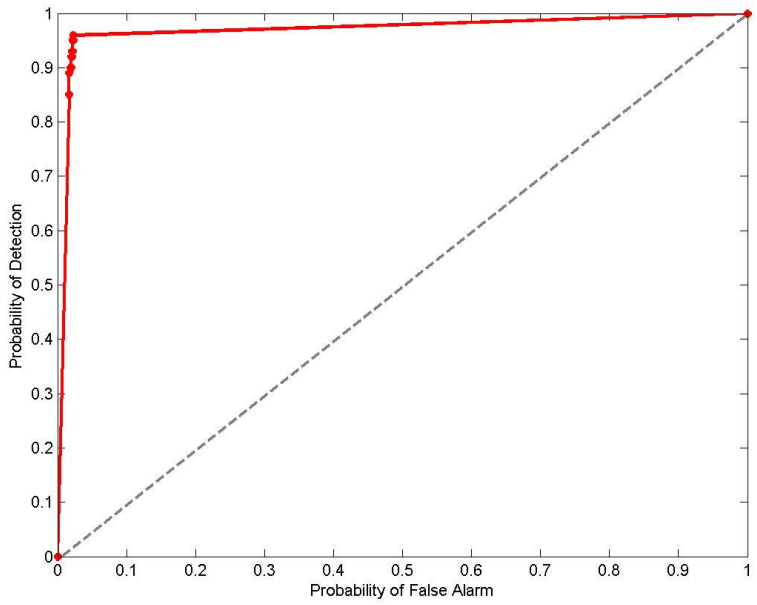

Figure 5.3: Statistical performance of NADA using the 10% KDD dataset: Probability of Detection (P_D) vs. Probability of False Alarm (P_F), $P_D = f(P_F)$.

5.3. Comparison of NADA with other IDS

The evaluation of any algorithm should include a comparison with other approaches. Particularly, for NADA, it would be interesting to compare its performance with the performance of others intrusion detection systems, moreover if such IDS could besides detect, classify anomalies.

The list of IDS that could be compared with NADA is significant. However, most of them were not available for comparison. Some of the IDS are integrated in commercial packages and not available for analysis; others have restrictions on the data format they use or hide their implementation characteristics, while others were simply impossible to access. So, comparison was accomplished against two IDS: the Gamma-FARIMA approach [Scherrer07] [Abry07], and the experimental Packet Header Anomaly Detection – PHAD approach [Mahoney01].

Gamma-FARIMA was chosen since it is an approach that has been developed in the framework of the MetroSec project, as NADA was. Moreover, Gamma- FARIMA model was evaluated using the same traces as NADA.

PHAD was selected since, beside of being fully available, it is an anomaly based IDS that is able of differentiate types of attacks, based on IP packet header features. However, due to incompatibilities between the DARPA traces (for which the IDS was originally designed) and the MetroSec traces (the DARPA traces worked with a big-endian convention and the MetroSec traces with a little-endian convention), the program had to be completely

rewritten using the PCAP API. Moreover, another tool had to be developed for replaying the traces into PHAD [Comerllato08].

5.3.1. Gamma-FARIMA Approach

The Gamma- FARIMA model, developed in the framework of the MetroSec project, uses a non-Gaussian Long Range Dependent (LRD) process to model Internet traffic aggregated data time series. Such model provides meaningful multi-resolution (i.e., aggregation level dependent) statistics to characterize the traffic with the evolution of estimated parameters, with respect to the aggregation level.

The detection procedure is based on the identification of changes in the model parameter evolution and, hence, ruptures in the statistical modelling. Therefore, the algorithm is generic and robust as it does not depend on any specific anomaly or attack production mechanisms. The detection procedure consists of computing quadratic distances between the statistics estimated from an observation sliding time window and those obtained from an a priori chosen reference window. Then, distances are thresholded to yield detections.

The model works with the mathematical description of byte or packet aggregated count processes, denoted $W_\Delta(k)$ and $X_\Delta(k)$, respectively. These consist of the number of bytes or packets, that live within the k^{th} window of size $\Delta > 0$, i.e., whose timestamps lie between $k\Delta \le t_l < (k+1)\Delta$. The main traffic characteristics that model concentrates are the joint of marginal distributions and covariance functions of $X_\Delta(k)$ or $W_\Delta(k)$.

For the modelling of the non-Gaussian marginals a Gamma distribution, $\Gamma_{\alpha,\beta}$, is used because they naturally offer a smooth and continuous evolution from exponential to Gaussian laws. And, empirical studies have shown that they are able to best capture the marginals of XΔ over a wide range of Δs. The $\Gamma_{\alpha,\beta}$ distribution is defined for positive random variable as in Equation 5.1:

$$\Gamma_{\alpha,\beta}(x) = \frac{1}{\beta\Gamma(\alpha)}\left(\frac{x}{\beta}\right)^{\alpha-1}\exp\left(-\frac{x}{\beta}\right) \qquad \textbf{Equation 5.1}$$

where $\Gamma(u)$ is the standard Gamma function.

The long-range and short-range dependencies are represented by means of a Fractionally Auto-Regressive Integrated Moving Average – FARIMA process, which allows accounting for both short and long-range dependencies. Its power spectral density or spectrum (Fourier transform of the covariance function) takes the analytical form of Equation 5.2:

$$f_X(\nu) = \sigma_\varepsilon^2 \left|1 - e^{-i2\pi\nu}\right|^{-2d} \frac{\left|1-\sum_{q=1}^{Q}\theta_q e^{-iq2\pi\nu}\right|^2}{\left|1-\sum_{p=1}^{P}\phi_p e^{-ip2\pi\nu}\right|^2} \qquad \textbf{Equation 5.2}$$

for -1/2< ν <1/2.

In this equation, parameter *d* accounts for the long-range dependence property and measures its "strength". The polynomials P and Q (found in the fraction part of the equation) can be used to fit the spectrum at high frequencies or, equally, the covariance function at fine scales, in an independent and versatile way. Therefore they model the short-range correlations. It is possible to approximate them with polynomials of degree 1, represented by the coefficients ϕ and θ.

So, modelling of the traffic is made with the 5-parameters model $\Gamma_{\alpha,\beta} - farima(\phi, d, \theta)$ and the detection of anomalies is accomplished through the computation of the mean quadratic distance of the coefficients of the model. A threshold level is defined and values under the threshold are deemed normal traffic and values above are considered anomalies.

5.3.2. Packet Header Anomaly Detection

Packet Header Anomaly Detection, or PHAD, is an anomaly-based IDS, which models normal traffic and signals, as suspicious, any deviation from this model.

In training mode, PHAD apprehends the normal/anomalous ranges of values for each packet header field at the data link (Ethernet), network (IP), and transport/control layers (TCP, UDP, ICMP). A total of 33 packet header fields are examined, mostly as defined in the protocols specifications.

Then, in detection mode, PHAD uses the rate of anomalies previously learned to estimate the probability of an anomaly. If a packet field is observed *n* times with *r* distinct values, there must have been *r* "anomalies" during the training period. If this rate continues, the probability that the next observation will be anomalous is approximated by r/n.

To consider the dynamic behaviour of real-time traffic, PHAD uses a non-stationary model, in detection mode. In this model, if an event last occurred *t* seconds ago, then the probability that it will occur in the next second is approximated by $1/t$. Because PHAD assumes that events tend to occur in bursts, during its training mode, only the first anomalous event of a burst is added to the model. So, only one anomaly is counted. This does not happen in detection mode. So, there is the need of discounting the subsequent events by the factor *t*, this is the elapsed time since the previous anomaly in the current anomalous field. Thus each packet header field containing an anomalous value is assigned a score inversely proportional to the probability of occurrence of an anomaly, as in equation 5.3.

$$scorefield = tn/r \qquad \textbf{Equation 5.3}$$

Finally, the scores of all the fields are added up to score the packet. Since each score is an inverse probability, it could be assumed that the fields are independent and could be multiplied to get the inverse probability of the packet. But packet header fields are not independent. Hence, an extension of the stationary model is used, which treats the fields as occurring sequentially. So, if all the tn/r equal, then the probability of observing *k*

consecutive anomalies in a non-stationary model is $(r/tn)(1/2)(2/3)(3/4)\cdots((k-1)/k) = (1/k)\,r/tn$. This is consistent with the score ktn/r that would be obtained by summation. Thus, the packet score is given by Equation 5.4, where anomalous fields are the fields with values not found in the training mode.

$$score_{packet} = \sum_{i \in anomalous\ fields} \frac{t_i n_i}{r_i} \qquad \textbf{Equation 5.4}$$

So, a system that ranks alarms by how unusual or unexpected they are is one that could be used to detect anomalies. More, if the assumption holds, the administrator can adjust the threshold to trade off between a high detection rate and a low false alarm rate. PHAD is based on the assumption that events that occur with probability p should receive a score of $1/p$.

5.3.3. Results

NADA, Gamma- FARIMA and PHAD are detection algorithms that aim at being generic, and not restricted to a single kind of traffic anomaly. However, all of them accomplish their goal using different approaches. According to the classification of detection algorithms in Chapter 2, all the algorithms are profile-based. On the other hand, while NADA and Gamma- FARIMA were developed and tested against an accurate anomaly database, the same is not true for PHAD. PHAD was developed and tested against the KDD'99 dataset. Despite their differences, all three-approaches claim to be effective in detecting DDoS attacks, and new types of anomalies.

The comparison of performances was accomplished using the MetroSec traffic traces, which spans different anomalies, with different intensities, low intensity attacks being of particular interest. Currently, the detection of high intensity anomalies is an easy task for most of the algorithms, but the same is not true when looking for low intensity anomalies. More, such anomalies are threatening, since a large set of attacking tools use low intensity procedures when attacking their target, and remain undetected.

Particularly, when creating the MetroSec database different tools were used to attack the targets. Some of the tools used are HPING, TRINOO, TFN2K and a version of TFN2K modified to generate bursts and ramp bursts.

PHAD

The execution of PHAD against the MetroSec traces was accomplished varying the gain *K*, which is the probability of an alarm raised by PHAD to be correct. Figure 5.4 exhibits a ROC curve, which summarizes the results obtained. Its analysis shows that PHAD performs just a little better than randomness. The statistical performance curve is parallel and very close of the diagonal corresponding to the performance of a random detection process, quite far from the y-axis which represents the ideal detection system.

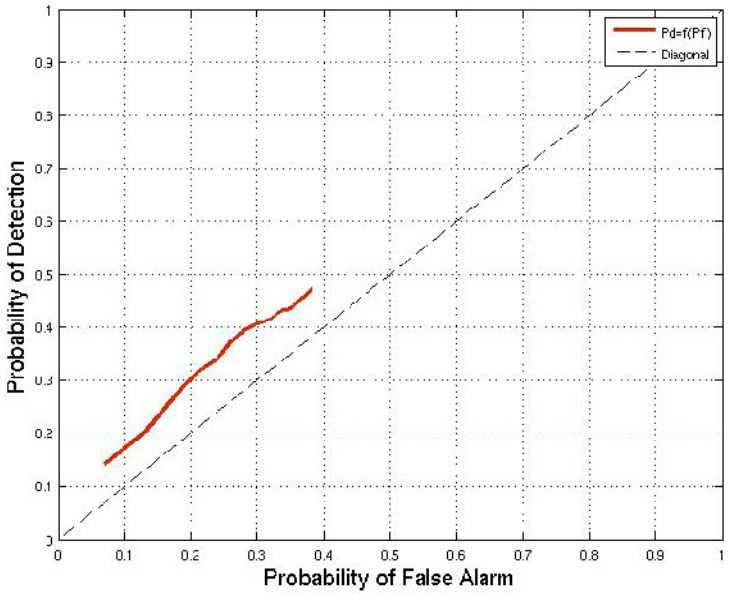

Figure 5.4: Statistical performance of PHAD using the MetroSec dataset: Probability of Detection (P_D) vs. Probability False Alarm (P_F), $P_D = f(P_F)$.

Figure 5.5 relates the probabilities of detection and False Alarm with gain *K*. It is possible to observe that as parameter *K* increases, P_D and P_F also increase in a similar way. Such behaviour suggests that PHAD performs very poorly: it detects more anomalies, but those are only False Alarms.

At lower values of *K*, P_D approaches the curve P_F, and both are not far from the x-axis. Such behaviour compromises PHAD performance, since it denotes a problem with False Alarms. Such problem was pointed in the work of Mahoney *et al.* [Mahoney03] as being common to most of detection algorithms that were evaluated using the KDD'99 dataset. Those algorithms had used the same dataset for training and evaluating data. If the dataset is a trustful representation of real networks, which happens with the MetroSec dataset, such double utilization may be allowed. Otherwise, results may be biased: the detection algorithm behaves optimally, but only for the anomalies in the dataset. Also, the curve in Figure 5.5 reflects the bad performance of PHAD when in presence of low intensity attacks: none of the attacks are detected.

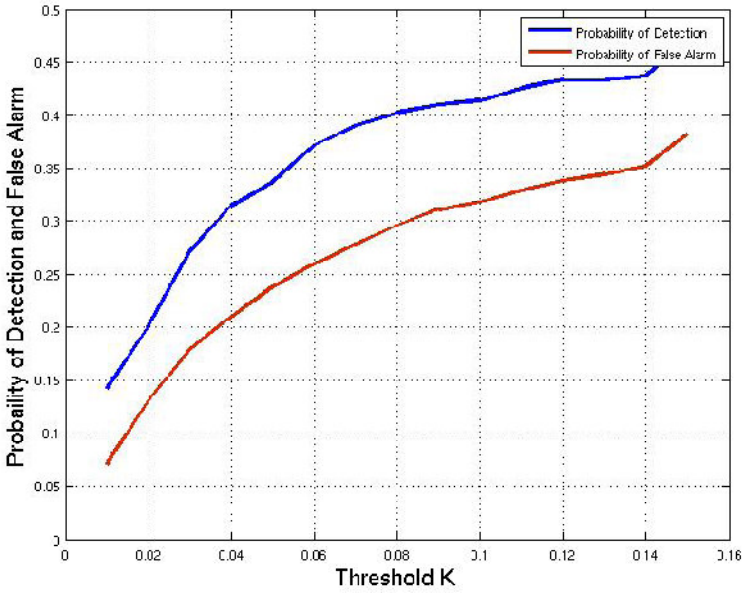

Figure 5.5: Probability of Detection (P_D) and Probability False Alarm (P_F) vs. threshold parameter *k*, $P_D = f(k)$ and $P_F = g(k)$.

Gamma-FARIMA Model

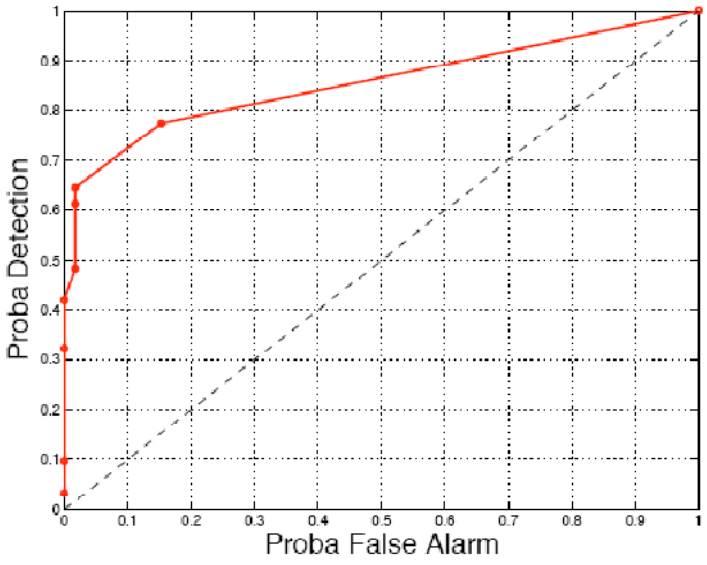

Figure 5.6: Statistical performance of Gamma-FARIMA using the MetroSec dataset: Probability of Detection (P_D) vs. Probability of False Alarm (P_F), $P_D = f(P_F)$.

The results gathered with the Gamma-FARIMA approach were obtained considering the Kullback-Leibler divergence [Kullback87] as the measure of proximity. Figure 5.6 shows the ROC curve obtained, from where one can see that the performance of the intrusion detection system is very good. More than 60% of the anomalies are detected with a False Alarm rate close to zero.

As stated above, Gamma-FARIMA execution depends on the definition of a threshold, named Lambda. Figure 5.7 shows how variations on the lambda value affect the probabilities

of detection and False Alarm of Gamma-FARIMA model. As expected, the IDS is performing very well: the False Alarm rate is always very low, being almost null, when the probability of detection is still very significant.

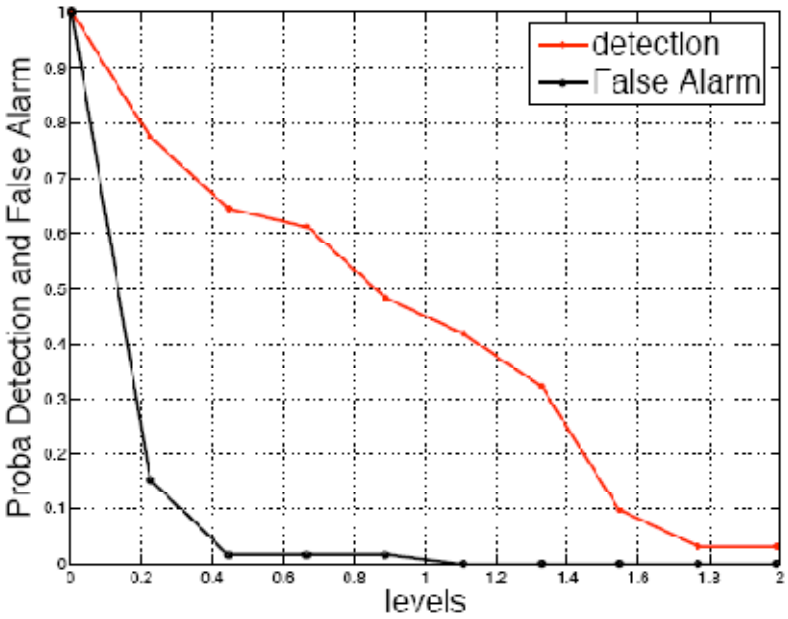

Figure 5.7: Probability of Detection (P_D) and False Alarm probability (P_F) vs. threshold parameter *k*, $P_D = f(k)$ and $P_F = g(k)$.

The observation of all three ROC curves, suggests that the Gamma-FARIMA model is the one that is performing better, against the MetroSec database. However, NADA also exhibits very good results that are not far from those exhibited by Gamma-FARIMA. Both algorithms were developed in the framework of the MetroSec project and intended to detect and classify known and unknown anomalies. Both approaches succeed, however using different approaches. While NADA uses simpler mathematical concepts, Gamma-FARIMA procedure is based on more complex mathematical procedures, which may be responsible for its performance advantage. However, the simplicity of NADA's algorithm, coupled with its accuracy may turn out in an advantage, when looking for an "easy to use" IDS approach. This is particularly true, if NADA could be coupled to an automatic process of identification of anomalous packets.

PHAD performed very poorly when evaluated against the MetroSec database (but was excellent when performing against KDD'99). Possible causes for such behaviour are the detection algorithm itself, and the anomaly database used to test the algorithm. The use of 33 different parameters and the need to assign a score to each anomalous occurrence seems to introduce some error without increasing the accuracy of detection. Moreover, as stated by the authors of PHAD, these 33 fields play only a minor role in detecting most attacks [Mahoney01].

5.4. Conclusion

Nowadays, emphasis has been put on assessing the performance of detection algorithms, which is noticed by the increasing number of related publications. However, this also enhanced the lack of appropriate datasets against which detection algorithms could be done, as well as guidelines describing evaluation procedures.

Until now, the most used dataset to evaluate the performance of detection algorithms was the KDD'99, despite all the faults that have been pointed to it. The main consequence of its use was the incorrect evaluation of detection algorithms that were considered accurate, but when working in a real network environment, exhibited low performance – they could only detect a restricted set of defined high intensity anomalies

The lack of confidence on KDD'99 dataset, and the unavailability of alternatives, leads to the creation of the MetroSec database, against which NADA was evaluated, as well as Gamma-FARIMA and PHAD approaches. In order to guarantee the quality of the MetroSec database some contingencies had to be taken into account. The main ones were to assure that only real traffic with real anomalies would be used. Only with such conditions it would be possible to guarantee the correct performance evaluation of the algorithms being tested, and extrapolate the behaviour of those algorithms to other network environments. Though artificial or simplistic this approach of creating a database of anomalies may looks like, this reference database production methodology is seen as a mandatory step for the reliable development and validation of NADA in particular, and detection mechanisms in general.

Despite its limitations, KDD'99 context positively contributed to the use of the ROC technique as a trustworthy method of measuring the performance of IDS. Moreover, the ROC technique proved to be an efficient way of comparing detection approaches. Such characteristics suited the need of evaluating and comparing NADA's performance with other IDS.

PHAD, as an example of an IDS developed under the KDD'99 context, clearly showed the importance of the anomaly dataset used when testing and evaluating detection algorithms. If the dataset is not a real representation of the network traffic and all its behaviours, its use may contribute to the development of inadequate algorithms for real traffic and anomalies. Taking PHAD as a sample for such algorithms, it was showed that they may not be able of detecting all types of anomalies, as they may claim, and that they may not perform well when considering low intensity versus high intensity detected anomalies or attacks. So, even if they are in theory able of detecting all kinds of attacks, in practice they can only detect attacks having a significant intensity. Otherwise the anomalies remain undetected suffering from a large number of False Negatives (which is certainly the worst lack for IDS).

On the other hand, NADA and Gamma-FARIMA approaches, as two algorithms developed under accurate foundations, exhibited good results in their capacity of detecting traffic

anomalies, even low intensity ones. However, despite the good results obtained during performance assessment, it is unaffordable to just conclude that one algorithm is better than the others for all applications. The integration of algorithms in complex systems may restrict their choice and use.

Chapter 6

Conclusion

This thesis addressed the problem of detection of traffic anomalies through the recognition of some of its main characteristics. The main contributions drawn from the work conducted and some issues to be addressed in future work are presented in this chapter. Section 6.1 describes the contributions and the most relevant conclusions that resulted from the work done, while in Section 6.2 some directions about the applicability of Network Anomaly Detection Algorithm (NADA) are presented. Section 6.3 contains some issues that could be addressed in future work, and would significantly improve NADA.

6.1. Main Contributions

NADA is an algorithm that intends to detect, classify and identify anomalies of any type in network traffic. NADA has been designed one step ahead of the works and results on anomaly detection previously published. Beside the integration in a single algorithm of detection, classification and identification of traffic anomalies, it was enhanced to be used with different types of traffic anomalies, and with different levels of intensities. Moreover, to complement other similar approaches, NADA provides information about the parties responsible for the anomaly, in a way easily understandable by technicians who are operating and managing networks.

In order to be efficient, and not restricted to just one type of anomalies, NADA analysis is performed simultaneously over three different axis: multi-scale, multi-criteria, and different (possibly all) IP aggregation levels. Such *modus operandi* seems to be unique, since until now nothing similar was found on the related literature. Despite this multi-dimensional analysis of traffic time series, the algorithm remains extremely affordable. In particular, the power of

NADA is related to its simplicity: data time series are screened for significant variations according to the filtering parameter *K*. More, such method allows even the detection of low intensity anomalies. Also important, is how the information provided by NADA is delivered: in graphical or textual formats, as a part of its integrated anomaly database. If the first format could be interesting for network administrators to discover, at a glance, what is happening in the network, the latter one could be easily used to trigger automatic responses suited to the anomaly that is occurring.

The effectiveness of NADA was validated in two ways, both with promising results. First, theoretical signatures of network anomalies were compared with the real signatures obtained with the execution of NADA, over real traffic traces. Thus, NADA has been evaluated based on its capability to detect known anomalies contained in documented traffic traces. Documented traffic traces were obtained during the MetroSec project, and resulted from the generation of traffic anomalies over normal operational traffic in the RENATER network. To guarantee that generated anomalies were equivalent to those perpetrated with “bad intentions”, many different parameters were considered. Particularly, the number of sources involved, the type of packet flood and the level of intensity were considered. Such concerns allowed the creation of a traffic trace database that may be used and useful to assess the capability of anomaly detection of completely different tools. Second, ROC curves were used to evaluate NADA statistically, as well its dependence on the filtering parameter *K*. Results showed that NADA behaves properly, and correct configuration of parameter *K* can lead to detection rates above 80%. More, when comparing NADA with other approaches it behaves better or at least like them, which is very promising.

However NADA is still a prototype and have some related. For instance, the parameter *k* is twofold. On one hand, it allows the control of filtering of anomalies, and on the other hand the choice of its value may be critical. Smaller values assigned to *k* may rise up the number of False Alarms, while higher values may reduce the number of anomalies detected. A proposal to assign the best value to *k* includes a pre-analysis of the traffic trace to understand the type of traffic that is being analysed. Although all, NADA represents a major progress in traffic anomaly detection, classification and identification.

6.2. Application of NADA

The knowledge that a network is being attacked and which flows are responsible for it are of great interest. Such information, with more or less extent, can be applied to different networking domains such as security, resource management, control of network congestion, or network management. However, all the work in this thesis was accomplished with two main applications in mind: Traffic Engineering (TE) and Intrusion Prevention System (IPS).

The first application deals with improving Traffic Engineering. Currently, in order to face with constant increasing of user demands and network traffic, network capacity is being

deliberately overprovisioned. Theoretically such scenario would lead to networks with virtually no data loss, even during the worst case scenario of network utilisation. However, this is not the case! In wide area end-to-end connections there are points where quality characteristics cannot be guaranteed, such as the Internet access links to an Internet Service Provider (ISP) or peering points between ISP. Those points are carefully rate limited and their capacity is controlled by Service Level Agreements (SLA) and peering arrangements, respectively. It is at these points that data packets experience delays and losses due to deliberate reductions of network bandwidth. It is at these points that the occurrences of traffic disruptions, such as anomalies, are critical.

Knowing exactly how Internet traffic behaves is a major step to define accurately how to act on traffic and improve TE functions. This involves adapting traffic routing to the network conditions, with the joint goals of good user performance and efficient use of network resources. Particularly, it would be interesting to improve TE in such a way that any decision taken would be done based on current traffic monitoring and analysis. So, it is arguable that through a deep and appropriate knowledge of Internet traffic characteristics, it is possible to define what are the major factors responsible for traffic variations at low frequencies (type of flows, rush hours, Denial of Service (DoS) attacks), and then to limit the degree of Long Range Dependence, and the range and amplitude of variations in the traffic. Particularly, it could be an option to adapt path metrics to Internet traffic flows at each moment, in such a way that traffic variations would be smoothed, LRD reduced and consequently QoS transmissions assured. Obviously, NADA perfectly suits the requirements for such implementation. It is a real-time approach, able of gathering information about major traffic variations, and the involved flows. Depending on the causes of variations, legitimate or illegitimate anomalies, the best decisions may be agreed: distribution or removal of traffic flows, respectively.

The second main application for NADA is its integration in an Intrusion Prevention System (IPS). An IPS is a computer security framework that monitors a network and/or system activities for malicious or unwanted behaviour and can react, in real-time, to block or prevent those activities. A network-based IPS, for example, will operate on-line to monitor all network traffic for malicious code or attacks. When an attack is detected, it can drop the offending packets while still allowing all other traffic to pass.

One of the major problems faced by such systems is the lack of accurate methods to distinguish between different types of anomalies. Most of the times, the line between some network attacks and legitimate traffic is blurred, which difficult fights against attacks. Moreover, as far as it is possible to know, there is no broad enough definition for attacks such as DDoS in the literature. With no definition for most common attacks, their detection by a general and automatic method is challenging. And, involving of human judgment is expensive and does not operate on the time scale of electronic packet networks.

NADA clearly faces such drawbacks, and so could be easily integrated in an IPS. In one hand, through its anomaly signature database, it is able of overcoming the problem of distinguishing between illegitimate anomalies and regular traffic. On the other hand, by assuring a continuous real-time analysis of traffic it can detect anomalies in an early stage, given time for the other IPS modules to take the correct actions. Moreover, the activity of NADA could be easily automatized using clustering algorithms [Hartigan75], which would reduce the human intervention, which is usually error prone.

Clustering definition perfectly suits the classification goal of NADA: a set of graphical plots (the clustering objects) with a specific pattern (anomaly signature) represent a specific type of anomaly (member of a group). Thus, clustering techniques could be used to identify patterns on each set of plots obtained with NADA, and by comparing them with the theoretical signatures, rise up an alarm. Among all the clustering algorithms available, K-means [MacQueen67] presents a good approach due to its simplicity.

6.3. Future Work

Network monitoring and traffic analysis are domains in continuous ongoing research, and the work developed in this thesis is just a small contribution, which can be further improved. Improvements of NADA may be accomplished in two fronts: one related with its performance, the other related with its accuracy.

Performance improvements are mainly concerned with NADA's response time. Response time is a critical factor, particularly in intrusion prevention/detection contexts. In such cases, network disruptions need to be uncovered as soon as possible, when a counter back measure is still effective. NADA, as a prototype that may be integrated in an IPS framework, still has two features that could be improved to shorten the time of response, and like this enhance its use.

The first feature is related with the filtering parameter *k*. Work should be done to automatically assign to *k* its best value according to the traffic trace under analysis, and the types of anomalies to be discovered. The second feature is related with the maximum level of IP aggregation to be considered. As it was observed, increasing the network mask may be time consuming without any return from the information gathered. So, it could be interesting to do some research on discovering the best value for the network mask to consider when running NADA.

Accuracy improvements are related with the enhancement of NADA's main tasks, for instance, the detection and classification of anomalies, and particularly the maintenance of an up to date database of anomalies. Currently, the supported database of anomalies is

complete and covers four main classes of anomalies: DoS/DDoS, Network Scans, Port Scans and Flash Crowds, each with its own signatures.

NADA is effective in detecting and classifying unexpected occurrences in one of the groups above, but it would be interesting beside the classification to look deeper on the anomalies being detected. For instance the same types of anomalies may be triggered by different means, and knowing it would allow a better response from detection/prevention systems. Such improvements could be reached by assaying other traffic parameters, variables to be included during analysis process, or even other areas of computing science.

Finally, a last improvement that could be considered is the integration of NADA in a collaborative Intrusion Detection System (IDS) framework. Collaborative Intrusion Detection System (CIDS) are becoming noteworthy since the focus of intrusion detection is moving from host intrusion detection to network intrusion detection. Such change is a response to the increase of intrusions in network environments. CIDS main goal is to find clues about where and how exactly a network intrusion occurred. To accomplish its goal CIDS uses a set of distributed agents that are able of recognizing such network disruptions. The empowerment of CIDS arises from the diversity of IDS that are used to spot irregularities, since each has its own intrinsic characteristics, strengths and limitations. Besides, as it was showed when evaluating and comparing NADA with other detection algorithms, the specificities of each approach make them more suitable than others to certain anomalies and environments. Thus making the ideal CIDS would require at each point coupling the action of several IDS, that altogether would contribute to a 100% “anomaly-invisible” network. NADA as a system that is able of simultaneously detecting, classifying and identifying anomalies fits well as an intrusion detection agent of a collaborative IDS.

Bibliography

[Abry07] P. Abry, P. Borgnat, and G. Dewaele, “Sketch Based Anomaly Detection, Identification and Performance Evaluation”, IEEE International Symposium on Applications and the Internet Workshops (SAINTW’07), January 2007

[Alessandri01] D. Alessandri, C. Cachin, M. Dacier, O. Deak, K. Julisch, B. Randell, J. Riordan, A. Tscharner, A. Wespi, and C. Wüest, “Towards a Taxonomy of Intrusion Detection Systems and Attacks”, Technical Report RZ 3366, IBM Research, Zurich Research Laboratory – Switzerland, MAFTIA project, 2001

[Almgren03] M. Almgren, E. Barse and E. Jonsson, “Consolidation and Evaluation of IDS Taxonomies”, Proceedings of the Nordic Workshop on Secure IT Systems (NordSec’03), pp. 57 – 70, Gjøvik – Norway, October 2003

[Arbor08] Arbor Networks 2008, http://www.arbornetworks.com/

[Axelsson00] S. Axelsson, “Intrusion Detection Systems: A Taxomomy and Survey”, Technical Report No 00-4. Department of Computer Engineering, Chalmers University of Technology – Sweden, March 2000

[Baldi05] M. Baldi, E. Barladis, and F. Risso, “Data Mining Techniques for Effective and Scalable Traffic Analysis”. Proceedings of the IEEE/IFIP Integrated Network Management (IM’05), pp. 105 – 118, Nice – France, May 2005

[Barford02] P. Barford, J. Kline, D. Plonka and A. Ron, “A Signal Analysis of Network Traffic Anomalies”, Proceedings of the Internet Measurement Workshop (IMW’02), Marseille – France, November 2002

[Beran94] J. Beran, “Statistics for Long-Memory Processes”, Chapman& Hall, New York, 1994

[Brugger07] S Brugger, and J. Chow, “An Assessment of the DARPA IDS Evaluation Dataset Using Snort”, Technical Report CSE-2007-1, Department of Computer Science – University of California, January 2007

[Brutlag00] J. Brutlag, “Aberrant Behaviour Detection in Time Series for Network Monitoring”, Proceedings of the USENIX LISA, New Orleans – USA, December 2000

[Cao01] J. Cao, W. Cleveland, D. Lina, and D. Sun, “Internet Traffic Tends Toward Poisson and Independent as the Load Increases”, Technical Report – Bell Laboratory, 2001

[Claffy85] K. Claffy, H. Braun, and G. Polyzos, “A Parameterizable Methodology for Internet Traffic Flow Profiling”, IEEE Journal – Selected Areas in Communications, vol. 13, pp. 1481 – 1494, October 1985

[Comerlatto08] G. Comerlatto, “Assessment of Intrusion Detection Systems for Detecting Denial of Service Attacks”, Report DAS 5511: End-of-Studies Project, Federal University of Santa Catarina – Brazil, March 2008

[Cormode04] G. Cormode, and S. Muthukrishnan, “What’s New: Finding Significant Differences in Network Data Streams”, Proceedings of the IEEE INFOCOM’04, vol. 13(6), pp. 1219 – 1232, Hong-Kong – China, March 2004

[Crovella96] M. Crovella, and A. Bestavros, “Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes”, Proceedings of the ACM SIGMETRICS’96, pp. 160 – 169, Philadelphia – USA, 1996

[DARPAset] DARPA Intrusion Detection Data Set, MIT Lincoln Laboratory, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[Debar99] H. Debar, M. Dacier, and A. Wespi, “Towards a Taxonomy of Intrusion-Detection Systems”, Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 31(8), pp. 805 – 822, April 1999

[Denning87] D. Denning, “An Intrusion-Detection Model”, IEEE Transactions on Software Engineering, vol. 13(2), pp. 222 – l232, February 1987

[Dewaele07] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho, “Extracting Hidden Anomalies using Sketch and Non Gaussian Multi Resolution Statistical Detection Procedures”, Proceedings of the ACM SIGCOMM – Workshop on Large-Scale Attack Defense (LSAD’07), Kyoto – Japan, August 2007

[Doukhan03] P. Doukhan, G. Oppenheim, and M. Taqqu, “Theory And Applications Of Long-Range Dependence”, Birkhäuser Press, pp. 687 – 716, Boston 2003

[Dunia98] R. Dunia, and S. Qin, “A Subspace Approach to Multidimensional Fault Identification and Reconstruction”, American Institute of Chemical Engineers (AIChE) Journal, pp. 1813 – 1831, 1998

[Durst99] R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo, "Testing and Evaluating Computer Intrusion Detection System", Communications of the ACM, 42(7), pp 53 – 61, 1999

[EndaceProduct08] Endace Network Monitoring, 2008 http://www.endace.com/our-products/dag-network-monitoring-cards/

[ERF08] ERF format description, 2008 http://endace.ru/pdf/Endace_DagCoprocessor_Rev_A.pdf

[Erramilli96] A. Erramilli, O. Narayan, and W. Willinger, “Experimental Queuing Analysis with Long Range Dependent Packet Traffic”, IEEE/ACM Transactions on Networking, vol. 4(2), pp. 209 – 223, 1996

[Estan03] C. Estan, S. Savage, and G. Varghese, “Automatically Inferring Patterns of Resource Consumption in Network Traffic”, Proceedings of the ACM SIGCOMM Conference, pp. 137 – 148, Karlsruhe – Germany, August 2003

[Freedictionary08] The Free Dictionary 2008, http://www.thefreedictionary.com/traffic

[Grossglauser96] M. Grossglauser, and J. Bolot, “On the Relevance of Long Range Dependence in Network Traffic”, Proceedings of the ACM SIGCOMM’96, pp. 15 – 24. Palo-Alto – USA, 1996

[Halme95] L. Halme, and R. Bauer, “AINT Misbehaving - a Taxonomy of Anti-Intrusion Techniques”, Proceedings of the 18th National Information Systems Security Conference, pp. 163 –172. Baltimore – USA, October 1995

[Hartigan75] J. Hartigan, “Clustering Algorithms (Probability & Mathematical Statistics)”, John Wiley & Sons Inc, April 1975

[Hettich99] S. Hettich, S. Bay, “The UCI KDD Archive”, Irvine – University of California, Department of Information and Computer Science http://kdd.ics.uci.edu, 1999

[Hping08] Hping, 2008 http://www.hping.org/

[Hurst51] H. Hurst, "Long Term Storage Capacity of Reservoirs", Transactions of the American Society of Civil Engineers, vol. 116, pp. 770 – 799, 195

[Hussain03] A. Hussain, J. Heidemann, and C. Papadopoulos, “A Framework for Classifying Denial of Service Attacks”, Proceedings of the ACM SIGCOMM’03, pp. 99 – 110, Karlsruhe – Germany, August 2003

[IPerf08] Iperf, 2008 http://www.noc.ucf.edu/Tools/Iperf/

[IPFIX07] IPFIX Working Group, “An IPFIX-Based File Format”, IETF Internet Draft, November 2007

[Jackson99] K. Jackson, “Intrusion Detection System (IDS) Product Survey”, Technical Report LA-UR-99-3883, Los Alamos National Lab, 1999

[Jung02] J. Jung, B. Krishnamurthy, and M. Rabinovich, “Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites”, Proceedings of the ACM WWW’02, pp. 293 – 304, Honolulu – USA, May 2002

[Jung04] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan, “Fast Portscan Detection Using Sequential Hypothesis Testing”, Proceedings of the IEEE Symposium on Security and Privacy, May 2004

[Karagiannis04] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, “A Non Stationary Poisson View of Internet Traffic”, Proceedings of the IEEE INFOCOM’04, vol. 3, pp. 1558 – 1569, Hong-Kong – China, March 2004

[KDD’99set] KDD’99 datasets, UCI KDD Archive, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[Kim04] H. Kim and, B. Karp, “Autograph: Toward Automated, Distributed Worm Signature Detection”, Proceedings of the Usenix Security Symposium, San Diego – USA, August 2004

[Kim04b] S. Kim, A. Reddy, and M. Vannucci, “Detecting Traffic Anomalies through Aggregate Analysis of Packet Header Data”, Proceedings of the Networking’04, Athens – Greece, May 2004

[Kim04c] M. Kim, H. Kang, S. Hung, S. Chung and, J. Hong, “A Flow-based Method for Abnormal Network Traffic Detection”, Proceedings of the IEEE/IFIP Network Operations and Management Symposium, Seoul – South Korea, April 2004

[Kim05] S. Kim, and A. Reddy, “A Study of Analyzing Network Traffic as Images in Real-Time”, Proceedings of the IEEE INFOCOM’05, vol. 1, pp. 2056 – 2067, Miami – USA, March 2005

[Kim06] H. Kim, J. Na, and J. Jang, “Network Traffic Anomaly Detection based on Ratio and Volume Analysis”, International Journal of Computer Science and Network Security, vol. 6(5), May 2006

[Ko97] C. Ko, M. Ruschitzka, and K. Levitt, “Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-Based Approach”, Proceedings of the IEEE Symposium on Security and Privacy, pp. 175, Oakland – USA, 1997

[Kohler02] E. Kohler, J. Li, V. Paxson, and S. Shenker, “Observed Structure of Addresses in IP Traffic”, Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, pp. 253 – 266, Marseille – France, November 2002

[Kompela07] R. Kompella, S. Singh, and G. Varghese, “On Scalable Attack Detection in the Network”, IEEE/ACM Transactions on Networking (TON), vol. 15(1), pp. 14 – 25, February 2007

[Krishnamurthy04] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, “Sketch-Based Change Detection: Methods, Evaluation, and Applications”, Proceedings of the USENIX Internet Measurement Conference, Miami – USA, October 2003

[Kullback87] S. Kullback, “The Kullback-Leibler distance”, The American Statistician, vol. 41, pp. 340 – 341, 1987

[Lakhina04] A. Lakhina, M. Crovella, and C. Diot, “Characterization of Network-Wide Anomalies in Traffic Flows”, Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC’4), pp. 201 – 206, Sicily – Italy, 2004

[Lakhina04b] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing Network-Wide Traffic Anomalies”, ACM SIGCOMM Computer Communication Review, vol. 34(4), pp. 219 – 230, August 2004

[Lakhina05] A. Lakhina, M. Crovella, and C. Diot, “Mining Anomalies using Traffic Feature Distributions”, ACM SIGCOMM Computer Communication Review, vol. 35(4), pp. 217 – 228, August 2005

[Larrieu03] N. Larrieu, and P. Owezarski, “TFRC Contribution to Internet QoS Improvement”, Proceedings of the International Workshop on Quality of Future Internet Services (QoFIS), pp. 63 – 72, Stockholm – Sweden, October 2003

[Lee99] W Lee, S Stolfo, and K Mok, “Mining in a Data-Flow Environment: Experience in Network Intrusion Detection”, Proceedings of the ACM International Conference on Knowledge Discovery & Data Mining (KDD’99), pp. 114 – 124, San Diego – USA, 1999

[Leland91] W. Leland, and D. Wilson, “High Time-Resolution Measurement and Analysis of LAN Traffic: Implications for LAN Interconnection”, Proceedings of the IEEE INFOCOM’91, vol. 31, pp. 1360 – 1366. April 1991

[Leland93] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, “On the Self-Similar Nature of Ethernet Traffic”, ACM SIGCOMM Computer Communication Review, vol. 23(4), pp. 183 – 193, 1993

[Lippmann00] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and Zissman, “Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation”, DARPA Information Survivability Conference and Exposition, vol. 2(2000), pp. 12 – 26, 2000

[MacQueen67] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations”, Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley – University of California Press, vol. 1, pp. 281 – 297, 1967

[Mahajan02] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, “Controlling High Bandwidth Aggregates in the Network (Extended Version)”, ACM SIGCOMM Computer Communication Review, Vol. 32(3), pp 62 – 73, July 2002

[Mahoney01] M. Mahoney, and P. Chan, “PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic”, Technical Report CS-2001-04. Department of Computer Sciences – Florida Institute of Technology, 2001

[Mahoney03] M. Mahoney and, P. Chan, “An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection”, Recent Advances in Intrusion Detection (RAID 20003), pp. 220 – 237, September 2003

[Mandelbrot69] B. Mandelbrot, and J. Wallis, "Computer Experiments with Fractional Gaussian Noises – Part 1: Averages and Variances", Water Resources Research, vol. 5, pp. 228 – 267, February 1969

[Mandelbrot82] B. Mandelbrot, “The Fractal Geometry of Nature”, W.H. Freeman and Company, New York, 1982

[Mao06] Z. Mao, V. Sekar, O. Spatscheck, J. Merwe, and R. Vasudevan, “Analyzing Large DDoS Attacks Using Multiple Data Sources”, Proceedings of the ACM SIGCOMM Workshop on Large-Scale Attack Defense”, pp. 161 – 168, Pisa – Italy, 2006

[McHugh01] J. McHugh, “Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory”, ACM Transactions on Information and System Security, vol. 3(4), pp. 262-294, 2001

[MetroSec08] MetroSec Project 2008, http://www.laas.fr/METROSEC/

[MitDarpa08] MIT Lincoln Laboratory, 2008 http://www.ll.mit.edu/mission/communications/ist/corpora/ideval

[Neumann99] P. Neumann, and P. Porras, “Experience with EMERALD to Date”, Proceedings of the USENIX/Workshop on Intrusion Detection and Network Monitoring, pp. 73 – 80, Santa Clara – USA, April 1999

[Niven73] L. Niven, “The Flight of the Horse”, pp. 99 – 164, 1973

[NLANR08] NLANR homepage, 2008 http://www.nlanr.net/

[OSCAR08] OSCAR – Overlay network Security: Characterization, Analysis and Recovery project http://www.laas.fr/laas/1-6953-Projets-OLC.php

[Owezarski04] P. Owezarski, and N. Larrieu, “Internet Traffic Characterization – An Analysis of Traffic Oscillations”, Proceedings of the IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC'04), Toulouse – France, July 2004

[Owezarski05] P. Owezarski, "On the Impact of DoS Attacks on Internet Traffic Characteristics and QoS", Proceedings of the IEEE International Conference and Computer Communications and Networks (ICCCN’05). San Diego – USA, October 2005

[Papagiannaki03] K. Papagiannaki, R. Cruz, and C. Diot, “Network Performance Monitoring at Small Time Scales”, Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC’03), pp. 295-300, Miami – USA, 2003

[Park96] K. Park, G. Kim, and M. Crovella, “On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic”, Proceedings of the IEEE International Conference on Network Protocols (ICNP’96), pp. 171, 1996

[Park97] K. Park, G. Kim, and M. Crovella, “On the Effect of Traffic Self-similarity on Network Performance”, Proceedings of the SPIE International Conference on Performance and Control of Network Systems, November 1997

[Park00] K. Park, and W. Willinger, “Self-Similar Network Traffic and Performance Evaluation”., John Wiley & Sons, Inc. New York – USA, 2000

[Paxson99] V. Paxson, “Experiences learned from Bro”., Usenix Associate Magazine, pp. 21 – 22, September 1999

[Robinson95] P. Robinson, “Gaussian Semi-Parametric Estimation of Long-Range Dependence”, The Annals of Statistics, vol. 23, pp. 1630 – 1661, 1995

[Roesch99] M. Roesch, “Snort – Lightweight Intrusion Detection for Networks”, Proceedings of the 13th USENIX Conference on System Administration, pp. 229 – 238, Seattle – USA, November 1999

[Ryu96] B. Ryu, and A. Elwalid, “The Importance of Long Range Dependence of VBR Video Traffic in ATM Traffic Engineering: Myths and Realities”, ACM SIGCOMM Computer Communication Review, vol. 26(4), pp. 3 – 14. October 1996

[Sabhnani04] M. Sabhnani, and G. Serpen, “Why Machine Learning Algorithms Fail in Misuse Detection on KDD Intrusion Detection Data Set”, Journal of Intelligent Data Analysis, 2004

[Schechter04] S. Schechter, J. Jung, and A. Berger, “Fast Detection of Scanning Worm Infections”, Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID’04), Sophia Antipolis – France, September 2004

[Scherrer07] A; Scherrer, N. Larrieu, P. Owezarski, P. Borgnat and P. Abry, “Non Gaussian and Long Memory Statistical Characterizations for Internet Traffic with Anomalies”, IEEE Transaction on Dependable and Secure Computing, vol. 4(1), pp 56 – 70, January 2007

[Snort08] SNORT Organization, 2008
http://www.snort.org

[Taylor01] C. Taylor, and J. Alves-Foss, “NATE: Network Analysis of Anomalous Traffic Events – a Low-Cost Approach”, Proceedings of the ACM workshop on New Security Paradigms, pp. 89 – 96, Cloudcroft – USA, September 2001

[TFN2k08] TFN2k Tool, 2008
http://vil.nai.com/vil/content/v_10535.htm

[Thottan03] M. Thottan, and C. Ji, “Anomaly Detection in IP Networks”, IEEE Transactions – Special issue of Signal Processing in Networking, pp. 2191 – 2204, August 2003

[Trinoo08] Trinoo Tool, 2008
http://service1.symantec.com/sarc/sarc.nsf/html/W32.DoS.Trinoo.html

[Veitch99] D. Veitch, and P. Abry, “A Wavelet Based Joint Estimator for the Parameters of LRD”, IEEE Transactions – Special issue on Multiscale Statistical Signal Analysis and its Applications, vol. 45(3), April 1999

[Wikipedia08] Port Scanner 2008
http://en.wikipedia.org/wiki/Portscanning

[Wikipedia08b] Flash Crowd 2008
http://en.wikipedia.org/wiki/Flash_Crowd

[Willinger97] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, “Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level”, IEEE/ACM Transactions on Networking, vol. 5(1), pp. 71 – 86, February 1997

[Whittle53] P. Whittle. “Estimation and Information in Stationary Time Series”, Ark. Mathematical Vol. 2, pp. 423 – 434, 1953

[Zhang04] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund, “Online Identification of Hierarchical Heavy Hitters: Algorithms, Evaluation, and Applications”, Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC’04), Taormina – Italy, October 2004

[Zhang05] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, “Network Anomography”, Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC'05), pp. 317 – 330, Berkeley – USA, October 2005

Annex A

Development of NADA Tool

NADA recursively executes a set of actions that allows the detection and identification of anomalies and the set of packets that are responsible for those anomalies, from a traffic trace of any duration and dimension. Traces can be analyzed off-line or on-line.

In the context of NADA, an anomaly is any event that is responsible for a significant variation in one of the data time series being evaluated (in the current version: number of packets, number of bytes, number of new flows and number of TCP SYN packets), during a time window. The level of variation allowed is related to the value attributed to a filtering parameter named *K*. Smaller values of *K* define a more coarse-grained filter, while higher values are associated to finer filtering.

Besides *K*, another important parameter of NADA is the time scale at which the algorithm is executed. This is important, because each anomaly has its own characteristics (e.g., type, intensity, duration, ...), and may not be detected at the same time scale. So, NADA permits the analysis of the same traffic trace at different time scales, ranging range from a millisecond to several hours, until an anomaly is found, if it exists.

At the end of its execution NADA presents a list of all anomalies detected in the trace and information related to the flows responsible for those anomalies. Such information includes source and destination addresses and ports information. NADA also produces a set of plots that altogether can be used as a visual signature for each anomaly. Currently the following categories of anomalies have a defined visual signature: DDoS, Flash Crowd, Network Scan and Port Scan.

A.1. Implementation

A.1.1. Input Data

The current version of NADA executes off-line or on-line, and accepts two different types of data formats. These are the ERF and the OSCARFIX format, each described in Table A.1.

Format	Description
ERF	Format of data files provided by DAG cards. Each element of this file is a packet time-stamped, with IP header information. It may also contain the packet payload.
OSCARFIX	Proprietary format developed in the context of OSCAR project. The OSCARFIX format follows the IPFIX format, and aggregates data packets in flows per unit of time. A flow is a sequence of packets
PCAP	Format associated to TCPDUMP data traces. Could be obtained from the ERF format using a format converter.

Table A.1: Input data formats recognized by NADA.

A.1.2. Developed Functions

The off-line and on-line versions of NADA have a common part, which is the core of the algorithm. Such part is composed by a set of functions which are called from a bash file named NADA_PROJECT.bash. Such functions may differ slightly depending on the type of data input.

Two groups of functions may be distinguished. The group of functions that is devoted to the anomaly detection analysis, and the group of functions responisble for the identification and classification of the anomalies. The executable versions of all functions of the first group are obtained through the execution of a Makefile available with the set of applications.

The executables of all other C functions (classification and identification family set) are obtained by entering the following command, once per function:

```
gcc –lm –o <final name of executable> <name of function.c>
```

***scanSlot*()**

Given a traffic trace with ERF format, the application *scanSlot*() is responsible for the localization of the time windows that may contain traffic anomalies. Obviously, considering lower values of K increases the probability of detecting significant variations in a time slot (which is not synonym of anomaly, at this stage of execution).

The execution of *scanSlot*() requires the set of parameters described in Table A.2, as input.

Input	Description
<name trace>	Path and filename of ERF traffic trace.
<data series>	Name of file where data time series will be written (packets, bytes, number of flows).
<time slots>	Name of file that will handle the time slots with suspicious variations.
<scope filter>	Value of parameter K. Usually K>0 and K<=3.
<granularity_us>	Definition of each time window in microseconds.

Table A.2: Input parameters of function *scanSlot*().

The execution of *scanSlot*() produces two files: <time slots> and <data series>.

The file <time slots > exhibits one line per each suspicious time slot, and looks like:

```
Col1   Col2   Col3   Col4   Col5   Col6   Col7
59     60     1770   1800   1      1      0
90     91     2700   2730   1      1      1
```

Columns 1 and 2 contain the identification of the time slots between which a significant variation was signaled. Columns 3 and 4 contain the time limits between which the anomalies were spotted. Columns 5, 6 and 7 contain a flag [0,1], that is 1 when a significant variation was detected in the number of packets, bytes and starting flows between time slots Col1 and Col2, respectively, and 0 otherwise.

The file <data series> gathers three time series, one per column. Column 1 contains the time measurement points in seconds, and the following columns contain the number of occurrences of each variable during that time.

```
Col1   Col2         Col3   Col4
30     14621990     28586  2
60     26559044     39498  1
90     43492484     38244  0
120    19814722     31761  10
150    21690808     35031  0
...
```

So, columns (1, 2) define the byte time series, columns (1, 3) define the packet time series, and columns (1, 4) define the new flows time series.

scanWindow()

Given a traffic trace with OSCARFIX format, the application *scanWindow*() is responsible to locate the time windows that may contain traffic anomalies. The core of this function is similar to the one presented above, except for the type of data being used. Because of this the input of this function differs slightly from the previous one.

The execution of *scanWindow*() requires as input, the set of parameters described in Table A.3.

Input	Description
<name trace>	Path and filename of OSCARFIX traffic trace.
<window us>	Size of time window in microseconds. This value is defined when creating the OSCARFIX trace.
<lim inf> <lim sup>	These variables contain the identification of the first and last time slots, respectively, to consider during the analysis. These values are of particular interest when running NADA online.
<scope filter>	Value of parameter K. Usually K>0 and K<=3.
<data series>	Name of file where data time series will be written (packets, bytes, number of flows).
<time slots>	Name of file that will handle the time slots with suspicious variations.

Table A.3: Input parameters of function *scanWindow*().

The execution of *scanWindow()* produces as with the ERF format two files: <time slots> and <data series>. These files are similar to the ones produced above, except they have one more column that corresponds to the number of SYN packets detected.

The file <time slots> exhibits one line per each suspicious time slot, and looks like:

```
Col1   Col2   Col3   Col4   Col5   Col6   Col7   Col8
1      2      30     120    0      0      1      1
2      3      120    180    1      1      0      0
```

Columns 1 and 2 contain the number of the time slots between which a significant variation was signaled. Columns 3 and 4 contain the time limits between which the anomalies were spotted. Columns 5, 6, 7 and 8 contain a flag [0,1], that is 1 when a significant variation was detected in the number of packets, bytes, SYNs and starting flows between time slots Col1 and Col2, respectively, and 0 otherwise.

The file <data series> gathers four data time series, one per column. Column 1 contains the time measurement points in seconds. The following columns contain the number of occurrences of each variable during each time interval.

```
Col1   Col2         Col3         Col4         Col5
60     91224290     117375 1384         9997
120    195292294    155450 1871         9451
180    155340384    141579 1875         11593
...
```

So, columns (1, 2) define the byte time series, columns (1, 3) define the packet time series, columns (1, 4) define the SYN time series, and columns (1, 5) define the flow time series.

scanTrace() and *scanTraceOscar*()

Functions *scanTrace*() and *scanTraceOscar*() allow the classification of the detected anomalies in a time slot, using a trace with ERF format in the first case, and OSCARFIX format in the latter one. At the end of the execution of these functions it is possible to know if the variations detected by functions *scanSlot*() and *scanWindow*() are anomalies or not. This is accomplished through an iterative process that extracts all anomalous flows detected in the time window, at all levels of IP aggregation (/8 ... /32). To execute both applications, the parameters in Table A.4 are used as input arguments.

Input	Description
<name trace>	Path and filename of traffic trace. Accepted formats are ERF and OSCARFIX.
<slot>	Number of time slot where a significant variation was signaled. Significant variations are detected between two consecutive time slots, the time slot <slot> reports to the first one.
<scope filter>	Value of parameter K. Usually K>=0 and K<=3. Same value as above.
<granularity us> <window us>	Definition of time window to use in microseconds. Same value as above.
<lim inf> <lim sup>	These variables contain the identification of the first and last time slots, respectively, to consider during the analysis. These values are of particular interest when running NADA online.

Table A.4: Input parameters of functions *scanTrace*() and *scanTraceOscar*().

Each iteration of *scanTrace*() or *scanTraceOscar*() creates three different files, one per each detected anomalous flow (a sequence of packets to a destination IP address/mask), that will be used during the classification and identification phases of NADA:

- **series**_*IPaddress_mask_window*.**txt**
- **result**_*IPaddress_mask_window*.**txt**
- **packet**_*IPaddress_mask_window*.**txt**

The files named **series**_*IPaddress_mask_window*.**txt,** as before, contain the data time series byte, packet, SYN packet and flow, related to the anomalous destination IP address/mask. The format is equal to the one of <data series> presented above.

```
Col1   Col2         Col3         Col4   Col5
60     91224290     117375       1384   9997
120    195292294    155450       1871   9451
180    155340384    141579       1875   11593
...
```

When considering the ERF format, the files **series**_*IPaddress_mask_window*.**txt** only have 4 columns, instead of 5 (the SYN packet time series is not calculated).

The files named **result**_*IPaddress_mask_window*.**txt** contain information about the anomalous flow during the time window under analysis.

```
140.93.192.0 13425040     40935  29454  2      3
Col1   Col2   Col3   Col4   Col5
65532  1      328    0      1
65531  1      328    0      1
65529  2      656    1      1
65527  3      984    0      0
...
```

The first line of this file has the following information: destination IP address, number of bytes to destination IP address, number of packets to destination IP address and number of different destination ports at destination IP address. If using the *scanTraceOscar*() function two additional values are inserted: the number of SYN packets and the number of new flows.

Each of the following lines has at column 1 the destination port, at column 2 the number of packets received at that port, and at column 3 the number of bytes received at that port. If using the *scanTraceOscar*() function columns 4 and 5 are inserted: the number of TCP SYN packets sent to this port and the number of new flows.

The files named **packet**_*IPaddress_mask_window*.**txt** contain information about the flow packets – one line per packet. The format of these files depends on the function. Hence, for function *scanTrace*() each line contains at Col1 the second at which the packet was stamped, at Col2 and Col3 the source IP address and port number, at Col4 and Col5 the destination IP address and port number, and at Col6 size of the packet in bytes.

```
Col1   Col2          Col3   Col4                Col5   Col6
1743   61.186.179.351484   140.93.192.131      1434   404
1744   193.206.136.21484   140.93.192.72       1434   84
1754   221.186.70.501152   140.93.192.79       1434   404
1757   65.150.79.81 1280   140.93.192.126      135    48
1759   219.153.2.1761044   140.93.192.54       1434   404
...
```

For function *scanTraceOscar*(), because the measurement unit is a flow identified by its source and destination IP addresses and ports, files of type **packet**_*IPaddress_mask_window***.txt** have the following format:

```
Col1   Col2           Col3   Col4                Col5   Col6   Col7   Col8   Col9
1743   61.186.179.35  1484   140.93.192.131      1434   6      404    2      1
1744   193.206.136.2  1484   140.93.192.72       1434   6      840    5      0
1754   221.186.70.50  1152   140.93.192.79       1434   6      4040   167    10
1757   65.150.79.81   1280   140.93.192.126      135    17     4848   200    0
1759   219.153.2.176  1044   140.93.192.54       1434   17     404    2      0
...
```

Col1 has the timestamp of the flow. Col2, Col3, Col4 and Col5 have information about the source and destination IP addresses and port. Col6 has information about the type of protocol (1-IMCP, 6-TCP and 17-UDP). Finally, Col7, Col8 and Col9 have information about the total bytes, packets and SYN packets associated to the flow.

Besides the output files generated by *scanTrace*() and *scanTraceOscar*() functions and presented above, another file generically named **AnomalousIP**_*slot***.txt** is created. Only one file of this type is created, and it allows linking a specific flow to the associated significant variations. Only anomaly-candidate flows are listed. For instance, the format of such file is as follows.

```
Col1   Col2   Col3   Col4   Col5   Col6          Col7
1      1      0      0      1      135.0.0.0     8
1      1      0      0      0      135.20.0.0    16
1      1      0      0      0      135.20.40.0   24
2      0      0      0      1      163.0.0.0     8
3      1      0      0      1      198.0.0.0     8
4      1      0      0      1      121.0.0.0     8
5      0      0      0      1      138.0.0.0     8
...
```

Col1 is a unique flow identifier (among all the flows of a same aggregation level). Col2 through Col5 takes the value 1 or 0 depending if a significant varation has been detected or

not in the packet time series, byte time series, SYN packet time series and flow time series, respectively. Col6 has the destination IP address of flow aggregate, and Col7 has the IP mask used to aggregate flows. In the EUQoS environment Col4 and Col5 do not exist.

At this point of the execution of the BASH file the potential anomalies have been detected. The following phases consist in the classification and identification of the anomalies. A set of functions is responsible for this task, and includes:

- *2Dplot_SourceDest*()
- *2Dplot_SourcePortDest*()
- *2Dplot_SourcePortDestPort*()
- *2Dplot_SourcePortPort*()
- *classificationModule*()

Input	Description
<anomalous IP file>	File with information about all the anomaly-candidate flows.
<slot>	Number of time slot at which the flow signaled as anomaly-candidate was detected.
<granularity us> <window us>	Definition of time window to use in microseconds. Same value as above.

Table A.5: Input parameters of family functions *2Dplot_()*.

The family of functions *2Dplot_*() as the name suggests, is responsible for plotting the graphs (by using gnuplot) that will be used in the visual anomaly signature and to create four files that will be used by the classification function. All these functions have the same input, which is described in Table A.5.

2Dplot_SourceDest()

This function relates the source IP addresses that are sending packets to the anomaly-candidate destination IP addresses. The output of this function has the following format:

```
Col1   Col2   Col3                    Col4   Col5
1      6      65.54.184.250       1      140.93.4.157
2      6      74.64.14.143        2      140.93.4.245
3      6      217.72.199.232             3      140.93.4.138
4      6      207.46.27.75        4      140.93.4.81
...
```

Col1 and Col4 have a unique identifier for the source and destination IP addresses, respectively. Col2 have information about the protocol type (in this case 6-TCP). In Col3 and Col5 are the source and destination IP addresses.

***2Dplot_SourcePortDest*()**

This function relates the source IP addresses/ports that are sending packets to the anomaly-candidate destination IP addresses. The output of this function has the following format:

```
Col1   Col2   Col3   Col4               Col5   Col6
1      6      80     65.54.184.250      1      140.93.4.157
2      6      443    74.64.14.143       2      140.93.4.245
3      6      3139   217.72.199.232     3      140.93.4.138
4      6      1863   207.46.27.75       4      140.93.4.81
3      6      443    217.72.199.232     3      140.93.4.138
...
```

This output file is similar to the one above, except for Col3 that is the port number that the source used to send packets to the destination IP address. All other Col numbers were shifted of one rank to the right.

***2Dplot_SourcePortDestPort*()**

This function relates the source IP addresses/ports that are used to send packets to the anomaly-candidate destination IP addresses/ports. The output of this function has the following format:

```
Col1   Col2   Col3   Col4               Col5   Col6   Col7
1      6      80     65.54.184.250      1      3400   140.93.4.157
2      6      443    74.64.14.143       2      16859  140.93.4.245
3      6      3139   217.72.199.232     3      3338   140.93.4.138
4      6      1863   207.46.27.75       4      3126   140.93.4.81
3      6      443    217.72.199.232     3      3339   140.93.4.138
...
```

As before, this output file only differs from the anterior in Col6 that now indicates the destination port number used to receive packets from the source.

***2Dplot_SourcePortPort*()**

This function relates the source IP addresses/ports that are used to send packets to an anomaly-candidate destination port. The output of this function has the following format:

```
Col1   Col2   Col3   Col4                Col5   Col6
1      6      80     65.54.184.250       1      3400
2      6      443    74.64.14.143        2      16859
3      6      3139   217.72.199.232      3      3338
4      6      1863   207.46.27.75        4      3126
3      6      443    217.72.199.232      3      3339
...
```

This output file has one column less than the previous one. The information gathered by this function is useful to distinguish the different types of Network Scan.

From the execution of these four functions a set of plots, identical to the one of Figure A.1, is created. Particularly, Figure A.1 represents the visual signature for a Network Scan of type *n* IP sources – 1 IP destination.

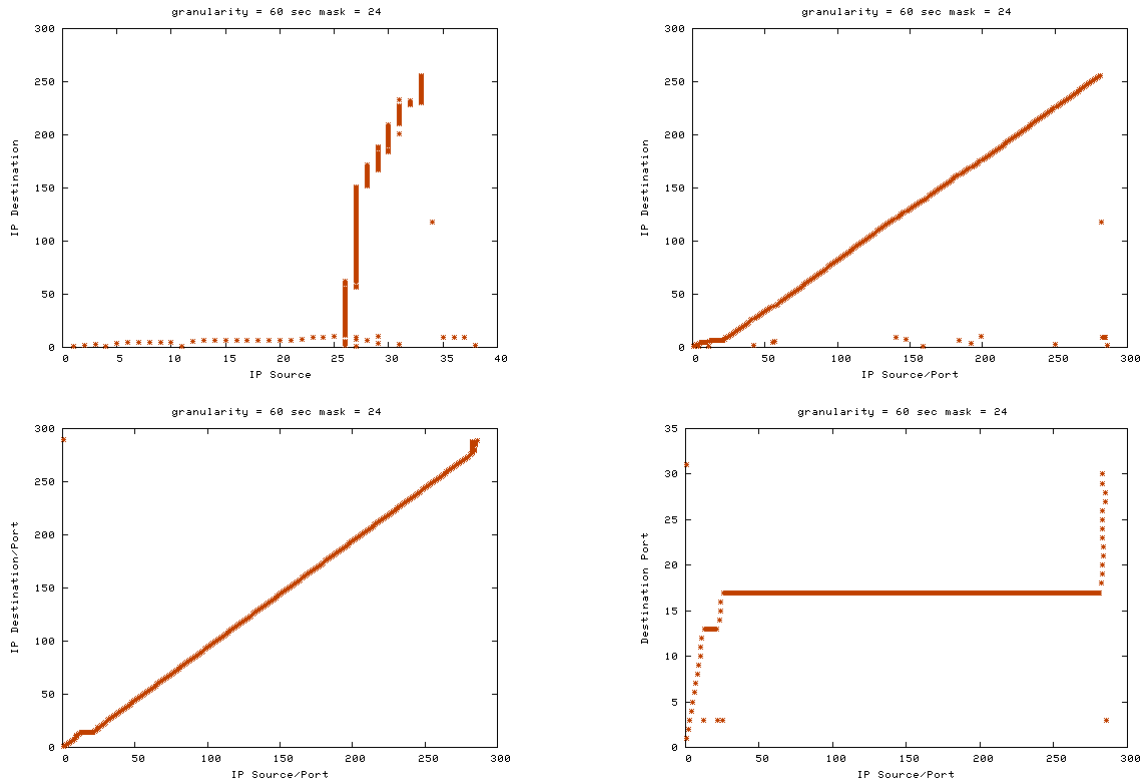

Figure A.1: Example of a visual signature of a Network Scan of type *n* IP sources – 1 IP destination.

classificationModule()

This function returns the type of anomaly associated to a flow or aggregate of flows, and all the IP features information associated (IP addresses and ports involved).

To accomplish such task, the function uses the files created by the *2Dplot_*() family. Particularly, from the identification information in each file, it constructs a matrix matrix[source_id, destination_id], in which matrix[s,d] = 1 if there is a flow from source *s* to destination *d*, and 0 otherwise.

With the matrix information, the knowledge of which data time series were affected by the anomaly-candidate, the level of aggregations at which the anomaly-candidate was detected, and the time-scales involved, it is possible to this function to classify the anomaly-candidate (or to reject it) , and to identify the flows involved. The input of this function is presented in Table A.6.

Input	Description
<anomalous IP file>	File with information about all the anomaly-candidate flows.
<slot>	Number of time slot at which the flow signaled as anomaly-candidate was detected.
<level>	Level of aggregation of flows to consider.

Table A.6: Input parameters for function *classificationModule().*

The output is a file named **Report**_*slot*.**txt** which informs about all the anomalies detected in the time slot <slot>. The structure of this file is depicted below. For each anomaly its type is identified, as well the destination and source IP addresses involved in the anomaly.

```
Type: DDOS – TCP Flood
Destination: 63.247.70.253
Source(s):
       140.93.4.251
-------------------------
Type: Flash Crowd
Destination: 224.2.127.254
Source(s):
       205.189.33.242
       130.92.2.10
       205.155.71.103
       131.111.168.120
       131.111.168.117
       205.155.71.101
       194.160.23.22
       136.244.96.102
       129.116.74.139
       136.142.65.154
```

Figure A.2: Example of content of file Report.txt.

A.2. Utilization

As stated above NADA is a tool that can be executed off-line or on-line. The execution off-line only requires the execution of a BASH file, while the execution online requires a tier-application that is responsible for simultaneously capturing traffic and analyzing it.

A.2.1. Offline Execution

The offline execution is assured by a bash file that calls orderly the functions presented in the previous section. An initial configuration of parameters is required before the execution of the script.

NADA_OSCAR.bash and NADA_EUQoS.bash are two scripts used to launch NADA process in the OSCAR and EUQoS environments, respectively. The input parameters required for both scripts are depicted in Table A.7.

Parameter	Description
<PATH_TRACE>	Complete path directory to the OSCARFIX or ERF traffic trace.
<PATH_DST>	Complete path directory for destination files.
<ID>	Short identification for trace. Could be the day of capture in format YYYYMMDD.
<TRACE>	Name of trace. Unless special cases it is always *flosws* in OSCAR environment. It can take any value in EUQoS environment.
<SERIES>	Common name to attribute to data time series.
<TIME_SLOT>	Common name to attribute to file with anomalous time slots.
<WINDOW_US> <WINDOW_SEC> <GRANULARITY_US> <GRANULARITY_SEC>	Definition of time window to use in microseconds and seconds, respectively. The word window is used in OSCAR environment and granularity in EUQoS environment.
<SCOPE_FILTER>	Value of K. Usually K>0 and K<=3.
<LEVEL>	Maximum level of aggregation of flows to consider. Defines a superior bound.
<PORT> <SERVER>	Port and server are configured with the port number and server name to where the Report file will be send. Their configuration is optional.

Table A.7: Initialization parameters to run the scripts *NADA_OSCAR.bash* and *NADA_EUQoS.bash*.

It is recommended to edit the scripts in a LINUX editor. An example of configuration for NADA_OSCAR.bash script follows:

```
#!/bin/bash
export PATH_TRACE=/home/test/trace/
export PATH_DST=/home/test/oscar
export ID=20060315
export TRACE=flows
export SERIES=time_series
export TIME_SLOT=time_slots
export WINDOW_US=60000000
export WINDOW_SEC=60
export SCOPE_FILTER=1.5
export PORT=4000
export SERVER=server.laas.fr
```

Finally, to launch NADA the final user just has to certify that he has the appropriate permissions to run scripts and type the command below in a LINUX shell. Because this is the offline mode, i.e., NADA will run over a previously captured traffic trace, the argument 0 must be typed when running the script.

```
./NADA_OSCAR.bash 0
./NADA_EUQoS.bash 0
```

All the intermediate and final files and directories will be created and removed as necessary.

A.2.2. Online Execution

Online execution is assured by an application, written in C language, and named *nada_oscar*() or *nada_euqos*(), depending of the type of environment being used.

Both applications simultaneously capture a traffic trace in the according data format, and detect, classify and identify the existing anomalous flows in traffic trace, by calling one of the bash scripts presented above. For each anomaly detected a file of type **Report**_*slot*.**txt**, as well the four plots that constitute the visual anomaly signature.

Since trace analysis is performed by the same functions of offline execution, before using the applications, the user must previously configure the bash file according to the rules in Section 3.1.1.

Table A.8 presents the input arguments for *nada_oscar*(), and Table A.9 the input arguments for *nada_euqos*().

Input Argument	Description
<window size>	Time window in seconds. This defines the minimum time slot size.
<port number>	Port number of server to where files with anomalies information should be sent.
<server name>	Name of server to where files with anomalies information should be sent.
<LAN prefix>	LAN prefix used during the conversion process of ERF format to OSCARFIX format.
<size LAN prefix>	Prefix used to aggregate traffic packets in flows. Used during the conversion process of ERF format to OSCARFIX format.
<capture duration>	Time of capture of data packets in seconds. If the user intends to capture traffic indefinitely the value 0 must be inserted.

Table A.8: Input arguments of application *nada_oscar*().

Input Argument	Description
<window size>	Time window in seconds. This defines the minimum time slot size.
<port number>	Port number of server to where files with anomalies information should be sent.
<server name>	Name of server to where files with anomalies information should be sent.
<capture duration>	Time of capture of data packets in seconds. If the user intends to capture traffic indefinitely the value 0 must be inserted.

Table A.9: Input arguments of application *nada_euqos*().

Finally, to launch NADA online, the user just has to type one of the command lines showed (the first one for OSCAR environment, the second one for EUQoS environment).

```
./nada_oscar 60 4000 server.laas.fr 140.0.0.0 24 0 #The tool will run indefinitely
./nada_euqos 30 4000 server.laas.fr 3000 #The tool will run 3000 seconds
```

All the intermediate and final files and directories will be created and removed as necessary.

Thesis: Contributions on Detection and Classification of Internet Traffic Anomalies

Abstract

The aim of this thesis is to develop a tool able of detecting, classifying and identifying traffic anomalies. Such occurrences are disturbing since they have potential to deviate network operations from their normal behaviour. Network Anomaly Detection Algorithm – NADA – is the approach developed.

The use of NADA and its accuracy are guaranteed by considering three axis of action: multi-criteria, multi-time and multi aggregation level. Together they allow the detection of traffic anomalies in traffic traces, as well as their classification through the definition of traffic profiles, particularly, anomaly traffic profiles. The latter ones are the basis of the definition of anomaly signatures databases. Hence, anomaly detection and classification form a couple that can be applied at several areas, ranging from network security to traffic engineering or overlay networks, to name a few.