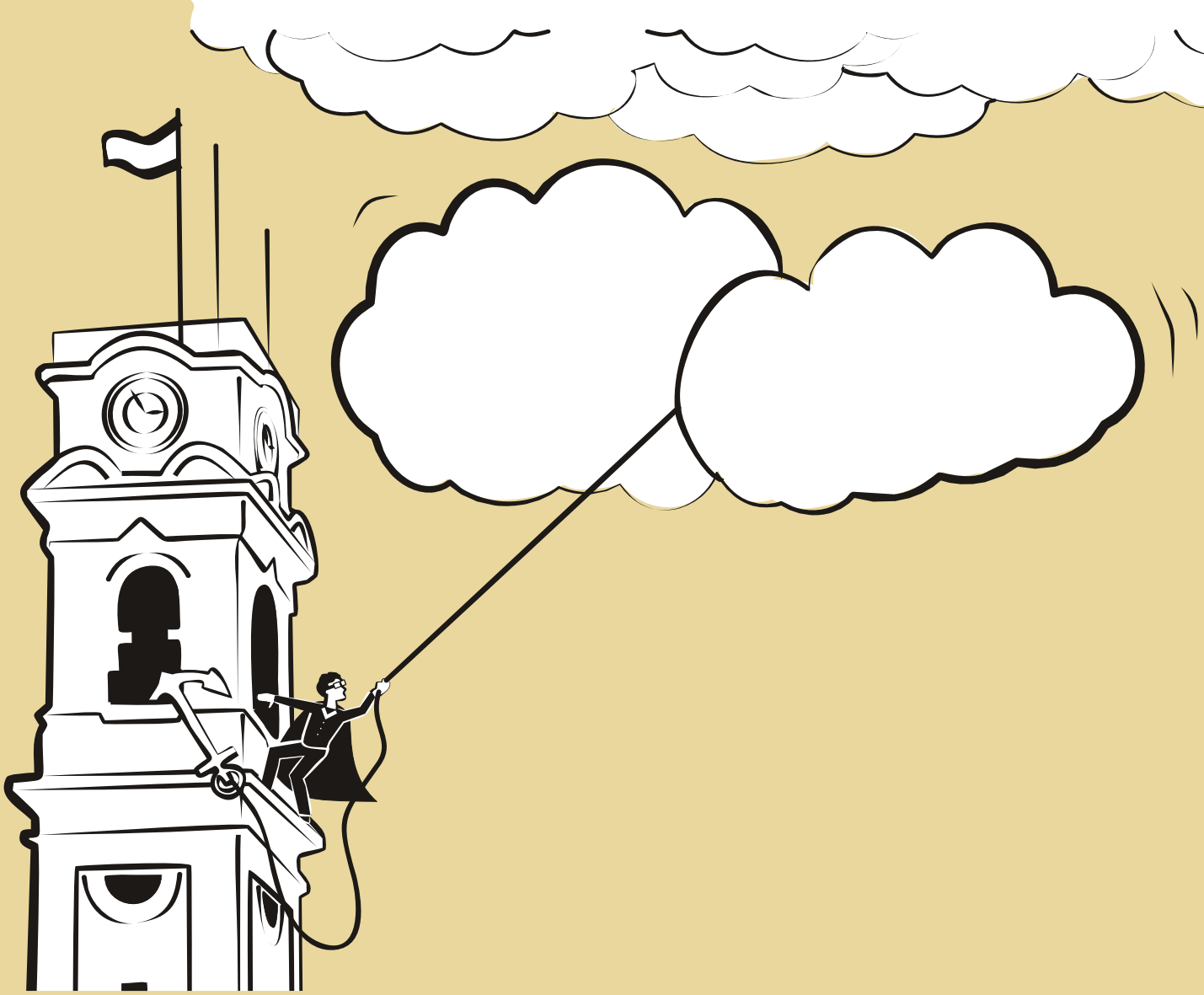


Bruno Lopes Dalmazo

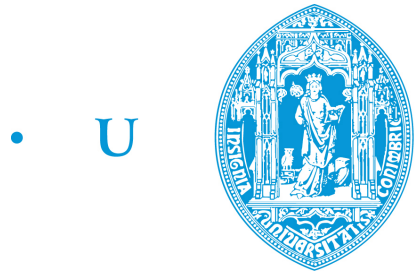
A Prediction-based Approach for Anomaly Detection in the Cloud

Tese de Doutoramento do Programa de Doutoramento em Ciências e Tecnologias da Informação, orientada pelo Professor Doutor João Paulo Vilela e pela Professora Doutora Marilia Pascoal Curado e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Outubro de 2018

· U C ·

Universidade de Coimbra



Department of Informatics Engineering
Faculty of Sciences and Technology
University of Coimbra

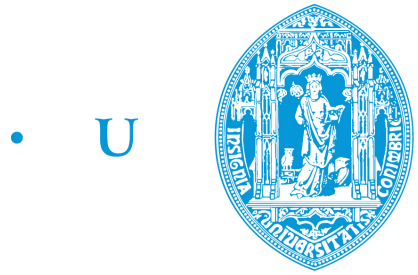
A Prediction-based Approach for Anomaly Detection in the Cloud

Bruno Lopes Dalmazo

Doctoral Program in Information Science and Technology
PhD Thesis submitted to the University of Coimbra

Advised by Prof. Dr. João P. Vilela
and by Prof. Dra. Marilia Pascoal Curado

October, 2018



Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Uma abordagem baseada em predição para Detecção de Anomalias na Nuvem

Bruno Lopes Dalmazo

Programa de Doutoramento em Ciências e Tecnologias da Informação
Tese de Doutoramento apresentada à Universidade de Coimbra

Orientado pelo Prof. Dr. João P. Vilela
e pela Prof. Dra. Marilia Pascoal Curado

October, 2018

This work was partially supported by the project CMU-PT/RNQ/0015/2009, TRONE - Trustworthy and Resilient Operations in a Network Environment; and the Brazilian National Council of Technological and Scientific Development (CNPq) under the grant 246645/2012-1.

Hakuna matata
— Jambo Bwana

Acknowledgements

Saying thank you, sometimes, is not enough to thank the kind and gentle people that in the moments of our lives, those more difficult, extend us the hand and offer us shelter.

I thank my mother, Fátima, for the support I always have received in all the moments of my life. Ricardo, my father, I would like to tell you that all of my achievements are to honour your memory. My thanks also to my sister, Aline, the rest of my family and friends that believed in me even when I did not. Of course, I would like to thank Filipa, for the patience and companionship that were essential to finalize this thesis, and also for doing everything you could to support me.

My gratitude to all friends that I have made in this journey: the laboratory guys, the Cambridge family, and my housemates. Especially, thank Roger and André for the support, insights, travels, parties, friendship and everything else. You are the best!

Finally, I would like to express my deepest gratitude to my advisors, Dr. João Vilela and Dra. Marilia Curado, for pushing me to high limits, to overcome my limitations with success, your remarkable guidance led me to arrive here. In addition, I would like to thank the Department of Informatics Engineering (DEI), the Centre for Informatics and Systems of the University of Coimbra (CISUC) and the University of Coimbra for the physical, administrative and human support that allowed me to conduct my research. This research could not have been possible without the Brazilian National Council of Technological and Scientific Development (CNPq) through its financial support.

Abstract

Computer networks are present everywhere, making them a key aspect to the proper functioning of products and services that are often served exclusively through the Internet. The pervasive nature of computer networks makes them particularly suitable to attacks. Therefore, more than just functional systems, we are also looking for systems that are reliable, available, scalable and secure. A solution to meet the growing demands of industries and customers alike is cloud computing. Among several other advantages of this paradigm, the possibility of increased profits by reducing costs with infrastructure and software licenses, while allowing for virtually unlimited growth is particularly relevant. However, these advantages are many times shadowed by the increased security risks that steam from having different entities involved, with relationships and responsibilities not properly identified. This may lead to misuse or malicious attacks against cloud computing, which may compromise sensitive information that is stored in shared third party facilities, and many open issues still prevail. Due to these and other issues, it is extremely important to devise new solutions that increase the trustworthiness of cloud computing environments and help to keep the continued growth in demand for virtualized resources. Facing this challenge, this work aims to study, analyze, propose, develop and evaluate several models and mechanisms to fill these gaps. Firstly, a systematic approach for selecting a group of candidate predictors that is suitable for cloud network traffic prediction is proposed. On the basis of this scenario, a predictor model for cloud network traffic that involves a tradeoff between prediction error, historical data dependence, computational costs, and timely response is proposed. Next, an Anomaly Detection System to support decision-making and counter attack malicious actions against cloud computing systems is presented. This contribution relies on network traffic prediction to obtain features that represent the expected appropriate behaviour of the cloud network traffic used jointly with a Support Vector Machine model for detecting anomalous events in the cloud environment. Finally, a mechanism for determining the similarity level between features of the alarms is proposed. This mechanism aims to optimize the efficiency for generating alarms, decreasing the network data traffic to manage the IDS and its associated transfer costs. The benefits and drawbacks of the contributions were demonstrated in realistic simulations using data from real network traces. Furthermore, the evaluations were conducted with well-known metrics and the results show that all the proposed mechanisms were able to outperform similar proposals in literature.

Keywords: Security; Cloud Computing; Network Traffic Prediction; Intrusion Detection System; Alarm Management.

Resumo

Redes de computadores estão presentes por todos os lados, se tornando um ponto chave para o funcionamento adequado de produtos e serviços que são oferecidos exclusivamente através da Internet. A natureza pervasiva das redes de computadores as tornam sujeitas a ataques. Desse modo, mais que apenas sistemas funcionais, também estamos a procura de sistemas que são confiáveis, disponíveis, escaláveis e seguros. Uma solução que vai de encontro com a crescente demanda da indústria e clientes é a computação em nuvem. Entre muitas outras vantagens desse paradigma, a possibilidade de aumentar lucros através da redução de custos com infraestrutura e licenças de software, ao mesmo tempo que permite um crescimento praticamente ilimitado é relevante. No entanto, essas vantagens são muitas vezes obstruídas pelo aumento dos riscos de segurança que cobrem as entidades envolvidas, com relacionamentos e responsabilidades não propriamente estabelecidos. Isso pode levar a abusos ou ataques maliciosos contra a computação em nuvem, o qual pode comprometer informação sensível que é armazenada e em instalações de terceiros compartilhada. Devido a esses e outros problemas, é de extrema importância conceber novas soluções que aumentem a confiança do ambiente de computação em nuvem e ajude a manter um crescimento contínuo na demanda por esses recursos. Diante deste desafio, esta tese tem como objetivo estudar, analisar, propor, desenvolver e avaliar vários modelos e mecanismos para preencher essas lacunas. Em primeiro lugar, uma abordagem sistemática para a seleção de um grupo de preditores candidatos adequados para a previsão do tráfego da rede em nuvem é proposta. Com base nesse cenário, um modelo de predição para o tráfego de rede em nuvem que envolve uma relação entre erro de predição, dependência histórica de dados, custos computacionais e tempo de resposta é proposto. Em seguida, um Sistema de Detecção de Anomalias para apoiar a tomada de decisões e combater ações mal intencionadas contra sistemas na nuvem é apresentado. Esta contribuição baseia-se na previsão de tráfego de rede para obter variáveis que representam o comportamento adequado esperado do tráfego de rede usado em conjunto com um modelo de Máquina de Vetores de Suporte para a detecção de eventos anômalos no ambiente da nuvem. Finalmente, um mecanismo para determinar o nível de similaridade entre as variáveis que descrevem um alarme é proposto. Este mecanismo visa otimizar a eficiência na geração de alarmes, diminuindo o tráfego de dados para gerenciar um IDS e seus custos de transferência associados. Os benefícios e desvantagens das contribuições foram demonstrados em simulações realistas usando dados de rede reais. Além disso, as avaliações foram realizadas com métricas bem conhecidas e os resultados mostram que os mecanismos propostos foram capazes de superar propostas similares na literatura.

Palavras-chave: Segurança; Computação em Nuvem; Predição de Tráfego de Rede; Sistemas de Detecção de Intrusão; Gerenciamento de Alarmes.

Foreword

The work detailed in this thesis was accomplished at the Laboratory of Communication and Telematics (LCT) of the Centre for Informatics and Systems of the University of Coimbra (CISUC), within the context of the following projects:

Project TRONE - Trustworthy and Resilient Operations in a Network Environment (CMU-PT/RNQ/0015/2009). The aim of this project was to enhance network quality of service (QoS) and quality of protection (QoP), operational efficiency and agility, in the context of accidental and malicious operational faults of expected increasing severity. The work performed in the project aimed to discuss the new challenges in network traffic predictors in the cloud and propose an analysis mechanism, focused on providing a standardized approach for evaluating the best candidate predictor models for these environments, which resulted in several publications.

Project iCIS - Intelligent Computing in the Internet of Services (iCIS) project (CENTRO-07-ST24-FEDER-002003), co-financed by QREN, in the scope of the "Mais Centro" Program. The goal of this project was focused in two complementary, yet interrelated, directions: advanced algorithms and infrastructures for information capture and management in the Internet, and intelligent algorithms for data analysis for smart Internet services. The activities include network traffic management and knowledge extraction. The achieved results were published in several conference papers and journals.

This work was funded by the following grants:

Student travel grant ACM Special Interest Group on Data Communication (SIGCOMM), held on UCLA campus in Los Angeles, USA, 2017.

Student travel grant ACM Special Interest Group on Data Communication (SIGCOMM), held in Florianopolis, Brazil, 2016.

Doctoral grant Brazilian National Council of Technological and Scientific Development (CNPq) (246645/2012-1).

Project grant Trustworthy and Resilient Operations in a Network Environment (CMU-PT/RNQ/0015/2009).

The outcome of the design, experiments, and assessments of several mechanisms on the course of this work resulted in the following publications:

Journal papers:

- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, **"Triple-Similarity Mechanism for Alarm Management in the Cloud"**, Computers & Security, vol. 78, pp. 33-42, Elsevier, 2018. Impact factor: 2.65
- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, **"Performance Analysis of Network Traffic Predictors in the Cloud"**, Journal of Network and Systems Management, vol. 25, 2, pp. 290-320, Springer, 2017. Impact factor: 1.75
- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, **"Online traffic prediction in the cloud"**, International Journal of Network Management, vol. 26, 4, pp. 269-285, John Wiley & Sons, 2016. Impact factor: 1.34

Conference papers:

- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, "**Security and Trustworthiness in Cloud Computing**", in Meeting with Science and Technology in Portugal, 2017
- Dalmazo, Bruno L. and João P. Vilela and Simões, P. and Marilia Curado, "**Expedite feature extraction for enhanced cloud anomaly detection**", in NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016
- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, "**A SVM Model based on Network Traffic Prediction for Detecting Anomalies**", in 21th edition of the Portuguese Conference on Pattern Recognition, 2015
- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, "**Online Traffic Prediction in the Cloud: A Dynamic Window Approach**", in The 2nd International Conference on Future Internet of Things and Cloud, 2014
- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, "**Predicting Traffic in the Cloud: A Statistical Approach**", in IEEE International Conference on Cloud and Green Computing, 2013

Cooperation papers:

- Nefoussi, Jonathan J. and **Dalmazo, Bruno L.** and Lunardi, Roben, "**PySoneta Um sistema baseado em Python para visualização de dados do tráfego de rede**", in 12th edition Escola Regional de Redes de Computadores, 2014

- R. L. dos Santos and J. A. Wickboldt and **Dalmazo, Bruno L.** and L. Z. Granville and L. P. Gaspary and R. C. Lunardi, "**Identifying the root cause of failures in IT changes: Novel strategies and trade-offs**", in 2013 IFIP/IEEE International Symposium on Integrated Network Management, 2013

Contents

List of Figures

List of Algorithms

List of Tables

Abbreviations and Acronyms

ACK	Acknowledgement
ACR	Absolute Category Rating
ADSL	Asymmetric Digital Subscriber Line
ANN	Artificial Neural Network
AP	Access Point
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
CAIDA	Center for Applied Internet Data Analysis
CIP	Cloud Infrastructure Provider
CNA	Computer Network Attack
CRM	Customer Relationship Management
CSC	Cloud Service Client
CSP	Cloud Service Provider
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DoS	Denial of Service
DR	Detection Rate
EMA	Exponential Moving Average
FARIMA	Fractional Autoregressive Integrated Moving Average
FNR	False Negative Rate
FPR	False Positive Rate
HIDS	Host-based Intrusion Detection System
IaaS	Infrastructure as a Service
IDS	Intrusion Detection System
IETF	The Internet Engineering Task Force
IP	Internet Protocol
IPS	Intrusion Prevention System

IT	Information Technology
LRD	Long-Range Dependency
MAPE	Mean Absolute Percent Error
MMSE	Minimum Mean Squared Error
MSE	Mean Squared Error
NACK	Negative Acknowledgement
NIDS	Network-based Intrusion Detection System
NIST	The National Institute of Standards and Technology
NMSE	Normalized Mean Square Error
OS	Operating System
PaaS	Platform as a Service
PMA	Poisson Moving Average
QoS	Quality of Service
R2L	Remote to Local
RTP	Real-time Transport Protocol
RTT	Round-Trip Time
SaaS	Software as a Service
SDN	Software-Defined Networks
SLA	Service Level Agreement
SMA	Simple Moving Average
SNMP	Simple Network Management Protocol
SRD	Short-range Dependency
SSL	Secure Socket Layer
SSLA	Security Service Level Agreement
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TLS	Transport Layer Security
U2R	User to Root
UDP	User Datagram Protocol
VBD	Virtual Block Device
VM	Virtual Machine
WLAN	Wireless Local Area Network

WMA Weighted Moving Average

WSLA Web Service Level Agreement

WWW World Wide Web

Chapter 1

Introduction

> Logic will get you from A to B. Imagination will take you everywhere.
>
> *(Albert Einstein)*

Contents

This work aims at improving security and trustworthiness of cloud computing environments by developing a model for predicting cloud network traffic, an approach for detecting anomalies in cloud network traffic that relies on traffic prediction, as well as a mechanism for aggregating similar alarms from an IDS in the context of the cloud network traffic. The motivation and research scope of this thesis are presented and investigated followed by the discussion of the proposed objectives together with the respective contributions, as well as the thesis outline.

1.1 Motivation and Problem Statement

A study performed by the Carbon Disclosure Project attests that large IT companies could achieve expressive cost savings and carbon reductions by 2020 if they move their IT assets to the cloud [Baumast, 2013]. This study claims that these companies could benefit from billions in savings. Furthermore, another study published in *The New York Times* [Gilmer, 2011] went deeper: they estimate that companies in the U.S.A. using cloud computing can save $12.3 billion per year by 2020. In addition, these companies could make an annual reduction in carbon emissions equivalent to 200 million barrels of oil.

Cloud computing enables the adoption of a promising computing model that allows consuming a computer resource rather than having to build and hold their own computing infrastructure. However, a widespread adoption of this paradigm

has been hampered by the lack of security mechanisms. For instance, it is predicted that cybercrimes will cost the world $6 trillion annually by 2021 [Networking, CISCO Global Cloud Index, 2018]. Furthermore, 74% of Information Technology executives believe that security is the top factor that needs to be addressed to expand the use of cloud computing [Subashini and Kavitha, 2011].

In this context, the last several years have been the most remarkable period from a cloud security threat perspective. For illustrative purposes, in 2017 there were at least two high-profile incidents involving gaming platforms. Firstly, Blizzard Entertainment reported a flood of junk traffic that caused problems for players of Overwatch and World of Warcraft. Another situation involves the UK National Lottery that was seriously damaged for being unable to place their stakes online or via applications for hours. These two organizations were victims of a Distributed Denial of Service (DDoS) performed by botnets powered by cloud infrastructures [Khalimonenko et al., 2017].

The cloud network is prone to several kinds of threats. Although some of these threats are able to leave traces, the task of identifying them is not trivial. To give an example, there are attacks that cause an anomalous behaviour in the network traffic which allows the threat to be detected. However, at the same time, the elastic and scalable nature of cloud environments makes them prone to undergo sudden changes [Ballani et al., 2011, Vieira et al., 2010], which makes it even harder to detect which parts of the incoming traffic are caused by vandalism or are being used legitimately.

Moreover, the cloud provider also has to deal with a huge number of devices and virtual machines so that it can manage all the assets of its network infrastructure. It is increasingly being found that neglecting this area of management can cause irreparable economic damage to businesses and their customers [Owezarski et al., 2013].

It is therefore imperative to monitor and analyze these networks. In doing so, relevant information about network traffic can be collected and used to support decision-making processes. After all this information has been gathered, it is possible, for example, to identify and analyse suspicious network traffic patterns to prevent similar situations from occurring in the future [Dainotti et al., 2012]. This calls for network traffic predictions mechanisms that favour timely detection of issues in the network, which is particularly challenging in cloud environments due to the volume and volatility of traffic and resources available, as addressed in chapter 3.

In addition to the network traffic prediction, several techniques have already been proposed to perform anomaly detection in the cloud environment, such as fuzzy logic [Patel et al., 2013], entropy-based [Wang et al., 2010], artificial neural networks [Vieira et al., 2010] and decision tree classifier [Fu, 2011]. Incidentally, different types of network traffic information are used to detect anomalies, such as the behaviour of protocols, CPU utilization and user logs. However, there is an apparent deficiency in their ability to detect anomalies when analysing a large amount of data. In particular, these techniques require extensive tuning

to improve their sensitivity to achieve satisfactory results. There is also no consensus about the best way to represent the huge volume of data generated by the cloud infrastructure. As a result, the literature lacks mechanisms that can improve the accuracy of anomaly detection for cloud environments while reducing false-positive detection rates, as addressed in chapter 4".

A further aggravating factor is regarding to the number of alarms generated. Alarm management approaches have been proposed in the literature such as alarm correlation [Benferhat et al., 2013], regular expression matching [Li et al., 2010] and clustering alarms [Lo et al., 2010]. However, these works are more concerned with increasing the number of true alarms. As a result, they fail to meet a low number of false alarms as well as decreasing the number of control messages in general. To make matters worse, it is known that around 99% of the alarms are false both in cloud computing [Patel et al., 2013] and in traditional environments [Elshoush and Osman, 2011, Hubballi and Suryanarayanan, 2014, Di Pietro and Mancini, 2008]. The wide disparity between the true and false alarms generated has certainly compromised the performance of IDS. From this, two significant problems arise, as addressed in chapter 5: the huge volume of control messages between the virtual machines and the servers and the associated transfer costs.

A review of diverse solutions that have been proposed to address the aforementioned challenges are explored and summarized in the following chapters. However, as far as the literature goes, there are still gaps to be filled. A set of diverse solutions that have been proposed to address the aforementioned challenges is explored and summarized in the following chapters. Their main shortcomings are identified and mechanisms designed to address those gaps are proposed and evaluated in this thesis.

1.2 Objectives and Contributions

The goal of this thesis is to enhance security aspects of the cloud computing environment. This is done by addressing security aspects including areas such as network traffic prediction models, intrusion detection systems for the cloud and methodologies for aggregating alarms from the IDS. This task brings attention for more specific activities, for instance, characterization of the needs, design new solutions, implement and evaluate such proposals.

More specifically, this thesis contains a comprehensive analysis of the state of the art in these fields in order to identify and mitigate numerous open issues. The proposed solutions were evaluated by using traces from real network operations. In addition, the entire methodology presented in this thesis allows a systematic comparison of results from a general perspective among several research efforts found in the literature.

The specific goals of this thesis are as follows:

Goal 1 – Proposing a new network traffic prediction model and a methodology

for selecting and evaluating a set of prediction models that is suitable for the highly dynamic cloud computing environment;

Goal 2 – Extracting features that represent the expected appropriate behaviour of the cloud network traffic, then used jointly with a Support Vector Machine (SVM) for detecting anomalous events in the cloud environment;

Goal 3 – Grouping similar alarms that may correspond to the same attack (or attack attempt) in order to optimize the efficiency for generating alarms, decreasing the network data traffic to manage an IDS and its associated transfer costs.

Taking into consideration the specific goals, this thesis has succeeded in producing the follow main contributions:

Contribution 1, The Poisson Moving Average Model
A new Moving Average approach based on the Poisson distribution is proposed. A Poisson process is used to determine the probable minimum and maximum number of transactions that can occur within a given time period, from a series of discrete values. The models and mechanisms proposed in chapters 3 and 4 of this thesis benefit from the findings generated in this contribution.

Contribution 2, The Dynamic Window Size Algorithm
To reduce the complexity of predicting network traffic, time-bounded past information is considered by means of a sliding window with size defined by the Dynamic Window Size Algorithm, which makes it suitable for online prediction in a cloud computing context. This contribution plays an important role as basis of the dynamic solution for predicting network traffic in the Section 3.3.3.

Contribution 3, The Feature Extraction Approach
Feature extraction involves reducing the amount of information required to describe a large set of data, therefore enabling its processing by the mechanism for detection anomalous events in the cloud environment. This contribution is presented in Section 4.3.2.

Contribution 4, The Anomaly Detection Mechanism
The proposal, design, and evaluation of an Anomaly Detection Mechanism is one of the contributions of this thesis. The purpose of this mechanism is to provide an efficient method to detect anomalies in cloud-based network traffic. Section 4.3 presents this contribution.

Contribution 5, The Triple-Similarity Mechanism
Another contribution of this thesis is grouping similar alarms that may correspond to the same attack in order to reduce the number of messages sent from the virtual machines to the servers. This contribution is detailed in Chapter 5.

Contribution 6, The Severity Adjustment of Alarms Algorithm
This contribution allows to analyse the output of the Triple-Similarity Mechanism ir order to seeking for alarms and classifies them according to a

database. Then, the algorithm assigns a level of severity. This mechanism is described in Section 5.3.4.

1.3 Outline of the Thesis

The remainder of this thesis is organized in six chapters, as described below.

Chapter 2 – General Background
Introduces a general background and main concepts about the environment around cloud computing and the solutions proposed. In addition, these definitions are used for designing the prediction model, the anomaly detection mechanism and the similarity approach in the rest of this thesis.

Chapter 3 – Network Traffic Prediction in the Cloud
Discusses the state of the art and presents a taxonomy for network traffic prediction models, as well as an analysis mechanism that provides a standardized approach for evaluating network traffic predictors based on global and local data analysis. Moreover, a new traffic prediction approach based on a statistical model where observations are weighted with a Poisson distribution is proposed. The outcomes of this mechanism enable the performance comparison of several predictors in the cloud, particularly in terms of accuracy, historical dependency, time and computational overhead.

Chapter 4 – Intrusion Detection for Cloud Network Traffic
Proposes an approach to detect anomalies in the cloud scenario. This work differs from previous anomaly detection techniques since it relies on a distributed and collaborative mechanism that combines a Support Vector Machine model with features extracted from a Poisson Moving Average predictor. Moreover, a sensitivity analysis showing the trade-off between the time granularity and the accuracy of the model is presented.

Chapter 5 – Triple-Similarity Mechanism for Alarm Management
Looks into the main issue generated by Intrusion Detection Systems for cloud computing, namely, the huge number of alarms generated over time. To address this problem, it was proposed the Triple-Similarity Mechanism, a methodology for aggregating similar alarms in the context of the cloud network traffic and an algorithm to assign severity level to alarms.

Chapter 6 – Conclusions and Future Work
Presents the conclusions that emerged from the research described in this thesis, as well as the outline of future research to further advance this work.

Chapter 2

General Background

All truths are easy to understand once they are discovered; the point is to discover them.

(Galileu Galilei)

Contents

This chapter introduces concepts and definitions required to provide theoretical basis for better characterization of the environment around cloud computing and the solutions for improving security aspects. An overview about cloud computing is presented in Section 2.1. After that, Section 2.2 discusses the characterization of network traffic in the cloud and the main threats that may jeopardize its operation. Section 2.3 provides a general description of Intrusion Detection Systems (IDSs) focused on virtualized environments. Finally, Section 2.4 depicts concepts about similarity in order to aggregate alarms and minimize the network traffic and its associated costs.

2.1 Cloud Computing

Cloud computing is a term commonly accepted to describe several different computing concepts that involve a large number of computers providing computer power over the Internet. This definition is also related to Network Provisioning in the sense of providing clients with software hosted in remote locations. In this case, services are assigned to the customer by the provider, and this kind of relationship is known as Customer Relationship Management (CRM). Usually, cloud computing is characterized as a CRM between one provider and many users (1:N) [Buttle, 2015].

Cloud computing enables the realization of a promising computing model that leverages on the need for more powerful and ubiquitously available resources. Despite of the several definitions of cloud computing mentioned in Section 2.1.1, some characteristics are widely agreed upon, such as the possibility to provide computing power dynamically on demand. In order to provide a better understanding about cloud computing challenges and identify important research directions in this topic, the key concepts such as the architectural principles, security responsibilities as well as research challenges will be addressed.

2.1.1 Definition

Among the several definitions of cloud computing, some of them deal with operational aspects and define cloud computing as a large scale distributed computing paradigm for storing, managing, processing and hosting dynamically scalable resources that are delivered on demand over the Internet [Zhang et al., 2010, Armbrust et al., 2010, Joshi et al., 2012]. However, other works prefer to highlight the economic aspects of cloud computing, for instance, defining cloud computing as a way to increase the capacity or add capabilities dynamically without investing in new infrastructure, training new personnel, or licensing new software [Subashini and Kavitha, 2011, Hwang et al., 2009, Jensen et al., 2009].

In this thesis, the view of the cloud computing paradigm is defined by The National Institute of Standards and Technology (NIST): *Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction* [Mell and Grance, 2011]. This definition covers all the essential characterizing features of cloud computing systems.

2.1.2 Architectural Principles

The NIST [Mell and Grance, 2011] classified cloud computing systems according to four deployment models: private cloud, community cloud, public cloud and

hybrid cloud. Furthermore, the cloud computing can also provide three service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Figure 2.1 depicts the relationship between the cloud deployment models and the delivery models in terms of the layers that compose the system and the corresponding management responsibilities.

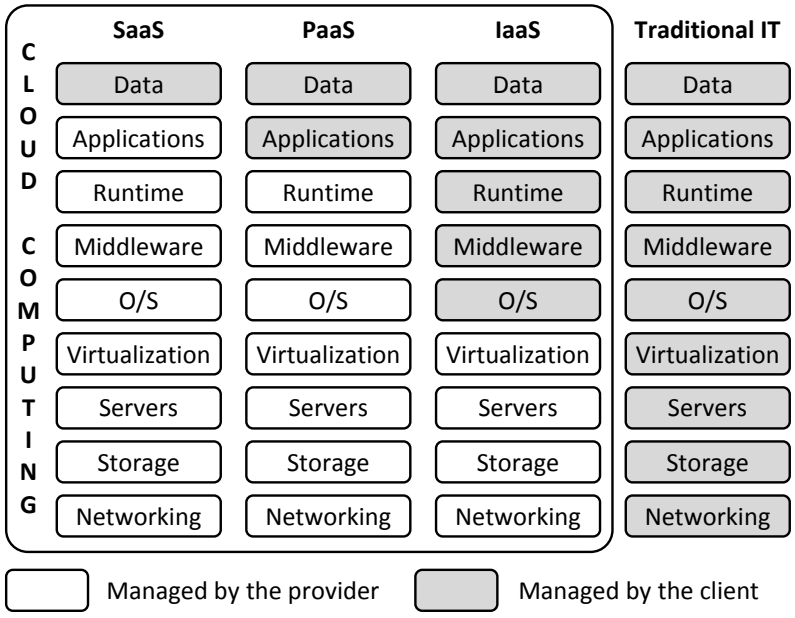


Figure 2.1: Cloud computing architecture and responsibilities

Deployment Models

There are essentially four cloud deployment models, each of which has specific trade-offs for organizations that are migrating services and operations to cloud-based environments. They are presented below:

- Private cloud – this cloud deployment model consists of a cloud infrastructure that is operated exclusively for one organization. However it can be managed by the organization itself or a third party;
- Public cloud – the cloud infrastructure is available to the general public or a large industry group and is owned by an organization selling cloud services;
- Community cloud – the cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns and goals (e.g., security requirements, business, e-commerce, and compliance considerations);
- Hybrid cloud – it is characterized by a cloud infrastructure that is composed of two or more clouds with different models (private, community or public).

These deployment models can be implemented following different types of delivery models, as described below.

Delivery Models

Delivery Models are service models by which different types of services are delivered to the end user [Kandukuri et al., 2009, Hwang et al., 2009]. Three common cloud delivery models have become widely established and formalized: Software as a Service, Platform as a Service and Infrastructure as a Service. They are described as follows:

- SaaS – consists of providing applications that are accessible to several client devices through a thin client interface such as a web browser (e.g., web email). One of the biggest benefits for these clients is the potential to reduce IT support costs by outsourcing hardware, software maintenance and support. In short, it is a software deployment model where applications are remotely hosted by the service provider and made available to customers on demand, usually over the Internet;

- PaaS – comprehends a platform layer consisting of operating systems and application frameworks that enable client to minimize the workload of deploying new applications directly on the virtual machines. Applications may be created using programming languages and tools supported by the provider. In this delivery model, the user does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and application hosting environment configurations;

- IaaS – comprises the hardware layer and the infrastructure layer, including provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. Usually, the client may be responsible for managing some of the physical resources (e.g., servers, routers, switches). The virtualization layer provisions instances of the physical infrastructure to different clients by means of virtualization techniques.

One of the main advantages of these service models is their automatic-scaling ability that allows them to provide the necessary means (not more, not less) to achieve a desired performance level with low operating costs. However, the fact that the data (in cloud computing) is usually placed far from the clients can raise some discomfort due to the lack of control and knowledge of how their data is stored and secured, specially for enterprises that still follow the traditional on-premise model, where the data continues to reside within the enterprise's boundaries, and subject to their own policies.

Different service models place distinct levels of security requirements in the cloud environment, and each service carries its own security issues [Kandukuri et al., 2009]. There is an inherent hierarchical structure among the three service models, with IaaS forming the base for all cloud services. Namely, PaaS builds upon IaaS to provide a platform that, in turn, is used as basis for SaaS. In this sense, while the capabilities are inherited along this hierarchy, the information security issues and the risks are as well.

There are many trade-offs to each model in terms of integrated features, such as management level vs. security. For example, if the cloud service provider is only responsible for implementing security mechanisms at the lower-layers of the cloud architecture, end users can unexpectedly become responsible for implementing and managing the security mechanisms at higher layers.

Clearly, cloud computing is a disruptive technology which changed the way traditional services are delivered and had an impact on the IT sector as a whole. More than ever, neglecting the management of these networks may cause irreparable economic harm to businesses and their customers. Network administrators of these cloud-supporting networks must then monitor and analyse those networks in order to collect relevant information about network traffic that may be used to support decision-making in case of failures.

Collecting statistics of the network is a useful management task that enables traffic patterns to be understood and strategies to be planned in order to prevent future problems. When these statistics accumulate over time, inferences can be made with respect to the future behaviour of the network traffic and when an abnormality occurs the administrator will have enough time to act before the problem worsens. In light of this, a general overview of this is subject of discussion in the next section.

2.2 Network Traffic Prediction

The uprise of next generation networking paradigms, such as the Internet of things and cloud computing, has entirely changed the way that networks are conceived and handled. By deploying a cloud computing model, organizations have many advantages such as on-demand computing services and reduced maintenance costs [Mell and Grance, 2011]. However, along with these benefits, cloud computing brought a multitude of challenges into the focus of worldwide research [Jennings and Stadler, 2015]. Many services and products rely on cloud-based systems and networks. Ignoring the management of these assets may cause irreversible economic damage to the cloud providers [Alshaer, 2015]. In this context, network managers need tools suited to deal with the high network traffic volume common in cloud environments.

2.2.1 Motivation

In order to collect relevant information about network traffic, the network administrator has to monitor and analyse the computer network. Monitoring a network allows the cloud provider to analyse many metrics with respect to the network traffic, such as throughput, response time, jitter, lost data, etc. Most monitoring tools available nowadays provide a graphical interface of network statistics, from which problems can be identified and remedied [Plonka and Barford, 2009].

Additionally, network administrators can use these statistics to perform routine tasks that may or may not show anomalies in the traffic. Eventually, these anomalies may be diagnosed as a problem, for example, locate a server down or a server that is getting an abnormal number of requests. This kind of problem was widely investigated by Ricardo dos Santos with Root Cause Analysis [dos Santos et al., 2013].

Monitoring the network traffic allows constantly keeping statistics of network connectivity and applications' availability, facilitating the detection of problems in hosts, networks, or servers. After capturing all this information, several patterns into the network traffic that characterize these anomalies can be identified. These patterns help us to plan strategies to prevent similar problems that may happen in the future [Dainotti et al., 2012]. When these statistics accumulate over time, it is possible to make inferences about the future behaviour of network traffic. Thus, when an abnormality occurs, the network administrator will have time to act before the problem worsens.

It is known that there are differences between the normal behaviour of network traffic and an anomaly. However, the transition between these two extremes is obscure, *i.e.*, we do not know at which point the network traffic ceases to represent legitimate use and should be considered an anomaly, as illustrated in Figure 2.2. Studies have shown that the task of identifying anomalous behaviour in network traffic is not a trivial matter [Dainotti et al., 2012, Yan-hui and Tao, hina]. This challenge is even greater in cloud computing due to the scalability and dynamics of this environment [Dhage and Meshram, 2012]. In particular, one must be careful at dealing with an abrupt request of high processing load, which can easily be confused with malicious use.

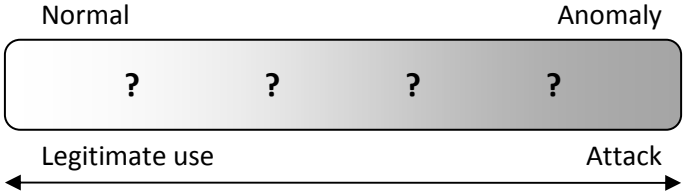


Figure 2.2: Identifying an attack by network traffic behaviour

As previously stated, the cloud network traffic is subject to a large number of

threats. It is worth noticing that the lack of security mechanisms is the top factor that prevents the wide adoption of cloud service models. Moreover, 74% of Information Technology executives believe that security is one of the main issues that needs to be addressed [Subashini and Kavitha, 2011]. Although misconfiguration or hardware failure can cause occasional anomalies in the network traffic, most of the time the root cause is an intentional attack. The next subsection will present the most common types of attacks that can be detected by using techniques of predicting the network traffic.

2.2.2 Computer Network Attack

Computer Network Attack (CNA) is composed by actions usually practiced by software designed with harmful intentions. The consequences are quite varied, some of them aim to infect or invade another's computer to then damage the hardware or software, by deleting files, changing the operation of the machine or even leaving the computer vulnerable to other types of attacks. There are also those aimed at sensitive user's data such as passwords and credit card numbers or just to capturing personal information.

Other definitions for CNA can be found in:

- The Committee on National Security Systems of United States of America defines an attack as: Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself [Kissel, 2011].
- The Internet Engineering Task Force (IETF) defines attack in RFC 2828 as: An assault on system security that derives from an intelligent threat, *i.e.*, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system [Shirey, 2000].

The cloud network is prone to several kinds of threats. Although some of these threats are able to leave traces, the task of identifying them is not trivial. For instance, there is a class of attacks that may cause anomalies in the behaviour of the network traffic, as seen in the next subsection.

2.2.2.1 Identifying an Anomaly

A network anomaly corresponds to a circumstance in which the network behaviour deviates from its normal operational pattern [Hajji, 2005]. The identification of an anomaly requires the network administrator to learn before-hand about the nature of normal traffic behaviour. A deep knowledge of its computer network is necessary to ensure an adequate degree of protection, and this is an important point to make the system more resilient and protect the network traffic against threats.

There is no simple and fast method from which one can affirm that the network presents a normal legitimate traffic. Traditionally, this requires monitoring the network for a long period to extract the characteristics of normal behaviour of the network. This type of methodology is known in literature as building a traffic baseline [Palmieri et al., 2013].

In general, there is no consensus about a rule or criterion regarding how much the normal traffic can be divergent from the abnormal. The judgment of this deviation is based on the context of past experience materialized through the observation of the network baseline. The key point for success in this task is defining which are the tolerable limits. These tolerable limits are known as thresholds and they are estimated as the deviation from normal range under which the measured feature is assumed to operate under normal conditions. During the network monitoring, whenever some observed variable exceeds these thresholds an alarm is generated [Rittinghouse and Ransome, 2016].

It is important to emphasize that a network traffic anomaly does not always correspond to the occurrence of an attempt to attack [Cuppens and Miège, 2002]. For instance, an anomaly could be generated from a network failure event or temporary misconfiguration that results in a problem or outage. Usually, it occurs soon after installing some new equipment in the network infrastructure. Network traffic anomalies are also common whenever a new software release becomes available or there is an increased external interest in a web site due to some kind of national (international) advertising [Barford et al., 2002, Sperotto et al., 2010].

2.2.2.2 Main Types of Attacks that Cause Anomalies

Without appropriate security tools and effective controls plan, the network data might be subjected to an attack. Among the existing attacks, it is possible classify them in two groups: passive and active.

Passive attack means that the information is just visualized without change of the data. For instance, passive attacks include traffic analysis, monitoring of unprotected communications, decrypting weakly encrypted traffic, and capturing authentication information such as passwords.

An active attack, instead, alters information with the intention of destroying or damaging the data. For example, active attacks include attempts to cheat or break protection features, to introduce malicious code, and steal or modify sensitive information.

There are many works in the literature about the most common attacks in cloud computing against networks or isolated virtual machines [Subashini and Kavitha, 2011]. This subject is so vast that could be the subject of a dedicated section. It is possible to list numerous types of attacks, but here will be mentioned only attacks that may generate some kind of anomaly in network traffic [Modi et al.,

2013]. Briefly, the main ones are:

- **Denial of Service (DoS)**: A DoS attack does not result in the theft of information, but in an attempt to block legitimate users of a service from using that service. Typical targets are sites or services hosted in the Internet, and the attack attempts to make the hosted pages (or services) unavailable on the World Wide Web (WWW) [Center, 1998].
- **Distributed Denial of Service (DDoS)**: DDoS is similar to DoS. The main difference between DoS and DDoS is that the DoS is done from one single machine, while DDoS is performed from multiple points simultaneously. The DDoS attack has the same goal of the DoS, turning a machine or network resource unavailable for a time period. Usually, the DDoS is more effective than DoS because it can use up to thousands of computers to attack a particular machine [Joshi et al., 2012]. Typically it occurs when a large number of Internet packets from compromised hosts (zombies) flood the bandwidth or resources of a single target (victim).
- **Sniffer Attack**: This attack is done by a computer program or a hardware that can intercept and log traffic passing over a network or part thereof. Its initial purpose was to help in managing and identifying network problems [Kotz and Essien, 2005]. Sniffers are used by hackers to capture sensitive network information, such as passwords and account information.
- **Remote to Local (R2L)**: In this attack, the intruder does not have an account on the victim machine. Therefore he tries to exploit the system vulnerabilities in order to control the remote machine through the network as a local user.
- **User to Root (U2R)**: This attack presents a particularity, since the attacker has local access to the victim machine. However, he tries to gain super user privileges without authorization.
- **Probe**: The intruder in a Probe attack tries to gain information about the target host. For instance, an attacker attempts to gather useful information about machines and services available on the network in order to facilitate looking for exploits.
- **Brute Force**: Consists of an attacker trying to guess a password via terminal, usually with the support of automatic scripts and dictionaries.
- **Flash Crowd**: This is not exactly an attack, but the anomalies generated for flash crowds are quite similar to DDoS attack [Li et al., 2009]. Flash crowds are large surges of legitimate traffic focusing on some specific sites, or new software release available on Internet over the relatively short period of time.

In cloud computing these kinds of attacks are even more dangerous and may negatively affect its users in two distinct ways. First, imagine an attacker who has obtained unauthorized access to a cloud infrastructure. He will be able to

trigger an attack with large proportions once this malicious action may be sent by hundreds of virtual machines at the same time. Secondly, the service provider must guarantee that management and traffic control are performed efficiently. If the provider fails in this task the consequences can be severe. From an irregular traffic control, the provider may be mistaken when dealing with a legitimate big request for resources. The provider traffic control system can relate this to an attempted of malicious use. As result, a large number of legitimate users would see their resources blocked and suffer the same effects of a DoS attack.

After addressing the main types of attacks and anomalies to which a network may be subject, it will be presented methods to detect such malicious actions. In this context, intrusion detection systems are one of the most used tools to increase network security.

2.3 Intrusion Detection System

IDSs are complex tools that include a number of concepts and techniques that may differ, depending on the situation. In order to clarify the understanding about IDS and the main types of approaches they use to identify attacks, the definition and a taxonomy are provided.

2.3.1 Definition

An IDS usually relies on two main approaches to detect intrusions that differ in the way the data is analysed and processed. The first approach corresponds to a search for evidence of an attack based on signatures of other similar attacks while the second approach consists of a search for deviations from the appropriate behaviour found in periodic observations of the system. The principal advantage of the signature-based detection method is that it leads to a low number of false alarms. However, signature-based IDSs are not able to detect new or variant forms of known attacks. One of the benefits of anomaly-based detection is that a new attack for which a signature does not exist can be detected if it occurs outside of the regular traffic patterns. In this thesis, we focus on the second class of IDS to detect threats to the network traffic in the cloud environment. Whenever such an event occurs, one entity or a local security office takes notice and should take appropriate measures, e.g. ousting the intruder or exposing it to proper external authorities.

The process of detecting attacks performed by an IDS comprises three key activities: collection, analysis and response. The collection corresponds to obtaining data from the monitored system. The collection of information may be made directly, through software or hardware called collector. The analysis consists in processing the collected data trying to identify the occurrence of an intrusion. There are two main approaches for data analysis: signature-based and anomaly-based (see also Section 2.3.2). The response is a set of actions that

the IDS performs when it detects an intrusion. Typically, the IDS generates alarms and reports, but may also be configured for automatic intervention in case of intrusion. The alarms generated in the response phase can be classified into four different states: true positive, false positive, true negative and false negative. Figure 2.3 illustrates and describes each of these situations (adapted from [Estevez-Tapiador et al., 2004]).

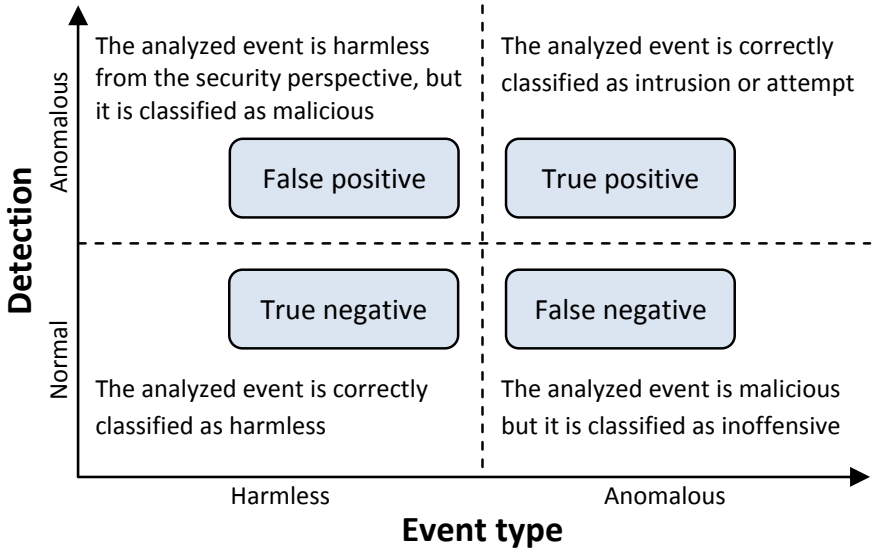


Figure 2.3: Types of response from an IDS

2.3.2 IDS Taxonomy

A classification of IDSs is important to help us understand the existing approaches, identify new challenges and aspects to be improved [Mallissery et al., 2011]. Despite several studies suggest some kind of classification [Sabahi and Movaghar, 2008, Zhu and Sastry, 2010, Debar et al., 1999], in this thesis it is presented a taxonomy in order to organize and detail the key advantages and disadvantages arising from each approach.

Figure 2.4 depicts such taxonomy, identifying features such as information source, detection approach, architecture and type of intruders. The aim of this classification is to identify important issues that characterize an effective IDS, regardless of what mechanism it is based on.

In the following we present a detailed description of each components illustrated in Figure 2.4.

Information Source

Although information may come from many different sources (host, network, sensors or exclusively by applications) [Vokorokos and Balaz, 2010], usually an IDS is classified according to two categories: Host-based Intrusion Detection System (HIDS) and Network Intrusion Detection System (NIDS).

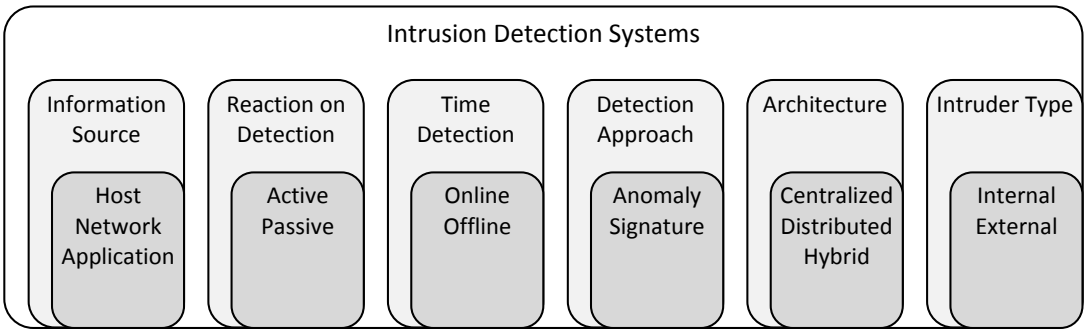


Figure 2.4: Taxonomy of Intrusion Detection Systems

The HIDS is placed on one device such as server or workstation, where the data is collected locally to the machine. The analysed data can come from different places (system files and logs). NIDS are, in turn, used to monitor and analyse network traffic in order to avoid or hamper attacks against networks infrastructure. NIDS are also referred as "packet-sniffers", because they capture packets passing through the communication medium.

Reaction on Detection

When identifying abnormal behaviour, an IDS could act in two distinct ways: passively or actively. In both cases an alarm is triggered, the difference is what the IDS will do after the alarm. The passive IDS will wait for expert human intervention, while in an active IDS a set of pre-established rules will be triggered automatically upon alarm.

Time to Detection

Regarding the runtime, there are two approaches that can be used to detect an intrusion (or intrusion attempt), they are: offline and online. The offline approach is used on two occasions: when we need to test the efficiency and effectiveness under specific conditions or in the training phase of an IDS. However, in general an IDS must operate online since its purpose is to identify attacks in real-time.

Detection Approach

Intrusion detection systems usually rely on two main approaches to identify intrusions that differ in the way data is analysed and processed. The first approach corresponds to search for evidence of an attack based of signatures of other similar attacks, while the second approach consists in searching for deviations/anomalies to the normal behaviour through periodic observations of the system [Mallissery et al., 2011].

A key advantage of the signature-based detection method is that signatures are easy to develop and understand. Moreover this approach presents a low number of false alarms. However, signature-based IDSs also have their disadvantages. Namely, they are only able to detect known attacks and a signature must be created for every attack.

Anomaly-based detection has the advantage that a new attack for which a signature does not exist can be detected if it falls out of the normal traffic patterns. However, defining rules for what constitute an anomaly is a difficult task. For each protocol under analysis, new rules must be defined, implemented and tested for accuracy [Modi et al., 2013].

Architecture

This feature relates to the location where the information from the traffic in question will be collected and processed. For instance, if the gathering, storage and processing are performed on a single component, we say the architecture is centralized. We say that the architecture is distributed when the gathering, storage and processing are performed by several entities in a distributed manner. Finally, an hybrid approach corresponds to a mix between the centralized and distributed methods, eventually following a hierarchy structure.

Intruder Type

IDS can also be classified with respect to the type of attacker they address (internal or external). External attackers do not have legal access and target machines in an unauthorized way. Internal attackers can be of two types: those who masquerade as another user or those who act clandestine by taking advantage of the possibility of turning off the audit control mechanisms [Sobh, 2006].

An IDS provides a wealth of information about network behaviour. This information can be seen as a fingerprint of the network that represents its history regarding intrusion attempts suffered. Based on this data it is possible to define proper metrics to infer the level of network security. Moreover, storing and analysing this data enable the cloud provider taking advantage of privileged information of the network. For example, by means of a similarity analysis between the features extracted from the IDS, it is possible to decrease the number of alarms and reduce the network data traffic to manage [Hudic et al., 2017]. More importantly, via the similarity analysis, it is possible to define the severity of the alarm according to the severity level predefined by the operator or by the historical data of attacks. Next section will provide general concepts about similarity.

2.4 Similarity

Cloud computing and IDSs involve information collected from several sources, for instance, infrastructure, platform for software development or applications. In light of this, a mechanism able to aggregate information and support management tasks is needed.

Similarity analysis is a technique which allows us to assess whether given features are considered similar or dissimilar according to the characteristics that describe them. Basically, there are two properties related to the similarity measures: the level and the commutativity. The similarity level relates to how much an entity X is similar to the entity Y, it ranges from 0 to 1. When two entities are identical the similarity level scores the minimum value, zero. Commutativity determines that the similarity level between X and Y is equal to the similarity level between Y and X.

Assessing similarity between features is a central issue in many research areas such as face recognition [Chopra et al., 2005], linguistic [Mohler and Mihalcea, 2009] and management systems [Cilibrasi and Vitanyi, 2007]. The importance of finding suitable similarity measures cannot be overemphasized [Cilibrasi and Vitanyi, 2007]. The choice of similarity measures depends on the measurement type or representation of the features. The distance between two entities can be used to indicate how similar they are and can be defined by the number of operations to convert an entity X in an entity Y.

There are several approaches to measure similarity, for instance: Manhattan distance, Chebyshev and Euclidian distance [Cha, 2007]. Manhattan distance depends on the rotation of the coordinate system but does not depend on its reflection about a coordinate axis or its translation. When Chebyshev is applied in one dimension, the distance is just the absolute value of the differences. If Chebyshev is applied for two dimensions, the distance is equivalent to the planar Manhattan distance. Euclidean distance measures the similarity between two points in a Euclidean space as illustrated in Equation 2.1:

$$Sim_{ED}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

The Tanimoto coefficient is another metric used to measure similarity level in finite sample sets based on the size of the intersection divided by the size of the union of the sample sets, as illustrated in Equation 2.2. The Tanimoto coefficient is a variation of the Jaccard index, the difference being that Tanimoto can be applied to non-binary or quantitative data (groups) while Jaccard can be applied to binary values [Leydesdorff, 2008].

$$Sim_{TM}(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i y_i} \tag{2.2}$$

2.5 Summary of the Chapter

This chapter presented a general background required to provide a theoretical basis for better characterization of the environment around cloud computing and the solutions proposed. More specifically, a description of the environment around cloud computing from The National Institute of Standards and Technology point of view is presented. Also, the characterization of network traffic in the cloud and the main issues that may harm its operation is introduced. A description of IDSs focused on virtualized environments and similarity concepts in order to aggregate alarms are presented. Furthermore, these definitions, as well as other concepts presented in this chapter, are used for designing the prediction model, the security mechanism and the similarity approach in the subsequent chapters.

Chapter 3

Network Traffic Prediction in the Cloud

Prediction is very difficult,
especially about the future.

(Niels Henrik David Bohr)

Contents

Predicting the inherent traffic behaviour of a network is an essential task, which can be used for various purposes, such as monitoring and managing the infrastructure of the network. However, the recent surge of dynamic environments, such as Internet of things and cloud computing have hampered this task. This means that the traffic on these networks is even more complex, displaying a nonlinear behaviour with specific aperiodic characteristics during daily operation. Traditional network traffic predictors are usually based on large historical data bases which are used to train algorithms. This may not be suitable for these highly volatile environments, where the strength of the force exerted in the interaction between past and current values may change quickly with time.

3.1 Introduction

Cloud computing is a basis for providing benefits well beyond Information Technology (IT) cost savings. It comprises a set of service models to scale on-demand, such as Infrastructure as a Service, Software as a Service or Platform as a Service. These services can be offered to a wide range of clients through an organization called cloud provider [Mell and Grance, 2011]. Cloud computing is at the core of the always connected paradigm, in which users access their data any time and anywhere requiring only a device with Internet access. This has increased the continuing demand for ubiquity and more powerful resources, making cloud computing a solution that perfectly matches need with efficiency.

On a daily basis, the cloud provider has to deal with a huge number of devices and virtual machines so that it can handle all the assets of its network infrastructure. It is increasingly being found that neglecting this area of management can cause irreparable economic damage to businesses and their customers [Owezarski et al., 2013]. In light of this, the network administrators of these cloud-supporting networks must monitor and analyse these networks so that relevant information about network traffic can be collected and used to support decision-making. After all this information has been gathered, it is possible to identify and analyse suspicious network traffic patterns. These patterns can help in planning strategies to prevent similar problems (*e.g.* anomalies in the network traffic) from occurring in the future [Dainotti et al., 2012].

Analysing network traffic is a means of facilitating the monitoring and management of computer networks. In this context, a network traffic predictor is a tool that uses accumulated statistics over time to make inferences about the future behaviour of network traffic [Prangchumpol, 2013]. Thus, when an abnormality is forecast, the network administrator will have time to act even before the problem arises. For this reason, network traffic prediction has been receiving a great deal of attention from the scientific community. In addition, network traffic prediction is also important in other fields such as: traffic shaping for

improved Quality of Service [Rahmani et al., 2009], prediction of bandwidth requirements [Sang and Li, 2002], conceiving more accurate simulation models [Papadopouli et al., 2006], admission control [Zhou et al., 2006], and adaptive applications [Salah et al., 2016].

3.1.1 Requirements for Cloud Network Traffic Prediction

Characterizing and monitoring network traffic is becoming a more complex task, particularly with the surge in traffic arising from the huge number of individuals and machines that are permanently connected to the Internet. The challenge is even greater in cloud computing because its traffic is apt to undergo sudden changes [Ballani et al., 2011, Vieira et al., 2010], and the elastic and scalable nature of cloud environments may be easily confused with traffic anomalies [Plonka and Barford, 2009].

Network traffic prediction requires an accurate model, which can capture the statistical features of the real condition of the traffic. However, the degree of accuracy needed for predicting the network traffic deteriorates quickly as the historic interval increases [Zhang et al., 2013], and thus calls for a model that is suited to the dynamics of cloud traffic. These observations suggest that this kind of network traffic is different from traditional IT network traffic, and requires special attention in some aspects. This implies that the cloud network traffic predictor must not only have traditional features such as accuracy, but also some other essential requirements for the cloud environment. These include the following:

1. **Low historical dependency**: Models for predicting data traffic usually take into account large amounts of historical data. However, employing these models is not the most suitable approach for carrying out the traffic prediction of cloud computing systems, because the network baseline does not have the same periodic behaviour as traditional networks [Xiong et al., 2014];

2. **Low complexity**: Several prediction models have been proposed in the literature that can be employed in various network traffic environments. However, most of them are not appropriate for dealing with a large amount of information in a short period of time and keeping a low computational complexity [Buyya et al., 2010, Lim et al., 2000];

3. **Online prediction**: Offline predictions are usually made to test the predictor efficiency under specific conditions and in the training phase. However, an effective monitoring of cloud computing networks must be constantly carried out to address detection issues as they occur or even beforehand. This statement applies to online traffic prediction mechanisms [Sang and Li, 2002].

In short, from this set of requirements it is possible to select a group of candidate

predictors that is suitable for cloud network traffic prediction. On the basis of this scenario, we consider that the choice of an ideal predictor model for cloud network traffic will involve a tradeoff between prediction error, historical data dependence, computational costs, and timely response.

3.1.2 Contributions and Outline

Most research studies on network traffic prediction have focused on classical methods that rely heavily on historical data such as time series, neural networks and machine learning. However, there is still no consensus among the research community about which model is best suited for cloud network traffic prediction.

Usually, there is only a fine line separating concerns about high accuracy from computational costs, and sometimes it is difficult to determine where the border line should be drawn. The challenge in cloud network traffic prediction is to minimize the computational cost as much as possible, while keeping acceptable levels of accuracy. This is not a trivial issue, since most of the current prediction models are not able to keep a low computational complexity while dealing with a high degree of workload information in a short period of time. The main problem with these models is the increasing computational overhead in accordance with the size of the input data. For instance, approaches based on large historical dependency could obtain a slightly better degree of accuracy than other models based on short historical dependency. However, when compared with local analysis approaches, they have a much higher computational complexity to compute the predictions [Buyya et al., 2010].

To address these issues, we now review the state of the art and provide a taxonomy for network traffic prediction models. This lays the foundation for selecting a set of prediction models that is suitable for the cloud environment. Thus, an analysis mechanism through which a standardized approach is adopted for evaluating the candidate predictor models using real traces is presented. The remainder of this chapter is organized as follows. Section 3.2 presents a review of state of the art and a taxonomy with the most prominent related studies on network traffic prediction models. Section 3.2.4 details prediction methods deemed applicable to the highly dynamic cloud computing environment requirements. Section 3.3 describes the methodology used to evaluate the prediction mechanisms, whilst Section 3.4 presents the evaluation and discusses the results. Section 3.5 summarizes the chapter.

3.2 State of the Art

The classification of network traffic prediction models is a useful means of helping us understand the existing approaches, addressing new challenges and find-

ing out features that need to be improved [Hoque et al., 2014, Whaiduzzaman et al., 2014]. In this section, the main concepts in network traffic prediction and an outline the most commonly used techniques are presented. In addition, a taxonomy for assessing and listing the main benefits and drawbacks of each prediction mechanism is proposed. Figure 3.1 depicts the taxonomy that we will now describe in detail.

The choice of the forecast model should take into account the purpose of the prediction as well as the characteristics that reflect the main properties of the data, such as trends, seasonality, patterns of variation and time dependence. After that, we are able to store, organize and analyse the data, and make inferences about the future behaviour [Lu et al., 2014]. There are several predictor models for network traffic in the literature. On top of the taxonomy, the models are divided into two distinct categories: Pattern Recognition and Time Series.

3.2.1 Pattern Recognition

Pattern Recognition is an attempt to model the human brain, which means that pattern recognition basically involves learning from experience. In addition, the model must be able to maintain a good accuracy, which is not simple in short time, since the precision of these techniques is dependent on sufficient historical data being available. Furthermore, predictors based on pattern recognition use quantitative information. The term "quantitative" refers to a type of information based on quantifiable data (objective properties). A prediction system based on pattern recognition requires taking note of many issues such as feature extraction, selection and cluster analysis. In this sense, Artificial Neural Networks (ANN) [Cortez et al., 2006], Bayesian Networks [Dalmazo et al., 2011],

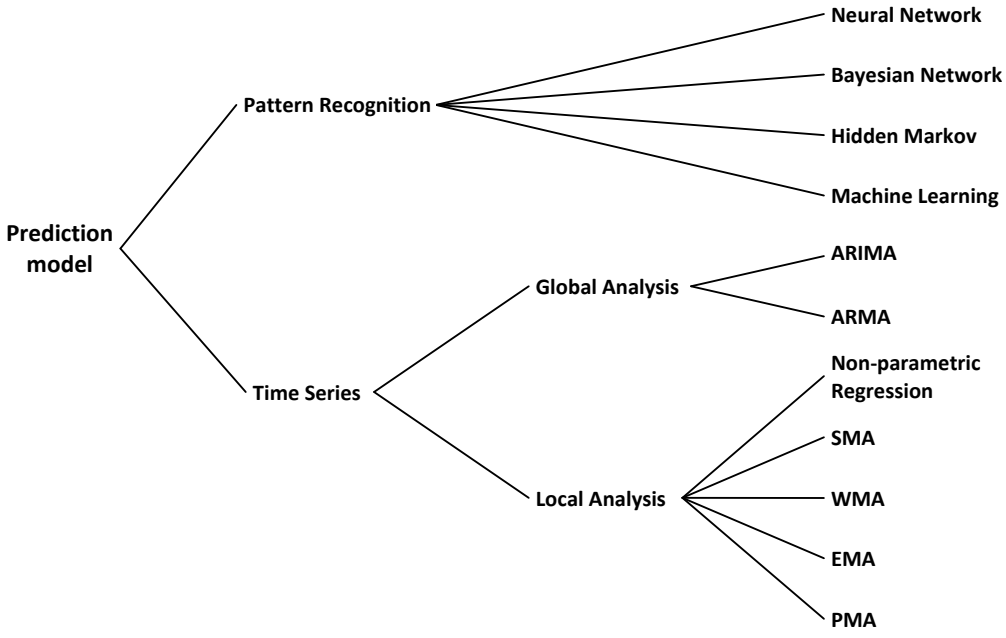


Figure 3.1: Taxonomy of prediction models

Hidden Markov [Dainotti et al., 2008], and Machine Learning [Erman et al., 2006] techniques have been receiving increasing attention in the field of prediction models.

Chen *et al.* use genetic programming to build a Flexible Neural Tree (a flexible multi-layer feed-forward neural network) for online network traffic prediction [Chen et al., 2012]. This approach was adopted to obtain a better understanding of the main features of traffic data. Moreover, the proposed method is able to forecast small time-scale traffic measurements and can reproduce the statistical features of real traffic measurements. However, to achieve reliable results, it requires an initial input that is dependent on the characteristics of the data under evaluation. Hongying also presents a neural network (back-propagation neural network) trained by a modified quantum-behaved particle swarm optimization in order to predict the dynamic network traffic flow [Hongying and Li, 2013].

Auld *et al.* propose a tool based on Bayesian Networks to support Internet traffic identification [Auld et al., 2007]. This model is useful for recognizing the future behaviour of the network traffic by taking into account past experience. By means of this approach, it was possible to classify several application types without a source or destination address, by using samples of traffic that could enable the categorization to be made on the basis of commonly available information. However, the authors concluded that the accuracy of the classification declines quickly over time as the nature of the Internet traffic changes. This limitation hampers its applicability in cloud computing.

Dainotti *et al.* resort to a statistically-based approach to classify network traffic by combining network traffic with different categories of network applications [Dainotti et al., 2008]. Traffic prediction is performed by taking into account characteristics such as inter-packet times and payload size, as well as their temporal correlation. The proposed solution involves a classification of the packet-level traffic based on a Hidden Markov Model. The aim of this work is to use the obtained classification to offer different levels of Quality of Service (QoS) that depend on the class of traffic. It also assists in the enforcement of security policies for different applications and the identification of malicious traffic flows. All the evaluations are carried out by analysing offline traffic from different network topologies.

Methods from the machine learning theory [Erman et al., 2006] involve developing systems with the capability of learning from the recognition of old patterns. Nguyen and Armitage conducted a survey with many applications for machine learning [Nguyen and Armitage, 2008]. Among them, machine learning also has the potential to support network traffic prediction. In this context, the application of machine learning involves several stages. First, the features must be selected to feed the algorithm. After that, these features are assigned to flows that are calculated over multiple packets. The machine learning classifier creates rules by linking the features with well-known traffic behaviour. Finally,

the algorithm is able to predict the network traffic based on previous learned rules.

Bermolen and Rossi propose a solution based on a support vector machine to solve the problem of link load forecasting based only on their past measurements [Bermolen and Rossi, 2009]. This machine learning approach showed a noticeable variation in the prediction performance carried out for a given training set size. When the training set is smaller than 50% of all the historical data, the authors consider the model to be under-trained and the prediction error is large, and thus precludes its application in real-time.

In general, pattern recognition approaches are performed in two stages: training and forecasting. The training and forecasting tasks have to be performed in different time-scales. Although the model training is an offline operation, it has to be done periodically so that a strict degree of accuracy is maintained, while the forecasting has to be made continuously and online [Bermolen and Rossi, 2009]. If these two stages overlap, there will be an increase in the workload, thus making it unsuitable to online traffic prediction in dynamic environments such as cloud.

3.2.2 Time Series

In general terms, time series are an ordered sequence of the values of a variable at equally spaced time intervals. Time series techniques take account of the internal features in the data flow such as autocorrelation and seasonality. On the basis of these features, it is possible to estimate the future behaviour of the data flow from a set of past data. In this context, the scope of the analysis enables us to classify time series approaches into two types: global and local.

Global Analysis

A global analysis is based on assumptions about the shape of the data that can be numerically described by taking into account all the values of the population. More importantly, the data sample has to represent the distribution of the population in question, and the normal distribution is usually adopted for this kind of approach. In this context, a global analysis procedure allows more conclusions to be made about the data. However, there is a limitation in the approach that follows the normal distribution since the model does not work properly for small sample sizes ($n < 30$). Nevertheless, a local analysis procedure is a good means of overcoming this drawback [Freedman, 2009].

If the series follows a repetitive pattern which is believed to be constant at each slot of time (for instance, from week to week or from day to day), a seasonal adjustment may be required to anticipate the behavioural pattern. The advantage of a seasonal prediction model is that it fits the seasonal pattern, by enabling the

periodic variations to be included in the prediction results. The disadvantage is that this type of prediction model requires a time window with a large number of values, and this adds an extra workload to compute the estimation. In this case, two models can be used to forecast the time series, Autoregressive Moving Average (ARMA) [Torres et al., 2005] or its Integrated variant (ARIMA) [Song and Li, 2008].

Sang and Li use the ARMA model to evaluate the network traffic prediction regarding two points of view: (*i*) how far into the future the network traffic can be forecast; and (*ii*) about the minimum acceptable forecast error [Sang and Li, 2002]. In the prediction assessment, the results show a tradeoff between prediction error and control time-scale. Furthermore, the paper shows that the prediction accuracy deteriorates as the slot time size increases.

Moayedi and Masnadi-Shirazi propose a network traffic prediction and anomaly detection model based on ARIMA [Zare Moayedi and Masnadi-Shirazi, 2008]. The ARIMA model gives a description of a stationary stochastic process in terms of polynomials to fit the dataset. In their paper, they decompose the data flow to isolate the anomalies from the normal traffic variation. The authors then try to predict anomalies separately from the normal traffic, and the paper evaluation shows that the anomalies have been successfully detected. Their work was evaluated with synthetic data and depends on large historical data. Zhao [Zhao, 2009] uses wavelets for analysing time domain signal of the time series for improving the prediction accuracy level. However, the computational complexity in predicting each wavelet coefficient is high.

Rajnish Yadav and Manoj Balakrishnan present a comparative performance evaluation between the ARIMA and an Adaptive Neuro Fuzzy Inference System (ANFIS) [Yadav and Balakrishnan, 2014]. The goal of this work is to model the behaviour of wireless network traffic. In the scenario evaluated, ANFIS shows the best results, but with high computational costs, hampering traffic prediction on virtualized environments.

Wen-Kuang Kuo and Kuo-Wei Wu propose a traffic predictor designed to provide online prediction with the goal of guaranteeing QoS in real-time live video transmission [Kuo and Wu, 2011]. The predictor, based on variable step size least mean square algorithm, achieves high channel utilization and guarantees the QoS requirements for real-time video. However, it obtains information only from the last simple scene, restricting the ability to forecast abrupt changes, common in cloud environments.

Li and Lim identify a noticeable traffic behaviour, which is called “burstiness” or “packet trains”, defined by peak-to-average transmission rate [Li and Lim, 2008]. This behaviour is characterized by a long repetition of time intervals in which firstly no packets are transmitted, and afterwards a wave of packets is sent. In this work, the network traffic is studied from the perspective of fractal time series. Using this approach, it is possible to project time series predicting the future behaviour of the network. This approach is used to study specific

parameters (such as the Hurst parameter) and relies on playback of offline traffic, taking into consideration traffic properties such as long-range dependency and heavy-tailed distribution. These properties relate to large historical data, therefore making this approach unsuitable for real-time traffic monitoring in dynamic environments.

Local Analysis

Local analysis procedures are considered to be more inaccurate than a global analysis because they use less information in making their calculations. The data correlation only relies on local data to make assumptions about the population. For instance, it fits in with the real data using just one subset to develop a function that describes the behaviour or the variation of the data. This is an advantage since the analysis does not require global information to make statistical inferences [Freedman, 2009].

Sometimes, the time series has a linear dependence with regard to its series of values, and shows a constant growth factor. In this case, non-parametric regression method can be regarded as a prediction model type [Zhang and Qi, 2005]. In light of this, regression models may be used for estimating a polynomial function that represents the time series trend. However, cloud computing provides a dynamic environment with a complex network traffic behaviour, and is far from having a linear trend pattern. This means that it requires high degree polynomials to fit the network traffic baseline. To achieve an online prediction with accurate results, the prediction model requires a constant adjustment of a polynomial function. However, this incessant process is expensive [Jin, 2005] and thus causes the regression models to conflict with the *Low complexity* requirement for cloud network traffic prediction.

When a local analysis is expected, a moving average model may compute a local average of data at the end of a time window, on the assumption that this is the best estimate to represent the current mean value around which the data is ranged. The size of the time window can be adjusted dynamically making these models more suitable if the time series change suddenly. A moving average model is closely correlated to a local analysis procedure since it combines simplicity with an attempt to build up a function to describe a local trend.

Papadopouli *et al.* evaluate a set of forecast algorithms to characterize the traffic load in an IEEE802.11 infrastructure [Papadopouli et al., 2006]. Their work describes the Simple Moving Average (SMA) as the unweighted mean of the previous data points in the time series. In addition, SMA is less demanding than more complex predictors, such as ARIMA, which requires more parameters to compute the prediction. They point out some of the advantages of SMA, such as its simplicity, low complexity and ease of application. Together with this, Lee *et al.*, show that SMA also provides a basic and efficient tendency index [Lee et al., 2012].

Li *et al.* study anomaly detection methods for high-speed network traffic. The purpose of this work is to come up with a sensible mechanism for detecting significant changes in massive data streams with a large number of flows. Through a model based on a Weighted Moving Average (WMA), the algorithm estimates the value of the next interval and compares with the real traffic [Li et al., 2012]. After that, all traffic that does not match the reference model is considered to be an anomaly, thus it is able to detect Distributed Denial-of-Service (DDoS) and scan attacks.

Klinker describes mathematical tools to identify and predict market trends. In particular, their study shows that the Exponential Moving Average (EMA) can be used for an effective forecasting of network traffic combined with a local analysis of the historical data [Klinker, 2011].

Chang and Tsai analyse the flow trend of two types of packet flows: inflow and outflow of data packets [Chang and Tsai, 2009]. This work highlights some problems that could occur, namely the volatility clustering problems and its effect on deteriorating the accuracy of short-term predictions. The proposed model is enhanced by Adaptive Support Vector Regression to form a linear combination of two models (Adaptive Neuro-Fuzzy Inference System and Nonlinear Generalized Autoregressive Conditional Heteroscedasticity) in order to not only simplify the complexity of the system, but also improve the prediction accuracy by solving the overshoot and volatility clustering problems. This scheme can act as a core component of network traffic analysis in order to help a web manager in providing network traffic control. Due to the several algorithms and statistical calculations employed, this approach is deemed heavy and requires high processing overhead, therefore not being suitable to cloud computing environments.

As part of the contributions of this thesis, Bruno Dalmazo *et al.* proposed a systematic approach for estimating network traffic by resorting to a statistical method based on a Poisson process (Poisson Moving Average - PMA) [Dalmazo et al., 2013]. In addition, we used a dynamic sliding window size algorithm (DyWiSA) to weight past observations by taking advantage of well-known network traffic features such as short-range dependence [Dalmazo et al., 2014]. These approaches will be presented in more details in the Subsection 3.2.5.

3.2.3 Pattern Recognition vs. Time Series

A comparison between Pattern Recognition and Time Series has already been performed in [Zhang and Qi, 2005] and [Wilamowski, 2009]. Zhang and Qi compare a neural network with an ARIMA [Zhang and Qi, 2005]. In this case, even when the neural network is trained with all original data available, its performance is inferior to the ARIMA model. This shows that neural networks are not able to capture trend variations effectively. Approaches based on pattern recognition, such as neural networks, have a serious drawback: they learn

the training patterns but lose the ability to make generalizations, which means that the model may give inaccurate results for unknown patterns [Wilamowski, 2009].

Another limitation of pattern recognition in network traffic prediction for cloud is the fact that a considerable amount of offline computation is needed to train it properly. In order to make pattern recognition approaches effective, these models must be trained on a representative data set; otherwise, it may not be feasible to achieve a satisfactory degree of accuracy [Akesson and Toivonen, 2006]. For instance, research studies have shown that using 50% of the data for training is not enough to provide accurate predictions [Bermolen and Rossi, 2009]. Other studies show that even using 100% of the historical data pattern recognition achieves worse results than the Autoregressive Integrated Moving Average [Wilamowski, 2009]. Wei Li and Andrew W. Moore show that the labour required in hand-classification process increases along with the size of the training set using Machine Learning approach [Li and Moore, 2007]. In addition, pattern recognition does not meet the *Low historical dependency* requirement, and is computationally expensive, going against the *Low complexity* requirement for cloud computing.

In regarding to time series approaches, Zhani M. F. *et al.* show that enlarging training data set does not really improve traffic predictability using a time series-based approach (ARMA and ARIMA models) [Zhani et al., 2009]. In addition, several weighted sampling schemes can be employed such as: Simple Moving Average (SMA), Weighted Moving Average (WMA), Exponential Moving Average (EMA) or Poisson Moving Average (PMA). The WMA is more sensitive to recent values than a SMA. However, an EMA is usually preferred to a WMA, because its exponentially weighted average does a more sensible work of discounting the older data and its smoothing parameter is continuous since it is readily optimized to each new iteration [Papadopouli et al., 2006]. PMA, which is based on a Poisson process, usually fits the network traffic behaviour better than the other short-term predictors [Dalmazo et al., 2014].

Summing up, as was shown previously, approaches based on pattern recognition and non-parametric regression method are not considered for cloud traffic prediction due to their high complexity and historical dependency. Table 3.1 summarizes several network traffic predictor models regarding desirable requirements for cloud computing environment.

Approaches based on local analysis generally display low levels of historical dependency. This leads to a low complexity solution for traffic prediction by reducing the amount of data required for processing, when compared with the strong historical dependency models. Both of them (local and global analysis), allow online traffic prediction (using DyWiSA for local analysis approaches), but with solutions of different levels of complexity. On the one hand, models with global analysis usually achieve accurate results in prediction. On the other hand, models based on local analysis provide a solution with lower computa-

Table 3.1: Summary of related work

Approach	Desirable features			
	Low historical dependency	Low complexity	Online prediction	High accuracy
SMA – [Papadopouli et al., 2006]	√	√	×	×
SMA – [Lee et al., 2012]	√	√	×	×
WMA – [Li et al., 2012]	√	√	×	√
EMA – [Klinker, 2011]	√	√	×	√
PMA – [Dalmazo et al., 2013]	√	√	×	√
DyWiSA – [Dalmazo et al., 2014]	√	√	√	√
ARMA – [Torres et al., 2005]	×	×	√	√
ARIMA – [Zare Moayedi and Masnadi-Shirazi, 2008]	×	×	√	√

tional complexity (see Subsection 3.2.6).

In this thesis, it is compared several local and global analysis approaches by taking into account the requirements of the cloud computing environment (highlighted in Table 3.1). In particular, it provides a systematic methodology for evaluation of predictors, that makes it easier to compare different models in terms of accuracy, historical dependency, time and computational overhead.

3.2.4 Prediction Approaches

The model of behavioural prediction can usually be characterized by using a time series of historical values (*e.g.* network traffic packets). With the aid of historical traffic data, it is possible to predict future cloud network traffic. Hence, future values can be forecast based on a correlation between the variation of the values at the time and in its current state.

As noted earlier, there is a large and growing body of literature that regards network traffic prediction as a means of facilitating the monitoring and management of computer networks [Chunlin and Layuan, 2014]. Although most research studies employ classical methods that are largely based on historical data such as time series and neural networks, some recent works [Ballani et al., 2011, Vieira et al., 2010] show that long-term historical dependence is not suited to cloud computing due to the high volatility of this environment. In this section we consider previous prediction models which carry out forecasting on the basis of local and global data analysis.

3.2.4.1 Simple Moving Average - SMA

The Simple Moving Average (SMA) is the most popular of the moving averages used for predicting based on local analysis. It is calculated as the unweighted mean of the previous n data values as shown in Figure 3.3(a).

The term moving is used because for each new slice of time, the oldest data value is dropped from the sliding window as soon as the new value becomes available. An example of a simple equally weighted moving average for a sample of n values v is the mean of the previous n values of the time series, as we can see in Equation 3.1:

$$SMA = \frac{\sum_{i=1}^{n}(v_i)}{n} \tag{3.1}$$

3.2.4.2 Weighted Moving Average - WMA

When using a moving average technique as described in the previous subsection, each of the coefficients used to compute the predicted value is weighted equally. However, it might sometimes be useful to put more weight on recent observations that are closer to the time period being predicted. In this case, we use a weighted moving average technique. As a general rule, a weighted average is any average that uses several coefficients to provide various weights for data at different positions in the sliding window. Figure 3.3(b) shows an example of this function inside a sliding window.

In this work, WMA specifically refers to weights that increase in arithmetical progression. In a sliding window with n samples, in the WMA, the newest value has weight w_n, the second newest w_{n1} and so on until the oldest value goes down to zero. The sum of the weights in a WMA have to be 1. In a normal case, for each weight w and value v in the sliding window, the denominator will always be the sum of the individual weights, as can be seen in Equation 3.2:

$$WMA = \frac{\sum_{i=1}^{n}(w_i * v_i)}{\sum_{i=1}^{n}(w_i)} \tag{3.2}$$

3.2.4.3 Exponential Moving Average - EMA

The Exponential Moving Average (EMA) is an exponentially weighted moving average that applies weighting coefficients which decrease exponentially in the course of time inside a sliding window. The weighting for each older value decreases exponentially, but never reaches zero. Figure 3.3(c) shows an example of the weight decrease.

In a similar way to other moving average techniques, EMA must only be used for a set of data without seasonal behaviour [Chatfield and Yar, 1988]. This moving average technique reacts faster to recent value changes than with a simple moving average that attributes more weight to the latest changes and

less to the changes that lie further away. The formula for calculating EMA is given by Equation 3.3:

$$EMA = \frac{\sum_{i=1}^{n} (exp^i * v_i)}{\sum_{i=1}^{n} (exp^i)} \tag{3.3}$$

3.2.4.4 Autoregressive Moving Average - ARMA

With regard to a time series analysis, the ARMA model provides a description of a stationary stochastic process in terms of two polynomials – first an autoregressive (AR) and, secondly, a moving average (MA). The ARMA is usually referred as the ARMA(p,q) model where p is the order of the autoregressive part and q is the order of the moving average part [Sang and Li, 2002].

Given a time series data, the ARMA model is able to characterize and then forecast future values within the time series and can be defined as shown below:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{3.4}$$

where c is a constant, φ_1, ..., φ_p are the autoregressive parameters of the model, θ_1, ..., θ_q are the moving average parameters of the model and the random variable ε_t, ε_{t-i} are white noise error, usually assumed to be a Gaussian distribution with zero mean [Wei, 1994].

3.2.4.5 Autoregressive Integrated Moving Average - ARIMA

The Autoregressive Integrated Moving Average - ARIMA is a model powered with a time series and used either to obtain a better understanding of the data behaviour or to forecast future points in the series [Makridakis et al., 2008]. ARIMA is a predictor model which is a generalization of an ARMA model [Yin et al., 2005].

The ARIMA model is referred to in the literature as ARIMA(p,d,q) where the parameters p and q have the same meaning as in the ARMA model. The difference is the d parameter which refers to the order of the integrated part of the model. All the parameters must be non-negative integer numbers. Given a time series of data X_t where t is an integer index and the X_t is any real number, an ARIMA(p,d,q) model is given by:

$$\left(1 - \sum_{i=1}^{p} \phi_i B^i\right) (1 - B)^d X_t = \delta + \left(1 + \sum_{i=1}^{q} \theta_i B^i\right) \varepsilon_t \tag{3.5}$$

where B is the backshift operator, *i.e.*, the previous element of a time series (X_{t-1}), ϕ_1, ..., ϕ_p are the autoregressive parameters of the model, θ_1, ..., θ_q are the moving average parameters of the model and the random variable ε_t is the white noise error, as described in the ARMA model.

3.2.5 Poisson Moving Average - PMA

In order to provide another alternative more suitable for cloud environments, this section proposes a new Moving Average approach based on the Poisson distribution. The Poisson distribution is a natural choice for describing the probability of the number of occurrences in a field or continuous interval (usually time or space), such as number of defects per square meter, number of accidents per day or number of network packets per minute [Dalmazo et al., 2013]. We note that in our field of application the unit of measure (time) is continuous, but the random variable (number of packets) is discrete. In other words, a Poisson process is used to determine the probable minimum and maximum number of transactions that can occur within a given time period, from a series of discrete values [Gardiner et al., 1985].

Algorithm 3.1. Poisson Procedure

Input : Lambda parameter
Output: Poisson slices vector

```
1: Start
2:     procedure POISSON(lambda)
3:         vector vPoisson
4:         var poissonSlice
5:         for (i = (lambda); i > 0; i − −) do
6:             poissonSlice ← e^{−λ}(λ)^i / i!
7:             vPoisson.add(poissonSlice)
8:         end for
9:         return vPoisson
10:    end procedure
11: End
```

Let k be a discrete variable taking the values 0, 1, 2, 3, ... ,∞. If k represents a time interval following the Poisson process with parameter $\lambda > 0$, then:

$$P[N(t) = k] = \frac{e^{-\lambda t}(\lambda t)^k}{k!} \tag{3.6}$$

where the Poisson parameter lambda (λ) represents the total number of events (z) divided by the number of units (n) of data ($\lambda = z/n$). The unit forms the basis or denominator for calculation of the average. A Poisson process is described in Algorithm 3.1.

In order to reduce the complexity of predicting network traffic, we consider

time-bounded past information by means of a sliding window. This window is applied by weighting past observations according to a Poisson distribution with λ sampled values. The example illustrated in Figure 3.2 shows a static sliding window with size three. Each value of the data flow is weighted with a portion of the Poisson process, and the most recent value of the data flow receives the highest Poisson slice/weight. Thus, at time t, the sliding window has a set of three values $\{0, 0, 3\}$. In the next turn, at time $t + 1$ the next value to enter inside the window will be 1, and when this occurs the oldest value (0) leaves the sliding window. This process will be repeated as long as there is a data flow from the network.

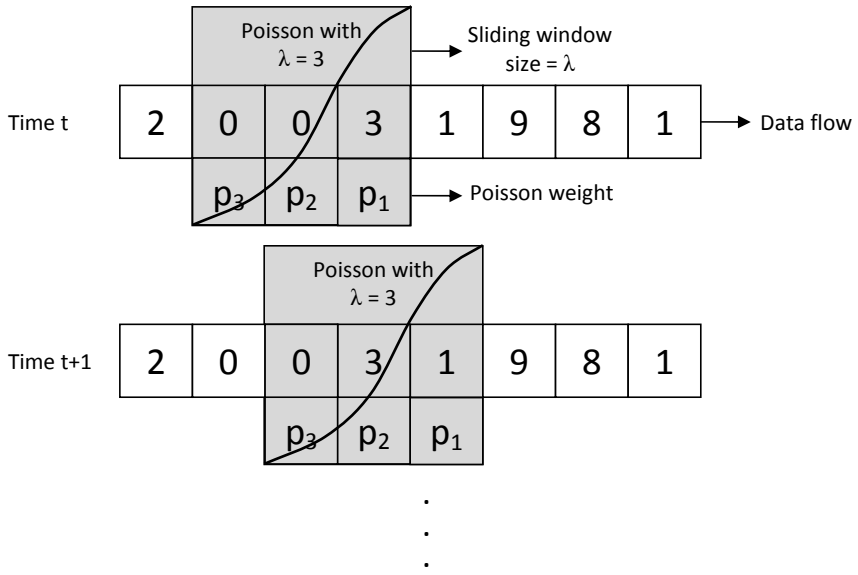


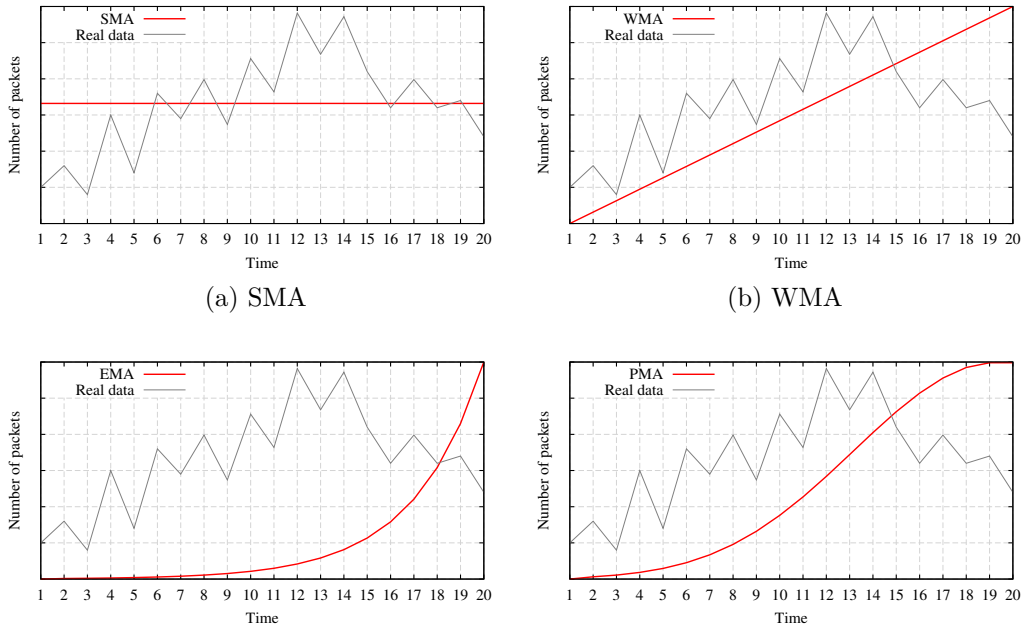
Figure 3.2: The operation of sliding window

The Poisson distribution is represented by a discrete function observed over time t, the distribution starts at $t = 0$ and, when $t = \lambda$ the function has its maximum. For a time interval of size $t = \lambda$, let a truncated Poisson distribution of size n be represented by values p_1, p_2, ..., p_n. To determine a prediction of the expected value of network traffic at time t, $\tilde{y}_t$, our solution uses a Poisson distribution truncated from $t = 0$ to $t = \lambda$. Then we weight previous values according to the Poisson distribution as follows,

$$\hat{y}_t = p_1 y_{t-1} + p_2 y_{t-2} + ... + p_\lambda y_{t-\lambda} \tag{3.7}$$

where $\hat{y}_t$ represents the result of the prediction process, namely, the expected value of the network traffic. Equation 3.8 summarizes this process, whilst Figure 3.3(d) presents the behaviour's function inside the sliding window.

$$PMA = \sum_{i=1}^{\lambda} p_i y_{t-i} \tag{3.8}$$



(a) SMA (b) WMA

(c) EMA (d) PMA

Figure 3.3: Behaviour of the models inside the sliding window

3.2.6 Computational Complexity Comparison

As presented in this chapter, there are various algorithms that can be used for solving the problem of predicting network traffic. However, there is an amount of resources required for running it [Cormen et al., 2001]. This computational complexity comparison deal with the number of elementary operations necessary to run the algorithms and time consumption.

The performance analysis is conducted for the average volume of network traffic, which defines the next sliding window size. Once the sliding window size is set, it is possible to estimate the number of operations that will be performed by the predictors.

For the purpose of our analysis, we consider that the operation time to read the elements inside the sliding windows yields negligible complexity (or the ever constant $O(1)$). The SMA-based predictor has a function that assigns equal action to all the elements of the sliding window, thus, the number of operations increases linearly with the size n. Other predictors evaluated in this work have special values to weight the elements inside the sliding window. For example, WMA uses a linear function that increases from zero to its maximum size, as illustrated in Figure 3.3(b). The EMA approach uses an exponential function, but the number of calculations and the window size increase at the same rate than other moving average models presented here. Therefore, EMA has $O(n)$ complexity. PMA uses a weighting function based on the Poisson process and like EMA, PMA also has $O(n)$ complexity, as illustrated in Table 3.2. Finally, the computational complexity for the ARMA predictor, as well as the ARIMA model, is $O(n^2)$ [Feng and Shu, 2005, Monahan, 2005, Krunz and Makowski,

1998]. The behaviour of all these local analysis approaches is illustrated in Figure 3.3, which shows a window for n = 20.

Table 3.2: Summary of the computational complexity comparison

Model	Complexity
SMA	$O(n)$
WMA	$O(n)$
EMA	$O(n)$
PMA	$O(n)$
ARMA	$O(n^2)$
ARIMA	$O(n^2)$

It is worth noting that this performance evaluation considers the recursive use of the algorithms. In other words, from 1 to i, each element e_i uses all the results from the previous e_{i-1} values. In summary, the analysis of algorithms is quite important since it is a means of obtaining performance evaluation criteria that are independent of the technology adopted or programming language used [Garey and Johnson, 1979].

3.3 Mechanism for Evaluating Traffic Predictors

The purpose of this analysis mechanism is to provide a standardized approach to evaluate the predictor models from real historical data of cloud-based network traffic. Figure 3.4 depicts the steps of our mechanism, by highlighting its main conceptual components, the personnel involved, and their interactions.

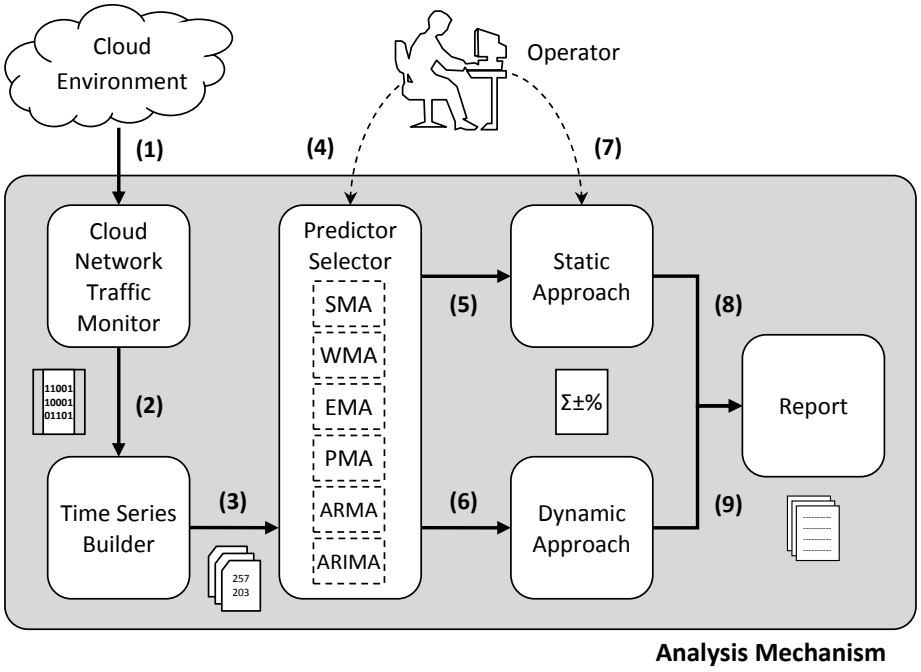


Figure 3.4: Elements of the proposed mechanism and interactions

Real-time cloud traffic data (Flow 1) is constantly being gathered from the cloud environment by the *Cloud Network Traffic Monitor*. This data is subsequently processed by *Time Series Builder* to retrieve and organize relevant information (*e.g.* timestamp, number of network packets, and the protocols involved) for the following prediction (Flow 2). Once the *Time Series Builder* process is complete, the time series is ready to be read (Flow 3). In addition, an *Operator* may interact with the *Predictor Selector* to determine which predictor model will perform the prediction (Flow 4).

Following this, the time series (combined with the predictor model) is sent to the next two components (Flows 5 and 6). At this point, the prediction will be performed fully- and semi-automated (Dynamic and Static approaches). The latter approach is subject to an *a priori* data flow analysis to enable it to produce the statistics about the data or intervention by an *Operator* (Flow 7) and thus aid the process (manually set up with the aid of the statistical parameters). As the output of these processes, the *Report* component will generate a description regarding the accuracy of both prediction approaches (Flows 8 and 9).

Having presented a general outline of the analysis mechanism, in the following subsections there will be a more detailed description of: (i) the monitoring process of cloud-based network traffic, (ii) creating a time series, and (iii) the Static and Dynamic approaches to predicting network traffic in a cloud computing environment.

3.3.1 The Cloud Network Traffic Monitor

Cloud computing provides a scalable and elastic environment, but geographically far away from the user. However, as previously mentioned, the prediction of the cloud network traffic requires having access to detailed information about the operation of the network (*e.g.* timestamp, protocols, etc.).

To address this issue and facilitate the prediction process, the *Cloud Network Traffic Monitor* must constantly monitor the cloud infrastructure (or a specific application). Thus, it will be able to capture the usage patterns and network traffic trends during a given time period. Basically, this component is responsible for counting all the traffic it receives from the network rather than counting only the frames that the controller is supposed to receive.

The cloud environment offers different levels of services to the clients. The lower level resources are usually restricted and hidden from the users (at the PaaS and SaaS level, for instance). Hence, the user does not have permission to monitor and control the whole network infrastructure. However, the cloud providers are able to monitor the network at the resource and virtualization layer since they are responsible for the low-level monitoring [Weingartner et al., 2015].

It is not within the scope of this study to propose a particular approach to monitor the cloud infrastructure. However, several tools have the potential to monitor the cloud environment with the aid of distributed agents in virtual machines, such as Nagios, OpenNebula, and Nimbus [Aceto et al., 2013].

3.3.2 The Time Series Builder

As illustrated in Figure 3.4, once the data has been collected, it is sent to the next component. The *Time Series Builder* handles the data by measuring the number of packets in the network traffic at regularly spaced intervals (*i.e.* the time between the observations must be constant), and thus forms a discrete time series ordered by the time.

At the same time, the *Time Series Builder* has to filter all the data in a search for similar protocols and the excess data that does not match the requirements of the filtering will be dropped. When building a time series, the input value is computed for each single variable. As a result of this process, the time series will be ready for analysis by any of the prediction models.

3.3.3 The Predictor Selector

Traffic predictors usually operate over all of the previous data or resort to windows of a finite but fixed size [Chen et al., 2012]. However, the network traffic in the cloud computing environment may undergo sudden changes due to the large number of requests and dynamic demands which are made without any prior notification [Ballani et al., 2011].

This led us to consider adopting the sliding window approach as a "forgetting" process, which makes it possible to restrict the amount of data to be processed. If the sliding window is small (which is normal for local analysis approaches), the model will be more sensitive to changes. In addition, it will generate low workload due to the reduced number of data packets that need processing. This occurs when the data flow has a stable behaviour. If the sliding window is large (such as the ARMA and ARIMA models), the predictor will hide any traffic anomalies. This situation arises when the time series is rapidly increasing (or reducing) the data flow.

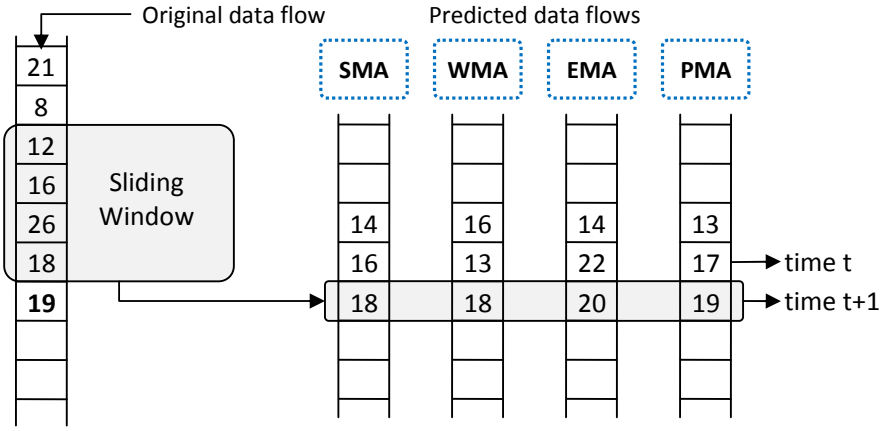


Figure 3.5: Sample of prediction inside the sliding window

Different approaches are required for local and global analysis so that these changes in traffic behaviour can be taken into account. When predicting net-

work traffic data based on a global analysis, the ARMA and ARIMA models are used. In the case of ARMA and ARIMA modelling, we use the statistical environment *R* [Team, 2012]. When estimating the network traffic by means of the ARMA model, the *Analysis Mechanism* selects the "FitARMA" package [McLeod and Zhang, 2008]. When the estimate was performed using the ARIMA model, the *Predictor Selector* used the "forecast" package [Hyndman and Khandakar, 2008] to fit the time series. For this, both packages use a function for returning automatically the best set of parameters (GetFitARMA and Auto.Arima, respectively) according to the algorithms presented in [McLeod and Zhang, 2008] and [Hyndman and Khandakar, 2008]. These functions conduct a search over possible model within the order constraints provided.

The variance between the previous and current sliding window was taken into account for the local analysis. The example illustrated in Figure 3.5 shows a sliding window with size four. Each value of the original data flow is weighted with a portion of the statistical distribution of the corresponding predictor model (SMA, WMA, EMA, and PMA). Thus, at time t, the sliding window has a set of four values $\{12, 16, 26, 18\}$. In the next turn, at time $t + 1$, the next value to enter inside the window will be 19, and when this occurs, the oldest value (12) leaves the sliding window. This process will be repeated as long as there is a data flow from the network.

At this point, the *Operator* might interact with the *Predictor Selector* component so that it can choose the predictor model and define some parameters for the static approach (*e.g.* arithmetic mean, variance of the data, and time period). Thereafter, the prediction will perform two types of approaches for each time period inside a sliding window for all the local analysis models, namely, static and dynamic.

Algorithm 3.2. Pseudocode for Predicting Network Traffic

Input: Time series trace
Output: Prediction of network traffic

```
1:  Start
2:      read tmTrace
3:      read parameters
4:      vector vPModel
5:      vector vPrediction
6:      for (i = 0; i < tmTrace.size(); i++) do
7:          var nextValue ← 0
8:          for (j = 0; j <= window.size(); j++) do
9:              if (i − j >= 0) then
10:                 var tmp ← (vPModel[j] ∗ tmTrace[i − j])
11:             end if
12:             nextValue ← (nextValue + tmp)
13:         end for
14:         vPrediction.add(nextValue)
15:     end for
16: End
```

3.3.3.1 Static Approach

Algorithm 3.2 shows the procedure for predicting traffic. The input (*i.e.* the data flow in Figure 3.5) corresponds to the time series traces (*tmTrace*). It should be remembered that the input data can be any set of cloud data. Furthermore, the *Operator* also has to set up the statistical parameters such as arithmetic mean, variance, standard deviation, number of slices, vector model, as illustrated at lines 3 and 4 of Algorithm 3.2 (for the static approach). When the algorithm works dynamically, these parameters are computed automatically inside the last sliding window. The number of slices is equivalent to the number of samples of data used for the calculation of the prediction coefficients (from the moving average models). Once the vector with slices has been properly shaped, the algorithm estimates the next value for the network traffic for each new value from the time series (line 12, *nextValue*). As output, the vector containing the network traffic prediction, is represented by *vPrediction*.

3.3.3.2 Dynamic Approach

The *Dynamic Approach* component is responsible for defining the window size that serves as input for the next sliding window. To reduce the complexity of predicting network traffic, time-bounded past information is considered by means of a sliding window of a size defined by the Dynamic Window Size Algorithm (Algorithm 3.3), which thus makes it suitable for online prediction in a cloud computing context [Dalmazo et al., 2016a].

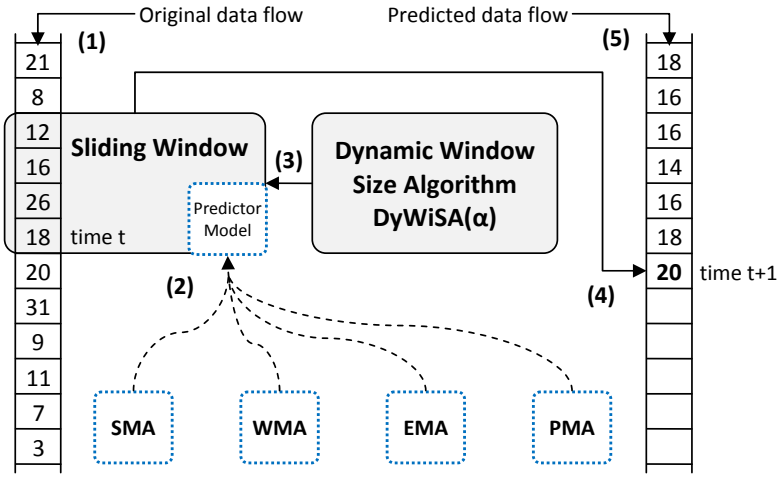


Figure 3.6: Elements of the Dynamic Window Size approach and iterations

Figure 3.6 illustrates the main conceptual components and their interactions. Real-time cloud traffic data (step 1) is gathered and analysed in the *Sliding Window* component in order to estimate network traffic from short historical data. This cloud data traffic is processed according to a particular predictor model, as illustrated in step 2. Possible candidates for the predictor model (described in Section 3.2.4) include Simple Moving Average, Weighted Moving

Average, Exponential Moving Average and Poisson Moving Average. The *Dynamic Window Size Algorithm* component is responsible for the definition of the window size that serves as input to the *Sliding Window* component (step 3). The next value for cloud data traffic is predicted (step 4) according to the chosen predictor model, therefore resulting in a sequence of predicted values for the cloud data traffic (step 5).

By employing a window of dynamic but limited size, we minimize the workload by reducing the amount of data that must be processed by the predictor model. We now describe each component in more detail.

Sliding Window

In order to reduce the complexity of predicting network traffic, we propose time-bounded past information by means of a sliding window of size defined by the *Dynamic Window Size Algorithm* (Algorithm 3.3 – DyWiSA). A window of the given size is used to weight past observations of data traffic according to the distribution employed by the predictor model.

The example illustrated in Figure 3.6 shows a sliding window with size four. Each value of the original data flow is weighted with a portion of the statistical distribution of the corresponding predictor model [Dalmazo et al., 2013]. For instance, the *DyWiSA* is familiar with the statistical behaviour of the predictor models. In this specific case, for each time slot, the *Sliding Window* considers the fourth part of the distribution to ponder the number of network packets.

It is worth pointing out that the Simple Moving Average, Weighted Moving Average and the Poisson Moving Average use a discrete function to weight the data. However, for the Exponential Moving Average, the DyWiSA divides the function into a finite number of discrete elements before using, namely, it discretizes of the exponential function.

Thus, at time t, the sliding window has a set of four values $\{12, 16, 26, 18\}$. In the next turn, at time $t + 1$, the next value to enter inside the window will be 20, and when this occurs, the oldest value (12) leaves the sliding window. This process will be repeated as long as there is a data flow from the network.

Dynamic Window Size Algorithm - DyWiSA

In order to fill the gap from the static model aforementioned, this section proposes a new Dynamic Window Size Algorithm. Traffic predictors usually operate over all of previous data [Chen et al., 2012] or resort to windows of finite but fixed size. However, the network traffic in the cloud computing environment may suffer sudden changes due to the large amount of requests and dynamic demands without prior notification [Ballani et al., 2011].

Considering the open issue aforementioned this section proposes and describes a sliding window approach as a forgetting process that limits the amount of data to be processed. If the sliding window is large, the predictor will be able to smooth

Algorithm 3.3. Dynamic Window Size

Input: Average of the current sliding window, $newAvg$
Average of the previous sliding window, $oldAvg$
Current sliding window, $sWindow$
Output: Next window size, $wSize$

```
1:  Start
2:      procedure DyWiSA(newAvg, oldAvg, sWindow)
3:          var wSize ← sWindow.size()
4:          var direct ← newAvg/oldAvg
5:          var inverse ← oldAvg/newAvg
6:          var ratio ← | direct − inverse |
7:          if (ratio > (1 + α)) then
8:              var volume = σ²max / σ²
9:              if (newAvg > oldAvg) then
10:                 wSize ← wSize + volume
11:             else
12:                 wSize ← wSize − volume
13:             end if
14:         end if
15:         return wSize
16:     end procedure
17: End
```

traffic anomalies. This situation happens when the time series are increasing (or decreasing) the data flow quickly. If the sliding window is small, the model will be more sensitive to changes, however it will generate lower workload due to the fewer number of data packets to process. This happens when the data flow presents a stable behaviour.

To take these traffic behaviour changes into account, we consider the variance (σ^2) between the previous and current values inside of the sliding window. Algorithm 3.3 describes the operation of the Dynamic Window Size Algorithm. It resorts to a sliding window of variable size, with size changes happening only when the difference between the average of current and previous window exceeds a threshold α.

The algorithm receives as input the average of the current sliding window, the average from the previous sliding window and the current sliding window. It compares the average of the old sliding window with the average of the current sliding window. In order to avoid unnecessary algorithm overhead, we consider a threshold α for changes to the sliding window. This threshold corresponds to a boundary value for the population parameter for which the difference between the current value and the mean of the last window is not statistically significant at the α level. The α estimation and evaluation are reported in Section 3.4.

Let *ratio* be a value which measures average changes between the current window and last window. If the difference between *newAvg* and *oldAvg* is higher than the threshold $(1 + \alpha)$, *i.e.* statistically significant, the window size is

increased (or decreased) by *volume*. In order to quantify the maximum variance of a sliding window and, consequently, know the variation of the window size, a measurement to express the largest variance possible inside of a subset of the entire population is needed. We consider the theoretical maximum variance (σ^2_{max}) to be the variance of the extreme values of a sliding window. For this, we use the ratio between the σ^2_{max} and the σ^2 inside a sliding window. This whole process is represented by the variable *volume* at line 6 of Algorithm 3.3.

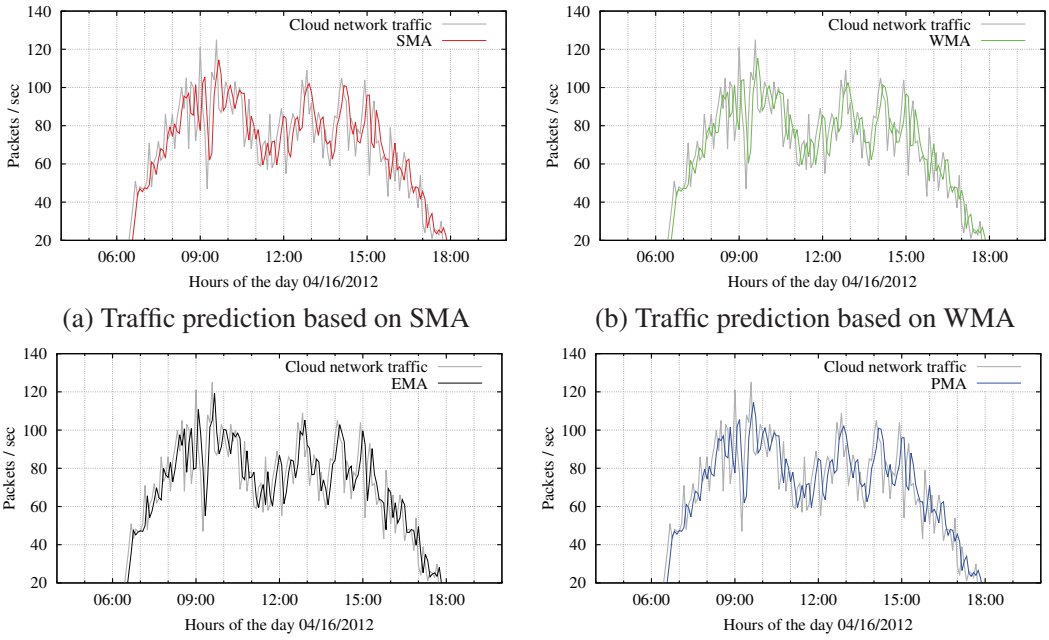
Proposition 1. The theoretical maximum variance of a given set of data can be estimated from the product of the difference of its extreme values, y_a (lowest value), y_b (highest value), and the average, as follows:

$$\sigma^2_{max} = (m - y_a)(y_b - m) \tag{3.9}$$

Proof. See Appendix A. ■

Figure 3.7 illustrates the performance of different approaches (using DyWiSA) with the traffic data set containing information from Dropbox monitoring (see Subsection 3.4.1). In order to provide a better viewing of the results, we only show forecasts for a limited period. However, the observable match between real values and predictions held for remaining time periods.

Finally, the algorithm returns the window size to be used by the *Sliding Window*



(a) Traffic prediction based on SMA

(b) Traffic prediction based on WMA

(c) Traffic prediction based on EMA

(d) Traffic prediction based on PMA

Figure 3.7: Sample of cloud network traffic prediction

component. This *Dynamic Window Size Algorithm* is at the core of online traffic prediction by dynamically adapting the window size resorting only to local data from current and previous sliding windows, instead of global traffic data.

α Parameter Estimation

For this thesis, we propose an approach to estimate the best α for each forecast. The goal of this methodology is twofold: to select the α parameter that provides the highest possible accuracy for predicting network traffic; and to minimize the computational costs as much as possible. We consider the α parameter a control mechanism which represents the minimum variation of data traffic needed for the window size to change. A small α will lead to frequent window size changes and higher computationally complexity, while a higher α leads to less frequent changes with corresponding lower computational costs. For instance, when the α is equal to 0.1 it means that the average of the current sliding window must be, at least, 10% greater or smaller than the last sliding window for the algorithm to require a new window size calculation.

Seeking a methodology that determines a value of α that provides good accuracy results without compromising the need for online traffic prediction (*i.e.* little dependence on historical data), we consider only an initial set of windows to determine an optimal α value. As observed in other works [Kwon et al., 2011, Loiseau et al., 2010], the resemblance between the first two sliding windows and the entire dataset suggests that the network traffic data exhibits the property of self-similarity. The algorithm will set the best α to predict the entire dataset taking advantage of this concept. Section 3.4.3 presents the evaluation of this process.

3.3.4 Report

For evaluating network traffic prediction techniques many different metrics are used to measure the quality of the forecasting [Joshi and Hadi, 2015]. The analysis mechanism provides as result a detailed report about the prediction models. For the static and the dynamic approaches, several statistical descriptors are calculated and arranged in a table such as arithmetic mean, mean square error, standard deviation, standard error, variance, etc.

The effectiveness of the prediction is measured through the Normalized Mean Square Error (NMSE) [Weigend and Gershenfeld, 1994] and Mean Absolute Percent Error (MAPE) [Makridakis et al., 2008]. NMSE is defined as:

$$NMSE = \frac{1}{\sigma^2}\frac{1}{N}\sum_{t=1}^{N}\left(X_t - \hat{X}_t\right)^2 \tag{3.10}$$

where σ^2 is the variance of the time series over the prediction duration, X_t is the observed value of the time series at time t, $\hat{X}_t$ is the predicted value expected

from X_t, and N is the total number of predicted values. This metric is widely utilized to assess prediction accuracy. Its results are compared with a trivial predictor, which statistically predicts the mean of the actual time series, in which case NMSE = 1. If NMSE = 0, this means that it is a perfect predictor, whereas NMSE > 1 means that the predictor performance is worse than that of a trivial predictor [Weigend and Gershenfeld, 1994].

MAPE measures expressed errors as a percentage of the actual data over the prediction data. It is calculated as the average of the unsigned percentage error, and is defined by the formula:

$$MAPE = \begin{cases} \left(\frac{1}{N}\sum_{t=1}^{N}\frac{|X_t-\hat{X}_t|}{|X_t|}\right) * 100 & if(X_t > 0) \\ \left(\frac{1}{N}\sum_{t=1}^{N}\frac{|X_t-\hat{X}_t|}{|\overline{X}|}\right) * 100 & otherwise \end{cases} \tag{3.11}$$

where, X_t is the observed value, $\hat{X}_t$ is the predicted value and N represents the total number of values in the time series as well as referenced in NMSE. If the denominator is zero then the actual value X_t is replaced by the average of time series, $\overline{X}$. When having a perfect fit, MAPE is zero.

3.4 Evaluation and Discussion

Throughout this section, the time series setup that is used to assess this work, is provided. Furthermore, we evaluate the performance of the static and dynamic sliding window mechanisms (the best parameter α) employed for the local analysis of traffic prediction. In addition, the results are compared with all the predictors outlined in Section 3.2.4. We consider two case studies for evaluation: Dropbox datasets and Data Centre dataset.

3.4.1 Dropbox Dataset

Two datasets from Dropbox monitoring were used for this case study, they are: Home 1 and Campus 2, as described in the [Drago et al., 2012]. Home 1 dataset consists of ADSL and Fiber to the Home customers of a nation-wide Internet Service Provider, but they might be able to use WiFi routers at home to share the connection. Campus 2 was collected in academic environments instead, such as wired workstations in research and administrative offices as well as campus-wide wireless access points.

Setup

The evaluated time series data captures the usage of Dropbox, the most widely used cloud storage system [Drago et al., 2012]. All the measurements and data

provided in this subsection were collected from March 24, 2012 to May 5, 2012. The original Dropbox dataset encompasses more than 100 metrics about network traffic. However, for the purposes of this study, the *Time Series Builder* (Subsection 3.3.2) considers the total number of packets observed from the client (server) to the server (client) and SSL/TLS protocol.

3.4.2 Data Centre Dataset

At the same time another dataset was used, for a better characterization of the cloud computing environment, and which provides data from monitoring a variety of services that are common in cloud computing [Benson et al., 2010]. In that work, the authors describe several services that can be found in the dataset such as webmail servers, web portals, instant messaging, web services and multicast video streams.

The authors use SNMP link statistics to examine the network-level impact in terms of link utilization, congestion, packet drops, and the dependence of these properties on the location of the links in the network topology and on a daily basis. In addition, the dataset has data from a two-layer topology that introduce server virtualization techniques in order to reduce heating and electric power consumption.

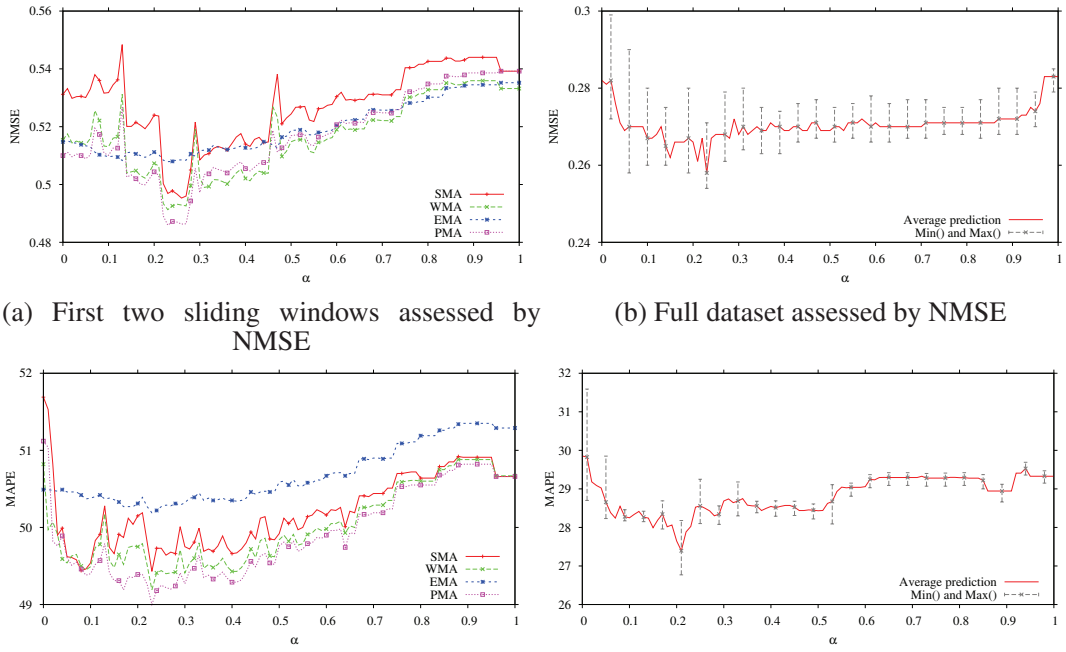
Setup

The data was collected in an academic environment and the dataset consists of more than five years of monitoring. However, the authors only provide a fraction of the total amount of data used in the original paper [Benson et al., 2010], for around 10 days. The granularity of the data generated is sixty seconds for each time slot and the resulting time series is called by *Data Centre* throughout this study. The *Data Centre* stores data from around 1000 servers located in the West and Mid-West of the U.S. The *Analysis Mechanism* was performed for the data in a similar way to what is described in the Dropbox case study.

3.4.3 α Parameter Evaluation

In this section we analyse the impact of α on the prediction accuracy and present a methodology for selecting its value for two scenarios: *Dropbox* and *Data Center* datasets. Figure 3.8(a) depicts accuracy results (NMSE) of the two initial windows for a range of α values between 0.01 and 1.0. This shows that the optimal α for the two initial windows is 0.23 (smaller NMSE), which incidentally is also the optimal α value when considering the whole dataset, as shown in Figure 3.8(b). The same happens when considering the MAPE accuracy metric, as shown in Figure 3.8(c) and 3.8(d), respectively for the two initial windows and the overall dataset.

Figure 3.9 presents similar results for the Data Center dataset. For this case, this methodology leads to an optimal α value of 0.15 for both metrics as well. While



(a) First two sliding windows assessed by NMSE

(b) Full dataset assessed by NMSE

(c) First two sliding windows assessed by MAPE

(d) Full dataset assessed by MAPE

Figure 3.8: Evaluation of α parameter for the Dropbox dataset

this may not provide an overall optimal value, it is figured out that selecting the best α parameter from the two initial windows provides, in general, a good approximation.

Table 3.3 shows the output results when we run the algorithm with different α values in a sample time series. With an α of 0.23, the Dynamic Windows Size Algorithm changes the windows size 886 times (just 7.13% of the total) with no significant changes in the overall average and standard deviation of the data predicted. As α becomes smaller, the overhead becomes higher and vice-versa.

From a computational point of view, the α estimation yields negligible complexity because this process requires only an initial set of two sliding windows. In comparison with the first version of the Dynamic Window Size Algorithm (with fixed α), the prediction time has been decreased by almost half after the α optimization approach. For instance, in the previous version of the DyWiSA, the Poisson Moving Average (the predictor model with the best prediction accuracy) spends 0.1570 seconds to compute the forecast for a data set with more than 12000 values, namely, Home 1 data set from Dropbox (see Subsection 3.4.1). After the α optimization, the DyWiSA computes the prediction in 0.0830 seconds. Details about the time consumption improvement, NMSE and MAPE for the other models are presented in Table 3.4.

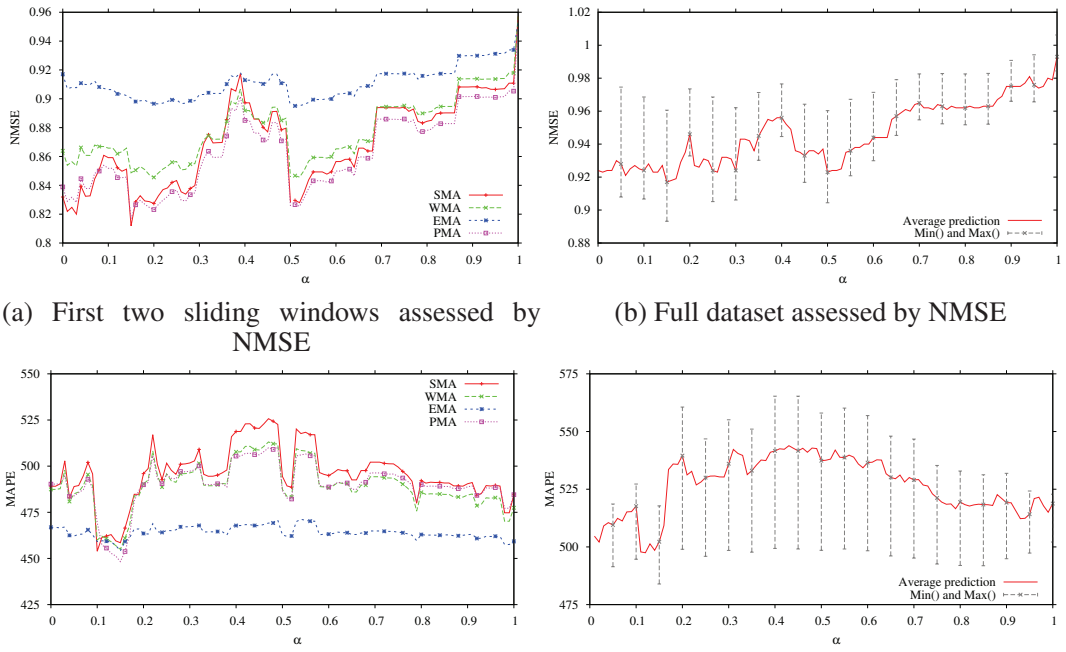
It is evident that α has an important impact on the prediction quality, while also affecting the computational requirements of the Dynamic Window Size Algorithm. To warrant a fair comparison between all datasets evaluated in this

Table 3.3: Sample of α evaluation

α ***value***	***Model***	***Average***	***Std. Deviation***	***Changes***
0.30	*SMA*	29.79	16.47	759
	WMA	29.80	16.49	759
	EMA	29.81	17.11	759
	PMA	29.80	16.64	759
0.20	*SMA*	29.79	16.25	916
	WMA	29.80	16.28	916
	EMA	29.81	17.09	916
	PMA	29.81	16.51	916
0.10	*SMA*	29.70	16.24	2093
	WMA	29.74	16.27	2093
	EMA	29.81	17.09	2093
	PMA	29.79	16.51	2093
0.02	*SMA*	29.18	15.92	8070
	WMA	29.42	16.05	8070
	EMA	29.79	17.12	8070
	PMA	29.70	16.48	8070

Table 3.4: Time consumption (seconds) and improvement

Model	α***-Fixed (s)***			α***-Optimized (s)***			***Time Improvement***
	Time	NMSE	MAPE	Time	NMSE	MAPE	
SMA	0.0287	0.4045	41.58	0.0169	0.2709	28.18	41.11%
WMA	0.1024	0.3422	36.50	0.0707	0.2590	27.47	30.96%
EMA	0.1174	0.2807	29.36	0.0768	0.2600	27.20	34.58%
PMA	0.1570	0.2720	28.90	0.0830	0.2543	26.77	47.13%



(a) First two sliding windows assessed by NMSE

(b) Full dataset assessed by NMSE

(c) First two sliding windows assessed by MAPE

(d) Full dataset assessed by MAPE

Figure 3.9: Evaluation of α parameter for the Data Center dataset

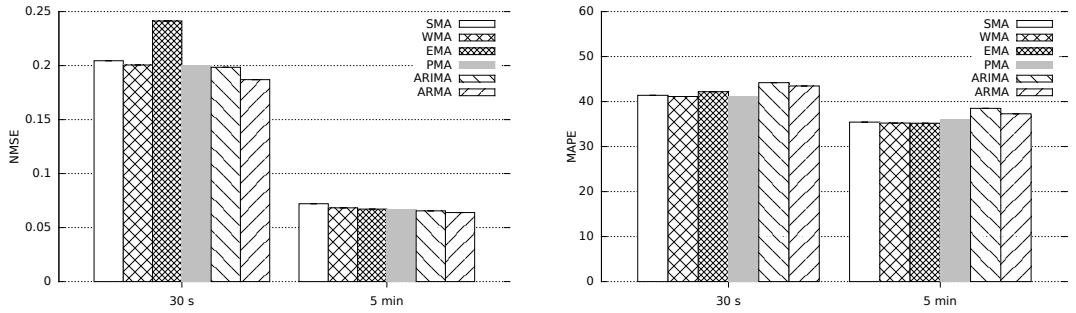
work, we consider the same methodology to determine the best α from the initial sliding windows, as described above.

3.4.3.1 Dropbox Case Study

Figure 3.10 illustrates the evaluation for two different windows size. For this, the time series was divided into intervals of thirty seconds and five minutes. In this scenario, we use the dynamic approach for evaluating the Campus 2 dataset, and the analysis mechanism was performed by applying a sliding window weighted with the four statistical models, ARMA and ARIMA models described in Section 3.2.4. The results show that the best level of accuracy can be achieved with 5 minutes interval because larger intervals present lower variance among the data. Other datasets evaluated in this work have presented similar behaviour when performed with thirty seconds and five minutes. For the remainder of the evaluation, we use 5 minutes interval for evaluating the predictions.

3.4.3.2 Results and Discussion

First, Table 3.5 shows the results of the static approach in which the sliding window size remains constant during the prediction data. After that, we show the performance of a dynamic approach which calculates the sliding window size by means of Algorithm 3.3 in Table 3.6. Finally, we compare the results with the forecast data that were obtained from the ARMA and ARIMA predictors.



(a) NMSE (b) MAPE

Figure 3.10: Evaluation for different sliding window sizes (Campus2)

Static Approach

The evaluation of the network traffic prediction based on sliding window with static size is determined by metrics of the overall historical data, such as the global average. Then, the *Static Approach* demands an *a priori* analysis of the dataset. After this process, the analysis of the data generates several statistical descriptors that are employed as the size parameter of the static sliding windows. Table 3.5 provides results for three different values of the sliding window: arithmetic mean, standard deviation and variance. The first line shows statistics of the Dropbox dataset that was used as input for the predictors, while the second line exhibits results achieved for a trivial predictor that always predicts the next value as the arithmetic mean of data. The following lines show results for Poisson prediction model as well as three others (EMA, WMA and SMA) presented in Section 3.2.4.

While for most metrics the results of the remaining predictors are not far from those obtained by the Poisson approach, it is worth noting that the results achieved by this approach have the best performance concerning NMSE and MAPE, where PMA excels when compared to the others. This means that the difference between the estimated values and the real values is the lowest in the evaluation's result.

Table 3.5: Descriptive statistics for Campus 2 dataset

Dataset		Mean			Std. Dev.	Variance	NMSE	MAPE
		Arithmetic	Square Error	Std. Error				
1	Dropbox	45.994	0.000	0.486	54.123	2929.4	0.000	0.000
2	Trivial	45.994	2929.4	0.000	0.000	0.000	1.000	337.98
Approach		**Sliding window size arithmetic mean**						
3	PMA	45.990	143.076	0.482	53.634	2876.71	0.0494	37.93
4	EMA	45.989	185.282	0.476	52.988	2807.81	0.0887	39.04
5	WMA	45.995	395.621	0.466	51.838	2687.19	0.1845	54.61
6	SMA	45.993	581.035	0.463	51.546	2657.02	0.2856	66.68
Approach		**Sliding window size standard deviation**						
7	PMA	45.988	208.785	0.474	52.834	2791.46	0.0745	38.36
8	EMA	45.990	278.076	0.468	52.045	2708.71	0.1026	41.93
9	WMA	45.999	594.747	0.455	50.679	2568.38	0.2313	57.43
10	SMA	45.993	931.031	0.449	50.063	2506.40	0.3714	71.31
Approach		**Sliding window size variance**						
11	PMA	45.988	268.790	0.471	52.468	2752.61	0.0974	41.43
12	EMA	45.990	381.512	0.457	50.829	2583.66	0.1475	46.19
13	WMA	46.003	1175.951	0.420	46.837	2193.73	0.5356	94.46
14	SMA	45.984	1908.381	0.405	45.121	2035.95	0.9372	134.69

Table 3.6: Dynamic sliding window size

Dataset		Mean			Std. Dev.	Variance	NMSE	MAPE
		Arithmetic	Square Error	Std. Error				
1	Home1	29.81	0	0.161	17.96	322.42	0	0
Approach								
2	SMA	29.802	93.58	0.146	16.31	265.94	0.2709	28.18
3	WMA	29.829	87.13	0.146	16.35	287.31	0.2590	27.47
4	EMA	29.818	84.62	0.153	17.13	273.38	0.2600	27.20
5	PMA	29.827	82.68	0.148	16.58	274.87	0.2543	26.77

As illustrated in Table 3.5, for the sliding window with size arithmetic mean, we note that the Mean Square Error ranges between 143.08 to 581.04 (the lowest error range), while for sliding window size with variance size, goes from 268.79 to 1908.38 (the highest error range). Thus, sliding window size arithmetic mean provides results more stable than other approaches. As we can observe, NMSE and MAPE values are the lowest for the arithmetic mean size (0.0494; 0.0887; 0.1845; 0.2856), indicating higher levels of accuracy.

We also use this methodology to estimate the best case scenario for the Home 1 in the static approach. It generates a table similar to Table 3.5 (omitted to avoid redundancy of information). Among different sliding window size assessed, the best result was achieved by the sliding window with arithmetic mean size, as illustrated in Table 3.5.

While for most metrics the results of the remaining predictors are not far from those obtained by the Poisson approach, we highlight the results achieved in terms of NMSE and MAPE, where PMA excels when compared with the others. This means that the difference between the estimated values and the real values is the lowest in the overall result of the evaluation. In addition, Table 3.7 attests that the Poisson predictor has the strongest correlation among the predictors assessed in this evaluation.

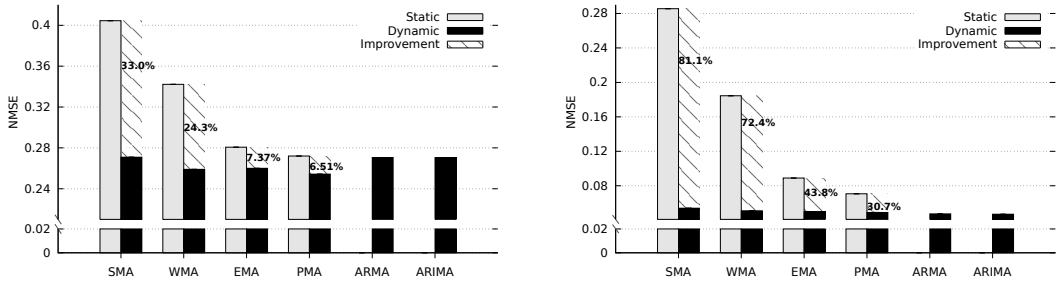
Table 3.7: Pearson correlation

	Dropbox	SMA	WMA	EMA	PMA
Dropbox	1,00	0,729	0,903	0,909	0,924

Dynamic Approach

For the assessment of the dynamic approach, the Algorithm 3.3 was used to calculate the sliding window size. Furthermore, in the dynamic approach, the predictor models were evaluated from the two Dropbox traffic traces (Home 1 and Campus 2). Figure 3.11 illustrates the NMSE accuracy of the predictor models. All the local analysis predictor models were tested in their original version with a static window size, as well as by employing the dynamic window size methodology. Although our focus is on making a comparison between predictor models operating with a static window size and a dynamic window size, it was clear that the SMA consistently provides the worst results, irrespective of what window size methodology was used. At the other extreme, there is PMA, which provides the best overall results.

With regard to the comparison between the static and dynamic approaches, our results show that all the predictor models achieve better results when employing the dynamic window size methodology. There is further evidence of this in the NMSE results of Figure 3.11, which show that all the predictors are improved from as little as 6.51% for the best predictor model (PMA in Home1) to as much as 81.1% for the worst predictor identified (SMA in Campus2). ARMA



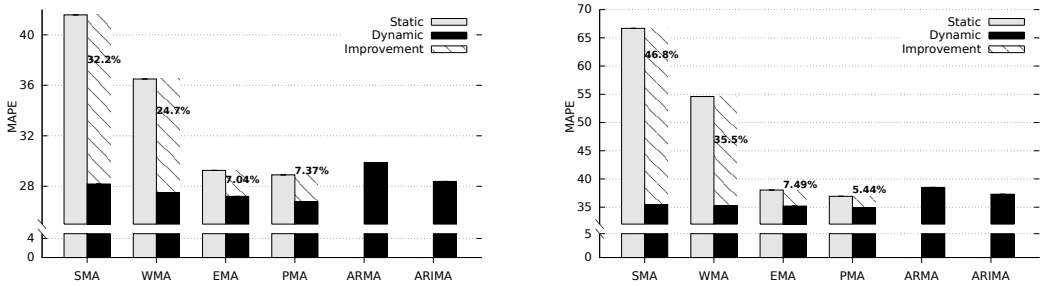
(a) Home1 (b) Campus2

Figure 3.11: The NMSE results from the Dropbox datasets

and ARIMA models provide results that are slightly better for the Campus2 dataset.

It is worth noticing that in Figure 3.11 (a), WMA achieves a better result than EMA and this is not confirmed in Figure 3.12 (a). When the predictor model is assessed by NMSE, the data normalization process tends to improve the results of the predictor with the highest variance (see Equation 3.10). In this case, the WMA obtains better results than EMA because its predicted data shows the higher σ^2 (see Table 3.6). In order to avoid the problem of a wide variance of data, the *Dynamic Window Size Algorithm* was also evaluated by MAPE.

Figure 3.12 shows the performance of the predictor models in terms of error percentage. This illustrates the fact that in both cases (Home 1 and Campus 2) the error rate decreases when the dynamic window size methodology is employed. The overall MAPE results can be seen in Figure 3.12, which shows that the dynamic methodology improves the prediction results for all the local analysis predictors, from 5.44% (PMA) to 46.8% (SMA). While the ARMA and ARIMA models have similar results for Home1 and Campus2 in terms of MAPE, the best being the PMA.



(a) Home1 (b) Campus2

Figure 3.12: The MAPE results from the Dropbox datasets

Running Time

In this subsection, we present the time cost of running the prediction models presented in Section 3.2.4. In addition, we compare the performance between the first version of the static window size approach and the Dynamic Window Size Algorithm. Table 3.8 presents details about the processing time of the prediction models with different size of input data. It is possible to observe that the SMA, WMA, EMA and PMA models spend a similar time to compute the prediction. Analogously, the ARMA and ARIMA models present a similar time for performing predictions. These results are in accordance with the computational complexity discussed in the Subsection 3.2.6.

It is worth noticing that other datasets evaluated in this work have presented a similar processing time to perform the prediction. An entire view of the results is displayed in Figure 3.13 which shows the *Static Approach* as the average time of the SMA, WMA, EMA and PMA models with static window size. The *Dynamic Approach* represents the average time of the SMA, WMA, EMA and PMA models using the Dynamic Window Size Algorithm. It is clear that the *Dynamic Approach* presents the best time performance due the lower number of operations [Dalmazo et al., 2014]. Moreover, the prediction time has been increased exponentially for ARMA and ARIMA models.

Naturally, the minimum real-time measurement that can be achieved depends on the hardware configuration (*i.e.*, processor clock rate, memory available, etc.). For comparative purposes, all the tests performed in this work were based on a standard personal computer with a DualCore Intel Core 2 Duo CPU 6300 1.86GHz and 3Gb DDR2-SDRAM.

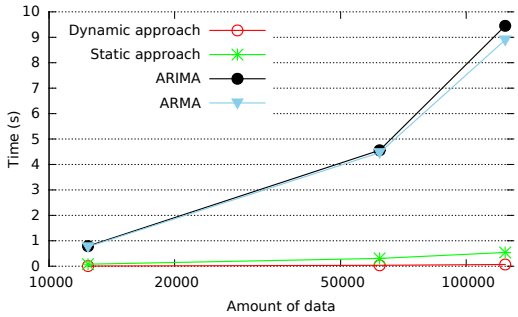
Table 3.8: Time consumption for Home1

Model	***123800 elements***		***61900 elements***		***12380 elements***	
	Static	Dynamic	Static	Dynamic	Static	Dynamic
SMA	0.069	0.238	0.034	0.130	0.006	0.026
WMA	0.076	0.614	0.038	0.345	0.006	0.085
EMA	0.076	0.645	0.038	0.363	0.006	0.092
PMA	0.076	0.670	0.038	0.391	0.006	0.112
ARIMA	-	9.456	-	4.558	-	0.791
ARMA	-	8.897	-	4.464	-	0.766

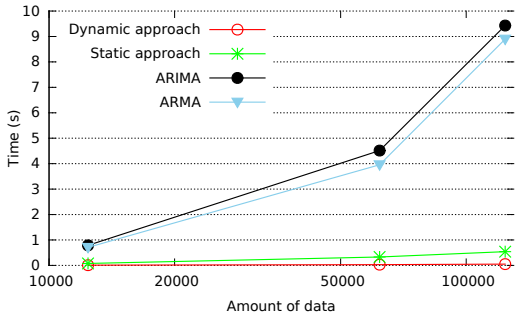
Results in seconds.

3.4.3.3 Data Centre Case Study

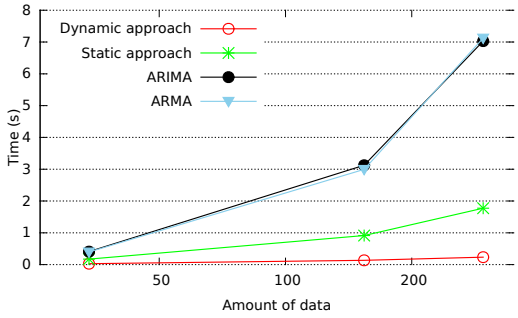
We divided the evaluation report into two parts. First, all the local analysis approaches were assessed using the *Data Centre* dataset as input. In this case, the *Analysis Mechanism* is employed for the data by adopting both a static and dynamic approaches. In the static approach, the window size is invariable during the prediction data. In the case of the dynamic approach, Algorithm 3.3



(a) Home1



(b) Campus2



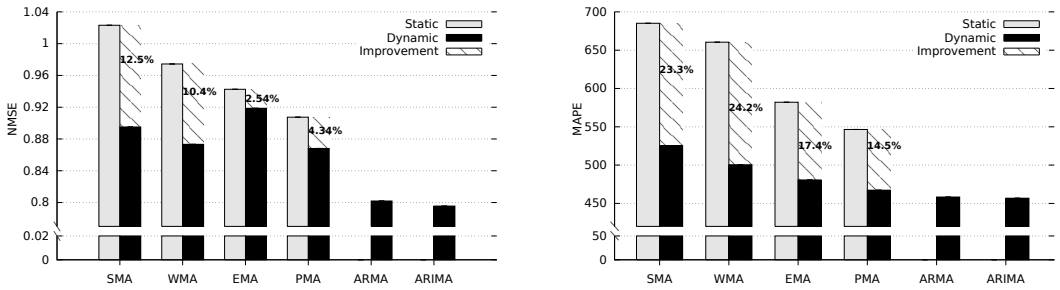
(c) Data Centre

Figure 3.13: Time consumption comparison for all data sets

calculates the sliding window size. After that, we carried out the prediction of the data with ARMA and ARIMA predictors.

Static Approach

In Figure 3.14, the prediction results are given by drawing on data from the *Data Centre*. In Figure 3.14 (a), we can see the bars in gray that represent the NMSE achieved for the local analysis predictors (SMA, WMA, EMA and PMA). Analogously to the Dropbox case study, PMA yields the best result which is close to 0.91 for NMSE. Figure 3.14 (b) shows the performance for MAPE. PMA, like in the NMSE evaluation, achieves a better result than the other local analysis predictors assessed in this work.



(a) NMSE (b) MAPE

Figure 3.14: NMSE and MAPE results for Data Centre

Dynamic Approach

The *Data Centre* dataset was also evaluated by Algorithm 3.3 so that it could generate results from a dynamic window size perspective. Figure 3.14 illustrates the accuracy of the prediction models.

It is clear that all the predictors significantly improve their results after adopting a dynamic window size approach to forecast the time series. This was confirmed in the NMSE and MAPE results of Figure 3.14. The EMA predictor showed the smallest improvement with 2.54% and the SMA showed the best improvement with 12.5% regarding NMSE. In the case of MAPE, Figure 3.14 (b) illustrates an improvement from 14.5% (PMA) to 24.2% for WMA. Figure 3.14 shows that the ARIMA model provides better results than all the others predictors to forecast the data from the *Data Centre* with NMSE = 0.796 and MAPE = 456.8%. Finally, regarding NMSE, the ARMA model achieves a high degree of accuracy when predicting the network traffic, but is slightly outperformed by the ARIMA model, which achieves the best prediction result. In the case of the MAPE evaluation, the results are not different and the ARIMA model provides the most accurate prediction in the context of the cloud *Data Centre*.

The results are based on a comparison between several predictors. In summary, the moving average approach represents a solution that computes a local average of data at the end of the time window, based on the assumption that this is the best estimate to represent the current mean value around which the data are ranging. These approaches are suitable if the time series is subject to sudden changes, as cloud computing traffic is. In this case, an anomaly may be easily absorbed within the time window without compromising the prediction as a whole [Modi et al., 2013].

Approaches using global analysis (ARMA and ARIMA) achieve more accurate results when predicting network traffic from the cloud *Data Centre*. However, local analysis approaches outperform the results for cloud applications monitoring, as was seen in the Dropbox case study. Moreover, with a smaller sliding window, oldest values also have less influence on the predicted network traffic.

This indicates that a predictor that gives priority to recent history achieves better results for dynamic cloud computing environments.

On the one hand the global analysis achieves accurate results, while on the other, this kind of prediction is more expensive in computational terms than predictors based on local data analysis. In addition, a local data analysis is able to provide accurate predictions with relatively low levels of historical data dependency and computational complexity.

3.5 Summary of the Chapter

This chapter presented the state of the art and a taxonomy for network traffic prediction models, as well as an analysis mechanism that provides a standardized approach for evaluating network traffic predictors based on global and local data analysis. The outcomes of this mechanism enable the performance comparison of several predictors in the cloud, particularly in terms of accuracy, historical dependency, time and computational overhead.

From the observation of the results of the Dropbox case study, it can be seen that all the predictions based on local analysis present a considerable improvement after using the Dynamic Window Size Algorithm (DyWiSA). Apart from this, the DyWiSA facilitates online traffic prediction due to its short dependency on historical data. Compared to other predictors, Simple Moving Average performed significantly worse. Furthermore, besides the good results, the Poisson Moving Average has maintained the same computational complexity of the predictor models based on local analysis assessed in this work. Considering the *Data Centre* dataset with traffic from a diverse set of common cloud services, the ARIMA model shows a slight advantage over the other predictors in terms of accuracy. However, this is achieved at the cost of high computational complexity and time consumption. Poisson Moving Average, which is more attractive due to its lower computational complexity, has shown itself to be more suitable for dynamic cloud environments than the other predictor models assessed.

The outcomes of this chapter include the following publications:

- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, **"Performance Analysis of Network Traffic Predictors in the Cloud"**, Journal of Network and Systems Management, vol. 25, 2, pp. 290-320, Springer, 2017. Impact factor: 1.75
- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, **"Online traffic prediction in the cloud"**, International Journal of Network Management, vol. 26, 4, pp. 269-285, John Wiley & Sons, 2016. Impact factor: 1.34
- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, "**Online Traffic Prediction in the Cloud: A Dynamic Window Approach**", in The 2nd International Conference on Future Internet of Things and Cloud, 2014

- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, "**Predicting Traffic in the Cloud: A Statistical Approach**", in IEEE International Conference on Cloud and Green Computing, 2013

Chapter 4

Anomaly Detection for Cloud Network Traffic

Intelligence is the ability to adapt to change.

(Stephen Hawking)

Contents

The widespread adoption of cloud computing has been hampered by the lack of security mechanisms. In view of this, an approach for detecting anomalies in cloud network traffic is presented. The anomaly detection

mechanism works on the basis of a Support Vector Machine (SVM) and prediction of the network traffic. The key requirement for improving an anomaly detection mechanism, in the context of cloud, is to reduce the total amount of network data traffic that will feed the SVM.

4.1 Introduction

Cloud computing is a paradigm that involves service delivery models over the Internet, such as: infrastructure, software and platform. These services can be offered to a wide range of clients through the cloud providers. A recent report by Cisco Global Cloud Index predicts that by 2021, 94% of all workloads will shift from traditional data centers to cloud data centers [Networking, CISCO Global Cloud Index, 2018]. According to the same report, the lack of security mechanisms is the top factor that prevents the wide adoption of cloud service models. For example, 30% of reported breaches in 2016 involved attacks on cloud applications.

Online threats are constantly evolving in the virtual environment. In this context, cloud computing introduces significant new paths of attack. To give an example, Denial of Service (DoS) is a well-known type of attack that disrupts online operations. The attack is usually carried out by hundreds (or thousands) of requests for a service and has to be detected before it breaks down the server. Owing to the large number of simultaneous requests, this type of attack causes an anomalous behaviour in the network traffic. At the same time, the elastic and scalable nature of cloud environments means that they are also apt to undergo sudden changes [Ballani et al., 2011, Vieira et al., 2010], which makes it even harder to detect which parts of the incoming traffic are caused by vandalism or are being used legitimately.

IDSs are complex tools that include a number of concepts, definitions and techniques that may differ, depending on the situation. An IDS usually relies on two main approaches to detect intrusions that differ in the way the data is analysed and processed. The first approach corresponds to a search for evidence of an attack based on signatures of other similar attacks while the second approach consists of a search for deviations from the appropriate behaviour found in periodic observations of the system. The principal advantage of the signature-based detection method is that it leads to a low number of false alarms. However, signature-based IDSs are not able to detect new or variant forms of known attacks. One of the benefits of anomaly-based detection is that a new attack for which a signature does not exist can be detected if it occurs outside of the regular traffic patterns. In our study, we focus on the second class of IDS to detect threats to the network traffic in the cloud environment.

4.1.1 Open Issues for Anomaly Detection in the Cloud

Several techniques have already been proposed to perform anomaly detection in the cloud environment, such as fuzzy logic [Patel et al., 2013], entropy-based [Wang et al., 2010], artificial neural networks [Vieira et al., 2010] and decision tree classifier [Fu, 2011]. Also, different types of network traffic information are used to detect anomalies, such as the behaviour of protocols, CPU utilization and user logs. However, there is an apparent deficiency in their ability to detect anomalies from a large amount of data, as detailed in the related work (see Section 4.2).

In particular, these techniques require extensive tuning to improve their sensitivity and achieve satisfactory results. There is also no consensus about the best way to represent the huge volume of data generated by the cloud infrastructure. In this context, extracting a good set of features that represents the behaviour of the cloud network traffic remains an open issue. As a result, the literature lacks mechanisms that can enable it to improve the accuracy of anomaly detection for cloud environments while reducing false detection rates.

4.1.2 Contributions and Outline

To fill these gaps, an approach to detect anomalies in a cloud environment is presented. Our proposal relies on traffic prediction to obtain features that represent the expected appropriate behaviour of the cloud network traffic. This information is then used jointly with a Support Vector Machine (SVM) model that is supplied with these features. The combination of these two tools represents a novel and effective approach for detecting anomalous events in the cloud environment. The forecasting is conducted by a statistical method based on a Poisson process, that has proved to be suitable for dynamic environments such as cloud computing. SVM is already known as one of the best machine learning algorithms for binary classification [Deng et al., 2012]. Binary classification meets the objectives of this proposal, since our aim is to detect anomalies inside the normal network traffic.

The remainder of the chapter is organized as follows. Section 4.2 covers some of the most prominent related work. Section 4.3 describes the proposed solution and the methodology used for this chapter, whilst Section 4.4 presents the evaluation and discusses the results. Section 4.5 summarizes the chapter.

4.2 State of the Art

The current state of the art in IDSs contains many different techniques. However, due to the focus on evaluating a feature extraction approach, we restrict ourselves to present SVM models applied in the IDS context and several models for detecting anomalies in the cloud computing environment. In light of this, it is worth noticing that SVM was used as a tool for evaluating the quality of the

features that represent the cloud network traffic. Thus, it is possible to perform a fair comparison among other works in literature that use other approaches for extract features. In the end, a discussion about the state-of-the-art and the open issues is provided.

4.2.1 Support Vector Machine IDSs

Horng *et al.* proposed a Network Intrusion Detection System on the basis of Support Vector Machine with features selected by a hierarchical clustering algorithm [Horng et al., 2011]. The SVM uses features such as the type of protocol, the status of the connection, the number of file creation operations, length of the connection and the number of root accesses. In spite of the good results for attacks that generate anomalies, this approach does not present the same effectiveness for other attacks such as User-to-Root (U2R) and Remote-to-Local (R2L). The DARPA dataset was used to evaluate the proposed IDS.

Mulay *et al.* presented an IDS that combines Support Vector Machine and decision trees to build a multi-class SVM [Mulay et al., 2010]. This model can classify the network traffic in normal or abnormal. For the second case, the system also tries to identify which is the attack. The SVM model uses data from the DARPA dataset for the training phase. However, the work does not provide evaluation results to validate the performance of the detection system.

Shon and Moon presented a hybrid machine learning approach to detect anomalies in the network traffic [Shon and Moon, 2007]. This model is a blending between supervised and unsupervised SVM model. From this, they aim to increase the performance in detecting new attacks. Besides, they use a Genetic Algorithm for extracting more appropriate packet fields (protocol, IP, TTL). However, in the evaluation section, the proposal presents a high false positive rate with data from the DARPA dataset.

Li and Liu proposed a new module for the Intrusion Detection System Snort, that is a well-known IDS based on signature [Li and Liu, 2010]. Although their proposal does not provide evaluation results on the paper, the authors said that the SVM improves the response time for new attacks. Features, such as protocols and network packets, are extracted from Netfilter and Iptables.

Chen *et al.* performed a comparative study between Artificial Neural Network (ANN) and Support Vector Machine to predict attacks on the basis of frequency-based encoding techniques to select the features [Chen et al., 2005]. The aim of this proposal is to increase the generalization capability of detecting more attacks from less training data. The results have shown that both approaches are able to detect anomalies in the network traffic, but SVM outperforms ANN.

4.2.2 Anomaly Detection in Cloud

Kholidy and Baiardi proposed a framework for a cloud-based IDS with features from signature attacks and user logs [Kholidy and Baiardi, 2012]. This solution

presents a distributed architecture without central coordinator to avoid a single point of failure. This framework, based on event correlation, has a drawback in terms of efficiency. According to the authors, this model presents an excessive overhead to update the neural network parameters. Besides, the authors use their own dataset to validate this approach, namely, the CIDD dataset.

Lee *et al.* built a multi-level IDS and log management approach on the basis of user behaviour to better fit the cloud system requirements [Lee et al., 2011]. The method supports classifying the logs in normal or anomalous, but the network administrator is responsible for taking the final decision. In this case, the system requires human intervention to detect an attack, increasing the detection time.

Song Fu proposed a framework for autonomic anomaly detection in the cloud context [Fu, 2011]. The detection mechanism is fed through an algorithm for metric selection based on mutual information: maximal relevance and minimal redundancy. After that, a semi-supervised decision tree classifier identifies anomalies considering information such as CPU usage, memory utilization, paging fault. Real data from a university campus is employed to assess the feasibility of the solution.

Vieira *et al.* showed some particularities of five IDSs and a comparative study is presented. The IDS proposed uses ANN for anomaly detection in cloud environment, and it improves the security level by integrating two approaches to intrusion detection: behaviour- and knowledge-based [Vieira et al., 2010]. This investigation covers some characteristics such as host- and network-based intrusion detection system; data from grid and cloud computing; IDS approach and validation. In the suggested proposal, each node cooperatively participates identifying local events (user logs) that could represent security violations. The authors use simulation to assess this IDS.

Xiong *et al.* surveyed different types of security threats in cloud communication. Furthermore, to reduce security risks, they propose an IDS to detect network traffic anomaly based on synergetic neural networks and the catastrophe theory [Xiong et al., 2014]. The DARPA dataset was used to validate this approach. The results show high detection rates (83% up to 97%) and low false alarm rates (8.3% to 11.4%). Ahmed Patel *et al.* presented a taxonomy and state-of-the-art of intrusion detection and prevention systems for cloud [Patel et al., 2013]. At the end, they propose an IDS focused on four applicable concepts for cloud-based intrusion detection systems: autonomic computing, ontology, risk management, and fuzzy theory. This proposal lacks a feature extraction approach and evaluation of the feasibility.

Table 4.1: Characteristics of related works concerning Intrusion Detection System

Proposal	**Approach**	**Features**	**Feature extraction approach**	**Dataset**	**Management structure**	**Cloud scenario**
Horng S. *et al.* [Horng et al., 2011]	SVM	Protocol, duration of connection, status, etc...	Hierarchical clustering algorithm	DARPA	Individual	×
Mulay S. *et al.* [Mulay et al., 2010]	SVM	Network packets	Decisions trees	DARPA	Individual	×
Li H. and Liu D. [Li and Liu, 2010]	SVM	Signature	Netfilter and IPtables	–	Individual	×
Shon T. and Moon J. [Shon and Moon, 2007]	SVM	Protocol, source port, destination port, IP, TTL, etc...	Genetic algorithm	DARPA	Individual	×
Chen W. *et al.* [Chen et al., 2005]	SVM and ANN	Protocol, source port, destination port, IP, TTL, etc...	Frequency-based encoding	DARPA	Individual	×
Kholidy H. and Baiardi F. [Kholidy and Baiardi, 2012]	ANN	User logs and signature	Event correlation	CIDD	Collaborative	√
Lee J. *et al.* [Lee et al., 2011]	–	User logs	–	–	Collaborative	√
Song Fu [Fu, 2011]	Decision tree classifier	CPU usage, memory and swap utilization, paging and paging faults, etc...	Maximal relevance and minimal redundancy	Own dataset	Individual	√
Vieira K. *et al.* [Vieira et al., 2010]	ANN	User logs	Event correlation	Simulation	Collaborative	√
Xiong W. *et al.* [Xiong et al., 2014]	Neural network and Catastrophe theory	Hurst parameter and dynamic associate factor	Synergetic dynamic equation	DARPA	Individual	√
Ahmed Patel *et al.* [Patel et al., 2013]	Fuzzy theory	–	–	–	Individual	√
Chengwei Wang *et al.* [Wang et al., 2010]	Entropy	CPU utilization, memory virtual block write/read	Multiple analytical methods	Simulation	Individual	√

Chengwei Wang *et al.* proposed multiple analytical methods to aggregate different levels of metrics in cloud data traffic for anomaly detection using entropy [Wang et al., 2010]. The results, based on synthetic data (CPU utilization, memory utilization and VBD read/write), show that the solution outperforms threshold-based methods in accuracy and false alarm rate.

4.2.3 Discussion

Table 4.1 summarizes several IDS approaches and displays their key features. On the one hand, all the SVM models proposed for traditional networks were designed as an individual and centralized module. On the other hand, a couple of the IDSs employed in cloud computing have a collaborative design, but most of them were assessed by synthetic data (simulations) or have a conceptual model that still has to be validated.

Regardless of whether or not anomaly detection methods are reliable, some requirements are still not being met when they are employed in the cloud computing environment, such as finding the best set of features and reducing the amount of information required to describe a large set of data. In the training phase, the SVM spends an amount of time that is proportional to the amount of input data. This means that reducing the amount of data is the key factor for successfully using the SVM in the context of cloud. According to the related work, among the approaches for feature extraction, there is none that is applied to cope with the massive volume of data traffic generated by the cloud infrastructure. Thus, extracting a good set of features that represents the behaviour of the cloud network traffic remains an open issue.

In the following section, a conceptual solution for detecting anomalies in the cloud network traffic is introduced. The mechanism works by means of a SVM model that is fed with features extracted from a predictor based on a Poisson process.

4.3 Anomaly Detection Mechanism in Cloud

The purpose of the *Anomaly Detection Mechanism* is to provide an efficient method to detect anomalies in the cloud-based network traffic. Figure 4.1 depicts the basis of our mechanism, by highlighting the application scenario and the main conceptual components.

The cloud provider offers several services by the Internet, such as infrastructure, software and platform to the clients. Real-time cloud traffic data (Flow 1) is continuously being gathered from the cloud environment by the *Cloud Monitoring* module. This information is subsequently processed by the *Feature Extraction Approach* that performs prediction based on information such as the protocol type, the number of network packets and timestamp. After that, the *SVM Model* is fed with features extracted from the aggregated data (Flow 2). Then, the *SVM Model* triggers a warning to the *Event Auditor* when an anomalous

behaviour is detected (Flow 3). In the meantime, the *Repository of Outcomes* component stores a detailed output regarding the historic of the Virtual Machine (VM) operation (Flow 4). Furthermore, the *Event Auditor* represents an agent placed in the VM that is able to communicate collaboratively with agents in the other VMs. This agent receives any anomalous event from the *SVM Model* and builds a message with information of all components (Flow 5) for sending alerts to other agents. Having presented an overview of the anomaly detection mechanism, in the following subsections there will be a more detailed description of the components.

4.3.1 Cloud Monitoring

Detecting anomalies in cloud network traffic requires access to detailed information about the operation of the network. In this context, the *Cloud Monitoring* has to continuously monitor the service provided by the virtual machine. Thus, the *Cloud Monitoring* is able to take the cloud network traffic patterns during a given period. In other words, this component is responsible for recording all the incoming and outgoing network packets in a virtual machine. As input, raw data is gathered continuously from the cloud network traffic. The next step is building a time series that will be analysed for the following prediction. As output, the *Cloud Monitoring* prepares the collected data by measuring the number of packets in the network traffic at regularly spaced intervals, and thus forms a discrete time series ordered by time. It is not within the scope of this study to propose a particular approach for monitoring the cloud infrastructure. However, several tools have the potential to monitor the cloud computing environment with the aid of distributed agents in virtual machines, such as Nagios, OpenNebula, and Nimbus [Aceto et al., 2013].

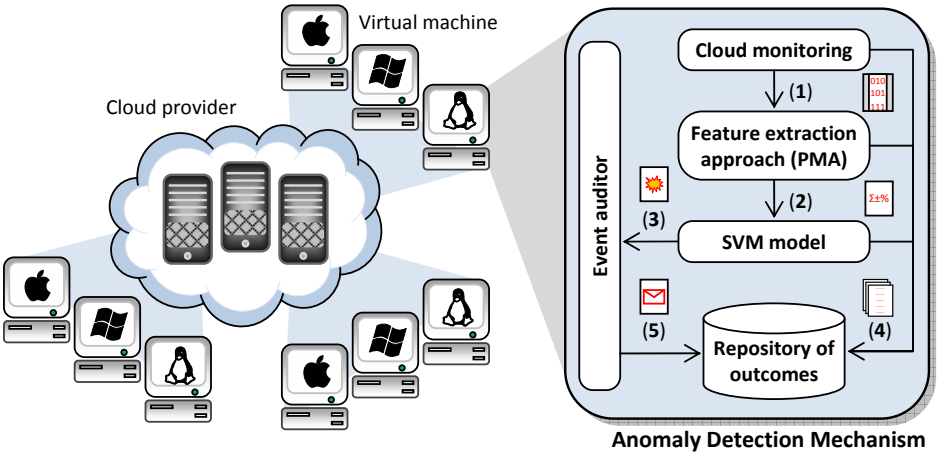


Figure 4.1: Application scenario and elements of the proposed mechanism

4.3.2 Feature Extraction Approach

Extracting features from the network traffic is of crucial importance for providing a better performance by adopting a non-parametric approach such as using a Support Vector Machine. In our view, feature extraction involves reducing the amount of information required to describe a large set of data, therefore enabling its processing by the SVM. Besides, it creates new features from functions of the original data. In this context, we propose a supervised learning technique that operates through a multi-level representation of data. Our approach uses multiple temporal layers of data for feature extraction so that it can express data in a compact representation by removing redundancy.

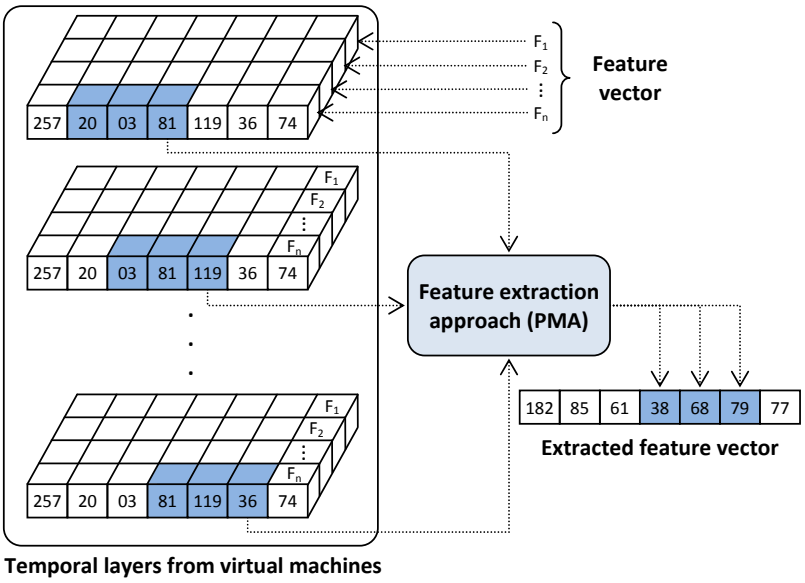


Figure 4.2: Feature extraction approach based on PMA

Figure 4.2 illustrates the process of extracting features. These features $(F_1, F_2, ..., F_n)$ are gathered from the *Cloud Monitoring* process described in the last subsection and they characterize the real operation of a virtual machine. The set of features is arranged into vectors. Each layer contains all the feature vectors from a virtual machine at time t. A sliding window of a given size is used to weight past observations of data traffic according to the Poisson Moving Average (PMA) predictor model [Dalmazo et al., 2014]. Based on the PMA methodology, at each time period, out feature extraction approach produces:

- a unique value that results from condensing the temporal layers by weighting past observations following a Poisson distribution;
- a predicted value for the subsequent time period.

These values are then used as input to the SVM for detection of anomalies.

Kind *et al.* identified a set of relevant features for network anomaly detection [Kind et al., 2009]. Cloud computing generates large amounts of monitored data, thus calling for a methodology to summarize this information. As detailed

in Table 4.2, the set of features from Kind *et al.* over which we perform our extraction technique consist of: the type of the protocol, the port number, the packet size, the number of packets, time interval between anomalies, the variance between real network traffic and predicted network traffic (Δ-variation) and attack type.

Table 4.2: Details of the extracted features

Feature	**Description**
Protocol type (f)	Dividing the network traffic by protocol type facilitates identifying anomalies not visible in the global network traffic
Destination Port number (f) Source Port number (f)	Port number analysis is useful for revealing attacks that attempt to scan ports
Source IP address (f) Destination IP address (f)	This information is useful for recognizing Denial of Service attacks
Packet size (f)	The sudden increase of this feature can indicate a SYN flood attack
Number of packets (c)	Consists of control information and user data used in the prediction
Time interval (c)	A time set containing information between the beginning and the end of the anomaly
Δ-variation (c)	The absolute difference between the real network traffic and the predicted network traffic
Attack type (f)	It describes which is the attack that is being executed
Alarm (f)	Boolean variable that indicates the presence of an alarm

These features are divided into two types: frequency (f) features (*e.g.* the number of times a packet from a protocol appears) and cumulative (c) features (*e.g.* the total number of packets received in a time period). For cumulative features, PMA aggregates values from different temporal layers and estimates the future behaviour of the network; for frequency features, PMA determines which port numbers are most common by a frequency function, and then estimates the occurrences of the port numbers for each slice of time. With the aid of the PMA algorithm, the outcome (*extracted feature vector*) contains just the most accessed port numbers. This operation is analogous to the other frequency features.

The *Poisson-based Predictor* [Dalmazo et al., 2016a] represents the core of our feature extraction approach. PMA has been adopted because of its high accuracy in terms of network traffic prediction, and its ability to generate a subset of representative features. It is worth noting that the feature extraction approach enhances generalisation by reducing the variance in the data. In other words, by employing PMA, the outliers (in a time series) are smoothed inside a sliding window but still noticeable in a global context. The PMA acts as a method of aggregating values over time so that the issue of processing large amounts of information by the SVM can be addressed, while still representing the data with sufficient accuracy.

Combining different traffic features is useful for detecting anomalies. For instance, joining source IP addresses, the port number and the number of packets is a potential mean of characterizing a DoS attack. At the end of this process, the extracted feature vector is available for feeding the SVM model.

4.3.3 SVM Model

The Support Vector Machine (SVM) is a supervised learning model that evaluates data and identifies patterns with the goal of classifying the data. SVM model uses a methodology for choosing the best hyperplane (among many others) that represents the largest margin between two classes, namely, normal network traffic and anomalies in this work. Then, the hyperplane is chosen such that the distance from it to the nearest support vector on each side is maximized [Deng et al., 2012], as illustrated in Figure 4.3.

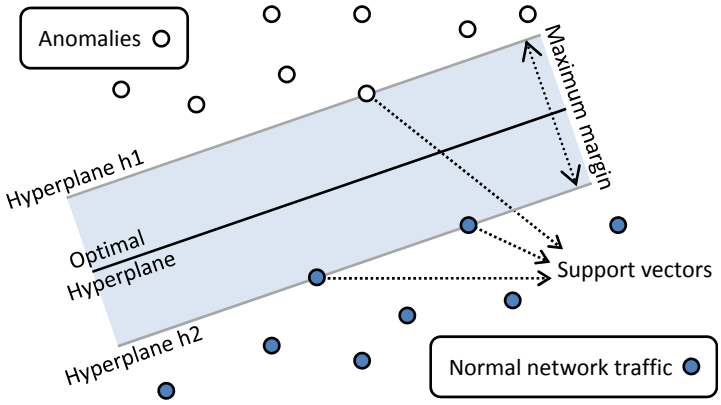


Figure 4.3: Support vectors and the optimal hyperplane for binary classification

The Support Vector Machine learning model includes two stages: training and testing. The first learns the two possible patterns of the network traffic (the normal and the anomalous behaviour). The second tests the knowledge achieved in the past stage to detect unknown anomalies. Separating data into training and testing data is an important part of validating the SVM model. At this point, the risk of learning from compromised data is reduced once the SVM can distinguish between the regular traffic and the anomaly. By this, we can minimize the effects of data inconsistencies and better understand the characteristics of the data. Once the SVM model has been processed by using the training set, it is needed to evaluate the prediction capability against the training set. Considering the data in the testing set already contains known values for the attribute that we want to predict, it is possible to determine whether the suggestions of the model are correct.

Summing up, the anomaly detection for the cloud network traffic based on SVM with PMA expresses a process of recognizing an unexpected comportment. In this process, the training data represents the standard pattern and the testing data alludes to identify such pattern. The process of identifying a particular behaviour inside of the testing data is a mapping process of the testing data in some existing pattern of the training data.

4.3.4 Repository of Outcomes

The component called *Repository of Outcomes* is a database that brings together a set of information used to describe the anomaly detection mechanism activities.

This database is done on the basis of the network traffic behaviour and its goal is to keep track of the virtual machine operating history. Furthermore, this information can be used to support the cloud decision-making. In this case, it can be used as subject to investigation by the operator or serves as a foundation to trigger alerts to other agents from other cloud services.

4.3.5 Event Auditor

In order to detect an anomaly, auditing events that describes the environment and the state of the system is needed. In this context, the *Event Auditor* consults the *Repository of Outcomes* periodically looking for unexpected occurrences. Once the *Event Auditor* finds a suspicious action, it will gather information from the *Repository of Outcomes* to build an alert message. The alerts represent that the current course of an event could be in some way dangerous or detrimental to the system. Although the anomalous pattern is being captured by the *Event Auditor*, other virtual machines may be unaware this event. Thus, the *Event Auditor* can collaborate with agents from other virtual machines so that appropriate actions are taken. This paves the way to identification of distributed attacks.

4.4 Evaluation and Discussion

Throughout this section, the anomaly detection model based on SVM that is used to assess this work is presented. Furthermore, we evaluate the effectiveness of the mechanism regarding the most common metrics found in similar studies in the literature. In addition, the results are compared with several SVM models outlined in Section 4.2. We consider two case studies for evaluation: DARPA [Haines et al., 2001] and CAIDA [Hick et al., 2007] datasets. Finally, a sensitivity analysis of the model showing the trade-off between the time granularity and the accuracy is provided.

4.4.1 Metrics

Typical metrics to evaluate the effectiveness of an anomaly detection system are: detection rate (DR), false positive rate (FPR) and false negative rate (FNR). The DR is the number of correctly classified as normal packets divided by the total number of the data of a test dataset (or true negative plus false positive), as showed in Eq. (4.1). The FPR is defined as the total number of normal data traffic, which were classified as anomalies wrongly, divided by the total number of normal data traffic (or true negative plus false positive), as depicted in Eq. (4.2). The FNR is expressed as the total number of abnormals data that were incorrectly classified as normal traffic, divided by the total number of real abnormals data (or true positive plus false negative), as illustrated in Eq. (4.3).

These metrics from the confusion matrix produce a numeric value that allows the comparison among other anomaly detection approaches. They are depicted in Eq. 4.1, Eq. 4.2 and Eq. 4.3.

$$DR = TN/(TN + FP) \tag{4.1}$$
$$FPR = FP/(TN + FP) \tag{4.2}$$
$$FNR = FN/(TP + FN) \tag{4.3}$$

4.4.2 DARPA Case Study

The Cyber Security and Information Sciences Group of MIT Lincoln Laboratory, under Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory sponsorship, collected the first standard dataset for evaluation of computer network intrusion detection systems [Haines et al., 2001]. This dataset was the first formal, repeatable, and statistically significant evaluation of intrusion detection systems. We would like to point out that the DARPA data set is a renowned data set for anomaly detection. Although the data set was created in 1998/1999, it is still being used by many works, including recent works in the context of the cloud [Xiong et al., 2014, Ganeshkumar and Pandeeswari, 2015, Liu et al., 2012].

The datasets contains data collected from February 1998 up to October 1999. The data consists of three weeks of training data and two weeks of test data. The first and third weeks of the training data do not contain any attack. The second and the fourth week of the training data contains a select subset of labelled attacks. In this work, we use the first and the second week for the *training phase* and the third and the fourth week for the *testing phase*. In order to make the dataset more realistic, we organized many of the attacks so that the resulting data sets consisted of 10% attacks and 90% normal traffic (for both datasets, *training phase* and *testing phase*).

4.4.2.1 Parameter Setup

The SVM model proposed in this work was implemented with LIBSVM version 3.20. LIBSVM is an integrated software for support vector classification, regression and distribution estimation. We consider the Radius Basis Function (RBF) kernel as SVM algorithm. RBF is a real-valued function whose value depends on the distance from the origin or on the distance from some another point called by *center*. The Euclidean distance is the main example of a Radius Basis Function. This kernel presents two parameters: C and γ. C is the parameter for the soft margin cost function, which controls the influence of each individual support vector. This process involves trading error penalty for stability. The γ is the free parameter of the Gaussian radial basis function, it defines how far the influence of a single training example reaches.

We use a *grid-search* on C and γ using the cross-validation process (performed automatically by the LIBSVM). In this process, several pairs of (C, γ) values are tried and the one with the best cross-validation accuracy is picked. In this process, for the DARPA dataset, the best pair is: (32768, $3.05e^{-5}$) and the best pair for the CAIDA dataset is (512, 0.5). As illustrated in Figure 4.4 and Figure 4.5, respectively.

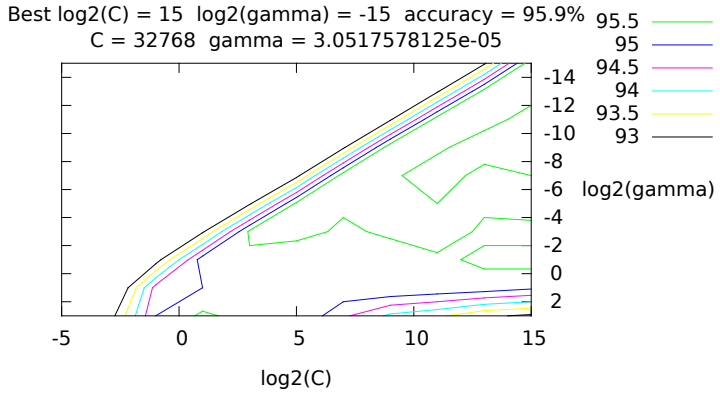


Figure 4.4: Grid searching for the best pair (C, γ) in the DARPA dataset

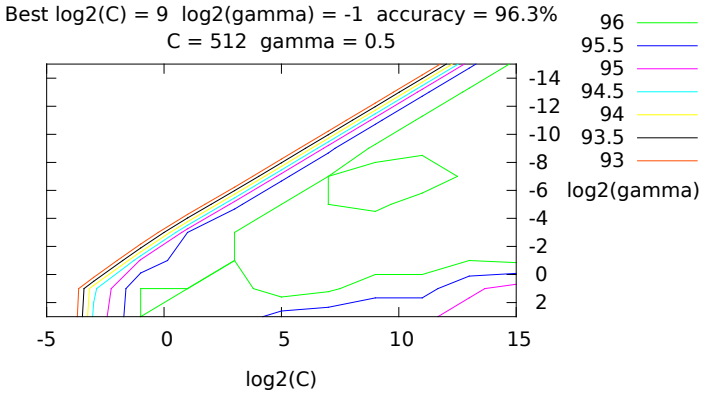


Figure 4.5: Grid searching for the best pair (C, γ) in the CAIDA dataset

4.4.2.2 Accuracy of the Model

Table 4.3 shows the comparison among several approaches that use SVM and DARPA dataset to validate the model. Regarding detection rate (DR) point of view, Soft margin SVM with Radial Basis Function (RBF) kernel obtained 97.48%, but at cost of high false negative rate (FNR), more than 11%. Another model with high DR, but low FPR, is the approach proposed by Chen W. *et al.* [Chen et al., 2005]. Although the model hits almost 90% of the time, it showed more than 10% of false positive rate (FPR). Other models presented in the Table 4.3 present at least one drawback: low accuracy, high FPR or high FNR.

In summary, our method on the basis of SVM and RBF kernel with features extracted from Poisson Moving Average predictor presents the best equilibrium in the results. It reaches 98.56% of detection rate and 8% of FNR. Also, our approach displays the lowest FPR among the related work, just 1.44%.

Table 4.3: Approaches that use SVM and DARPA dataset

Approach	**Kernel**	**DR(%)**	**FPR(%)**	**FNR(%)**
LIBSVM and PMA	RBF	98.56	1.44	8.00
Horng S. *et al.* [Horng et al., 2011]	RBF	95.72	N/A	N/A
Enhanced SVM [Shon and Moon, 2007]	Sigmoid	89.59	10.41	27.27
Soft margin SVM [Shon and Moon, 2007]	Inner product	89.52	10.48	4.36
Soft margin SVM [Shon and Moon, 2007]	Polynomial	94.80	5.20	10.45
Soft margin SVM [Shon and Moon, 2007]	RBF	97.48	2.52	11.09
Soft margin SVM [Shon and Moon, 2007]	Sigmoid	96.06	3.94	12.73
One-class SVM [Shon and Moon, 2007]	Inner product	52.67	47.33	36.00
One-class SVM [Shon and Moon, 2007]	Polynomial	54.57	45.43	46.00
One-class SVM [Shon and Moon, 2007]	RBF	82.23	17.77	44.00
Chen W. *et al.* [Chen et al., 2005]	RBF	89.65	10.35	N/A

4.4.3 CAIDA Case Study

The CAIDA (Center for Applied Internet Data Analysis) DDoS Attack 2007 Dataset contains a traffic trace of a DDoS attack. This dataset contains pseudonymised traces occurred on August 4, 2007 for approximately one hour (20:50:08 UTC to 21:56:16). The entire CAIDA dataset is divided into 5 minutes packet capture (pcap) files. Only attack traffic to the victim and responses to the attack from the victim are included in the trace. The trace corresponds to a Ping Flood Attack that greatly increases the ICMP packets in the network traffic. The attack in the dataset is not labelled, precluding the training phase of the machine learning algorithms. However, to overcome this gap, the evaluation of the anomaly detection model through the CAIDA dataset was used just in the *testing phase*. We use the DARPA dataset for the *training phase*.

4.4.3.1 Parameter Setup

For the CAIDA case study, we consider the same RBF kernel as SVM algorithm. The *grid-search* presented (512, 0.5) as the best values for C and γ, as illustrated in the cross-validation process in Figure 4.5.

4.4.3.2 Accuracy of the Model

The CAIDA dataset contains around 66 minutes of network traffic monitoring. This dataset exposes a flood attack that begins after 25 minutes until the end of the monitoring process. In this case, the anomaly detection approach was able

to entirely identify the attack (100% of DR and 0% of FPR) with delay less than 5 minutes.

4.4.4 Sensitivity Analysis

The goal of the sensitivity analysis is providing a better understanding of the relationship between the time granularity and the accuracy of the model. This evaluation is used within specific sliding window boundaries that will reflect on the effectiveness of the solution, such as the effect that changes in time granularity will have on the model detection rate.

In the literature, similar studies usually perform anomaly detection considering periods of 5 minutes as monitoring window size. This consensus acts as a default benchmark that facilitates the performance comparison among different approaches. Table 4.4 depicts the anomaly detection approach accuracy facing three different sliding window sizes. It is worth noticing that all the sensitivity analysis is performed in the same dataset, namely, DARPA dataset.

The first line shows the performance of the model using 1 minute to aggregate data in the monitoring process. For this scenario, the model presents around 96% of DR and the worst FPR, reaching almost 4%. In the second line, it is presented the results for the default time granularity used in this work, 5 minutes. In this case, the SVM model achieves the result displayed before in this chapter (98.56% of DR and 1.44% of FPR).

Table 4.4: Sensitivity of the model

	Model	**Time granularity**	**DR(%)**	**FPR(%)**
1	LIBSVM and PMA	1 minute	96.10	3.90
2	LIBSVM and PMA	5 minutes	98.56	1.44
3	LIBSVM and PMA	10 minutes	98.61	1.39

Although the 10 minutes of time granularity presented the best results (98.61% of DR and 1.39% of FPR), it was achieved at the cost of higher monitoring time. More specifically, the 10 minutes case shows a small advantage in comparison with the 5 minutes case, yet requiring double the monitoring time. Also, note that the DARPA documentation [Haines et al., 2001] shows that each attack takes, on average, 9 minutes and 3 seconds to be performed. This further attests the irrelevance of increasing the monitoring time to 10 minutes, as most of the attacks will be successfully performed before the anomaly was detected.

4.5 Summary of the Chapter

In this chapter, an attempt has been presented to shed light on the main obstacle to the adoption of the cloud service models: the lack of security. To address this problem, an approach to detect anomalies in the cloud scenario was proposed. This work differs from previous anomaly detection techniques since it relies on a

collaborative mechanism that combines a Support Vector Machine model with features extracted from a Poisson Moving Average predictor.

By analysing the results of the evaluation, it can be seen that the anomaly detection mechanism was able to detect anomalies by means of two case studies with real data. The SVM model achieved a high degree of accuracy. In particular, compared with other approaches, the best level of detection rate and the second best number of false negative rates were achieved. The sensitivity analysis has shown the trade-off between the time granularity and the accuracy of the model, showing that our scheme performs accurate detection within a 5 minute timeframe. Finally, it is worth pointing out that the mechanism outperforms other approaches in the literature, owing to the high quality of the features extracted from the Poisson-based predictor, such as its accurate prediction.

This chapter resulted in the following papers::

- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, "**Security and Trustworthiness in Cloud Computing**", in Meeting with Science and Technology in Portugal, 2017
- Dalmazo, Bruno L. and João P. Vilela and Simões, P. and Marilia Curado, "**Expedite feature extraction for enhanced cloud anomaly detection**", in NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016
- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, "**A SVM Model based on Network Traffic Prediction for Detecting Anomalies**", in 21th edition of the Portuguese Conference on Pattern Recognition, 2015

Chapter 5

Triple-Similarity Mechanism for Alarm Management

For a successful technology, reality must take precedence over public relations, for Nature cannot be fooled.

(Richard P. Feynman)

Contents

Its distributed nature and ubiquitous service make the cloud subject to several vulnerabilities. One of the main tools used for reporting suspicious activity in the network's traffic is the Intrusion Detection System. However, two significant problems arise: the huge volume of control messages between the virtual machines and the servers; and the associated transfer costs.

5.1 Introduction

An IDS is designed to monitor a system or a network in order to report any suspicious activity that may compromise its operation. The report of the suspect activity represents an output of the IDS, namely, an alarm. Usually, alarms carry information about the suspicious activity such as: type of attack, the timestamp, the number of packets, the IP address and the port number. Thus, alarms are considered valuable information to support the administrator in decision-making about whether it is a true attack or a false alarm which came from one or more collaborative IDSs [Rittinghouse and Ransome, 2016] [Zissis and Lekkas, 2012].

An IDS may be based on two main approaches to recognize an attack (or attempt) that differ in the way the data is analysed and processed. The signature approach refers to the detection of attacks by looking for specific patterns based on other similar attacks, while the anomaly approach consists in searching for deviations from proper behaviour through periodic observations of the system. Signature-based detection methods usually present a low number of false alarms but do not have the ability to detect new or variants of known attacks, while anomaly-based detection has the benefit that a new attack, for which a signature does not exist, can be detected if it falls out of the regular traffic patterns.

Intrusion detection in a cloud environment involves other aspects that need to be considered, for instance, the relationship between the server and the virtual machine (VM). Usually, a server may host hundreds of virtual machines that provide different services, for instance, storage, web server, e-mail, and others [Mell and Grance, 2011]. Another important feature relates to where the information in question will be collected and processed. In this case, the information may come from the infrastructure, platforms of software development or applications. Furthermore, the distributed architecture design of clouds is seen as the key point on which IDSs rely for detecting threats.

The distributed nature and ubiquitous service make cloud computing vulnerable to several types of attacks. For example: a denial of service attack, data privacy and integrity, identity management and access control, and others [Ali et al., 2015, Hudic et al., 2017]. Furthermore, the amount of alarms generated can be overwhelming [Ballani et al., 2011], thus requiring alarm management solutions for an effective management of resources. Managing alarms triggered by traditional intrusion detection methods is even more challenging in the cloud computing environment. In this case, the network traffic is apt to undergo sudden changes and these may be easily confused with traffic anomalies [Plonka

and Barford, 2009].

5.1.1 Requirements and Open Issues for Managing Alarms in the Cloud

In recent years, new approaches regarding alarm management have been proposed in the literature such as alarm correlation [Benferhat et al., 2013], regular expression matching [Li et al., 2010] and clustering alarms [Lo et al., 2010]. However, these studies are concerned with increasing the number of true alarms and they fail to respond appropriately to a low number of false alarms or decrease the number of control messages in general [Patel et al., 2013].

The number of alarms generated over time is even greater in cloud computing. Besides the sudden changes that the traffic suffers due to the elastic and scalable nature of cloud environments, the number of messages increases proportionally with the number of virtual machines. Moreover, it is known that around 99% of the alarms are false both in cloud computing [Patel et al., 2013] and in traditional environments [Elshoush and Osman, 2011] [Hubballi and Suryanarayanan, 2014] [Di Pietro and Mancini, 2008]. The wide disparity between the true and false alarms generated has certainly compromised the performance of IDSs.

The problem is further aggravated in cloud computing due to the huge volume of control messages between the virtual machines and the server. This situation makes the detection system inefficient because it provides an unmanageable amount of alarms for the administrators [Elshoush and Osman, 2011]. In addition, according to a technical report by the University of California, the cost of the data transfer lies in the region of $100 to $150 per terabyte [Fox et al., 2009]. Therefore, besides reducing the number of alarms and supporting the management of the cloud infrastructure, managing alarms also facilitates minimizing the network's bandwidth and the associated transfer costs.

From these observations a set of key requirements for managing alarms in the cloud emerge, which are listed as follows:

1. **Self-adaptive**: this requirement refers to the model's ability to learn or train itself based on current information, which is different from static approaches. Cloud computing ensures elasticity which provides scalability. In this context, the cloud provider should be able to preserve its operation under conditions of unexpected change by constantly evaluating its own behaviour.

2. **Low message overhead**: as important as decreasing the false alarm rate, a key point is to reduce the number of control messages between the server and virtual machines. An alarm reduction technique is an absolute necessity for solving this problem [Hubballi and Suryanarayanan, 2014]. This requirement calls for a compatible model for detecting attacks and classifying alarms dynamically, generating the minimum possible workload.

3. **Collaborative**: this requirement is characterized by the sharing and construction of knowledge among multiple information sources in order to accomplish a task. A large number of heterogeneous entities usually have different information, and combing them can potentially provide better alarm management to the cloud networks and their applications. A collaborative approach is particularly well suited to the cloud environment because these entities have to communicate continuously in order to support decision-making.

4. **Distributed**: an approach in which components are located in different virtual machines and coordinate their actions by passing messages represents, in this context, a distributed alarm management. This feature ensures that the VMs interact with each other in order to join forces to recognize an attack. Moreover, adverse events generated by an individual failure may be minimized [Arshad et al., 2013].

By following this set of key requirements, it is possible to devise an alarm management system suitable for cloud computing.

5.1.2 Contributions and Outline

In order to address these issues and requirements, we have made several contributions in this work: (*i*) grouping similar alarms that may correspond to the same attack or attack attempt in order to reduce the number of messages sent to the server/administrator and; (*ii*) using the number of occurrences of these groups to adjust the severity of a single alarm based on a similarity analysis. From these contributions, we intend to optimize the efficiency for generating alarms, decreasing the network data traffic to manage IDS and its associated transfer costs.

The remainder of the chapter is organized as follows. Section 5.2 covers some of the most prominent related work. Section 5.3 describes the proposed solution and the methodology used for this thesis, whilst Section 5.4 presents the evaluation and discusses the results. Section 5.5 summarizes the chapter.

5.2 State of the Art

Improving alarm management in the cloud is a useful means of supporting the cloud provider to manage its assets. Besides decreasing the number of false alarms, it may reduce the amount of alerts that need to be handled. In this section, latest research findings are organized into two parts. First, several approaches for alarm correlation are presented, outlining the main benefits and drawbacks of each technique. Finally, a discussion about the state-of-the-art and the open issues is provided.

5.2.1 Managing Alarms

There are numerous approaches to improve the quality of alarms by increasing the number of false positives, such as fuzzy logic [Patel et al., 2013], artificial neural networks [Vieira et al., 2010], decision tree classifier [Fu, 2011] among others. Other approaches on alarm management aim to decrease the false alarms rate by recognizing relationships between them [Hubballi and Suryanarayanan, 2014]. The main approaches found in the literature are described below.

Zhichun Li *et al.* [Li et al., 2010] applied signature detection techniques for enhancing an IDS looking for vulnerabilities in a high speed network. An approach that uses semantic information can potentially reduce the number of false alarms. They decreased the false alarm rate based on a signature parsing and regular expression matching engine by using the Single PDU Multiple Signature Matching algorithm. However, there is no a real implementation of this proposal yet, just a small prototype implementation which can handle a limited number of protocols.

Parikh and Chen [Parikh and Chen, 2008] used statistical pattern recognition techniques among multiple sources of information to decrease the cost of operations associated to intrusion detection activities. Different errors in classifying network traffic generate different costs associated with them, for instance, the cost of a false alarm and the useless traffic generated by it. Although the cost minimization strategy has been successful based on a objective function minimization, the capabilities of learning incrementally and adaptively are not assessed which is important due to the dynamically changing characteristic of network traffic in cloud environments.

Lo *et al.* [Lo et al., 2010] extended the Snort IDS. In this proposal, four modules are created to work in cooperative mode: intrusion detection, alarms clustering, threshold computation and comparison. The intrusion detection is based on analysis of the number of packets over time. In comparison with pure Snort-based IDS, they showed that the solution spends almost the same time to compute the detection. The benefit of the proposed modules is preventing the system from a single point of failure attack for cloud environment.

Salem Benferhat *et al.* [Benferhat et al., 2013] proposed a generic approach that can be applied for any classifier (e.g. Hidden Naive Bayes, decision tree classifiers, etc) for alarm correlation. The term "expert knowledge" refers to a person who has extensive skill or knowledge in a particular field. Then, they leverage the expert knowledge for increasing the accuracy of the classification model. However, approaches based on Neural Networks have a serious limitation: they learn the training patterns but lose the ability to make generalizations. For instance, a new adaptation of a known attack may not be detected, thus limiting finding slight variations of instances already classified by the model.

Table 5.1: Characteristics of related works concerning requirements for alarm management

Proposal	**Self-adaptive**	**Low message overhead**	**Collaborative**	**Distributed**	**Cloud scenario**
Zhichun Li *et al.* [Li et al., 2010]	×	×	×	×	×
Parikh and Chen [Parikh and Chen, 2008]	×	×	√	×	×
Lo *et al.* [Lo et al., 2010]	×	×	√	√	√
Salem Benferhat *et al.* [Benferhat et al., 2013]	√	×	√	√	×
Leau Beng *et al.* [Beng et al., 2014]	√	√	√	×	×
Elshoush and Osman [Elshoush and Osman, 2012]	×	√	√	√	×

Leau Beng *et al.* [Beng et al., 2014] perform an extensive study about existing efforts to address the identification of similarities and causality relationships between alerts. They elect the main problem that researchers are trying to solve, namely, the generation of large number of alerts and false positives. Furthermore, they point out the most popular alert correlation approaches with their advantages and drawbacks. This paper surveys existing works and does not propose a new solution.

Elshoush and Osman [Elshoush and Osman, 2012] presented a multiple components approach for dealing with different features of alert correlation, each responsible for a different aspect of the overall correlation aim. However the sequence order of acting components affects the process performance. In this context, they introduce a method based on the Alert Fusion Algorithm to merge unrelated alerts and thus reducing the number of messages. Nevertheless, the total time needed for processing this approach increases depending on the number of alerts triggered in each component.

5.2.2 Discussion

This section compares several approaches by taking into account the requirements for managing alarms in a cloud computing environment. In particular, it summarizes the related work regarding their key requirements, application in cloud context and the remaining open issues as illustrated in Table 5.1.

IDSs are designed to report basic events that are malicious in nature but powerless to recognise causal relationships between the consecutive instances of an attack. Approaches focused on alarm management are a potential solution for reducing the number of false or missed alerts. However, these approaches can not deal with a low number of messages and a self-adaptive solution in a distributed architecture at the same time. Moreover, the related work approaches are not designed for adapting itself to the elastic and scalable nature of cloud environments because they fail to learn based on current information. Furthermore, the cloud environment lacks an approach for alarm management able to cope with large amounts of information and control messages. In addition, all types of alarms are processed in the same way in the current approaches, without any level of distinction between the severity of alarms.

In order to tackle these limitations, we introduce a conceptual solution for managing alarms in the cloud network context by means of a similarity analysis between the features extracted from the IDS, in the following section. From this, it is expected to improve the performance in terms of the usage of the network's bandwidth and the number of alarms. More specifically: the solution proposed reduces the network data traffic to manage IDS and its associated transfer costs; and raises the severity of an individual alarm based on the number of occurrences of similar alarms.

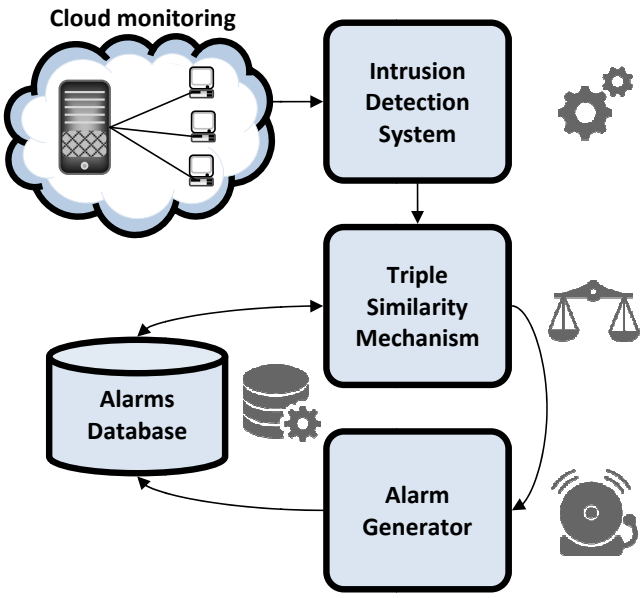


Figure 5.1: Conceptual components for managing alarms in cloud

5.3 Triple-Similarity Mechanism

The purpose of this proposal is to provide an efficient method to aggregate similar alarms in cloud-based network traffic. Figure 5.1 depicts the basis of our proposal, by highlighting the application scenario and the main conceptual components.

Monitoring data from cloud infrastructures is the input of the system. From this data, an *Intrusion Detection System* works by supervising any suspicious network activity. Whenever necessary, it triggers alarms that hold vital information about the abnormal activities. At this point, the *Triple-Similarity Mechanism* relies on data from the *Alarms Database* for aggregating alarms and, consequently, reduces the network data traffic and decreases the associated transfer costs. Then, the *Alarm Generator* produces a single alarm at a higher level of severity. We now describe each component in more detail.

5.3.1 Intrusion Detection System

At this point, the *Intrusion Detection System* monitors the network traffic generated by the service provided by the virtual machine and its applications during a given period. In this process, raw data is gathered continuously from the cloud network traffic in order to build a network baseline. As an output of this process, the IDS prepares the data collected by measuring the number of packets in the network traffic at regularly spaced intervals, and thus forms a discrete time series ordered by time.

It is important to mention that the similarity mechanism is an open model that could be applied individually or collaboratively for different IDSs. However, for this work we use an IDS that generates alarms based on a distributed mechanism

that combines a Support Vector Machine model with features extracted from a Poisson Moving Average predictor [Dalmazo et al., 2017a]. The features are classified into two types: frequency (f) features (*e.g.* the number of times a packet from a protocol appears) and cumulative (c) features (*e.g.* the total number of packets received in a time period), as presented in Table 4.2.

5.3.2 Proposed Mechanism

The *Triple-Similarity Mechanism* (T-SyM) acts upon two sets of features: the dataset of alarms and the network traffic. The entire process for determining the similarity level comprises 3 steps: individual similarity, clustering relevant features and generating the output.

- Firstly, the T-SyM needs to measure the individual similarity of each feature with the *Alarms Database*. Giving preference to simplicity, the individual similarity is given by the Euclidean Distance between both features from the alarm dataset and the IDS output. Figure 5.2 illustrates *First Similarity* which compares two vectors: the vector from the IDS output and the vector from the *Alarms Database* which holds information about previous attacks. These vectors hold all the features that describe the alarm. In order to consider the different types of features generated by the IDS, the similarity approach relies on the set of features responsible for identifying the attack.

- It is then necessary to cluster relevant features based on the individual similarity level, as illustrated at the *Second Similarity* in Figure 5.2. The feature's behaviour is correlated according to the type of attack. For instance, in a Denial of Service, the attacker uses more than one unique source IP address, often thousands of them, so a high value for this feature is expected (source IP address) [Yan et al., 2016]. On the other hand, the destination IP address is restricted to only one or a small set, in this case a low number of occurrences for this feature is expected. Therefore, the set of features representing this attack presents a strong individual similarity level for some features and a weak individual similarity level for others. In order to avoid discrepancies between features, a clustering approach is required, namely, the k-means algorithm. This procedure divides the features into two groups: relevant and not relevant.

- Finally, an approach for comparing the similarity of the two groups is needed. The literature presents several techniques with this aim: the Sorensen index, the Jaccard index and the Tanimoto coefficient. However, the Sorensen index and the Jaccard index are metrics that only measure the similarity between objects of purely binary attributes. The Tanimoto coefficient, on the other hand, is not restricted to working with only binary attributes. In *Third Similarity* all similar alarms that correspond to the same attack are aggregated based on the Tanimoto coefficient. At this point, this procedure is useful for minimizing the number of control messages sent to the server/administrator of the cloud provider. In addition,

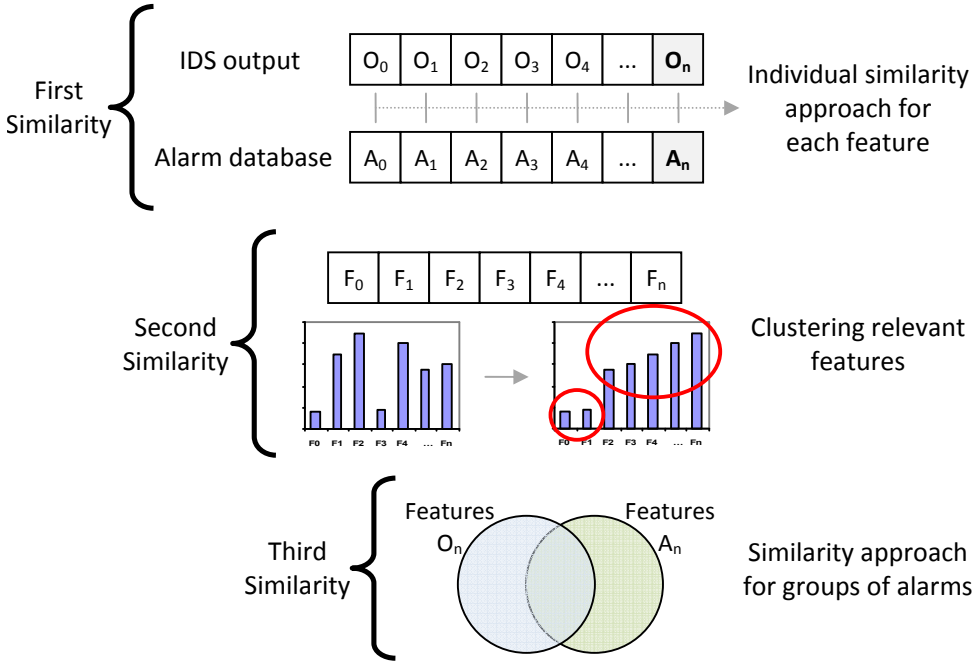


Figure 5.2: The triple-similarity mechanism

the severity of a single alarm increases according to the number of similar occurrences. This procedure is expected to enhance the alarm management in the cloud by reducing the network data traffic and its associated transfer costs.

5.3.3 Alarms Database

The component called *Alarms Database* is a repository that stores a set of information used to describe an alarm triggered by the IDS. Besides keeping track of the IDS operating history, the *Alarms Database* also includes a finite number of categories by which each alarm is classified. Then, before sending an alarm to the server, the *Triple-Similarity Mechanism* looks for a similar alarm category inside the *Alarms Database*. This is useful to decrease the number of messages, as various alarms can be replaced by just one but with a greater impact.

5.3.4 Alarm Generator

The IDS causes the system to generate a signal regarding the suspicious activity as soon as it is recognized. Algorithm 5.1 describes the procedure to calculate the severity level for alarms. The *Alarm Generator* calculates the level of severity based on the *Alarms Database*. Optionally, an *operator* of the cloud infrastructure may interact with the *Alarm Generator* to influence the process for evaluating the level of severity for the alarms. He/she may determine, based on his/her knowledge, how important an attack is, in comparison with others by increasing the initial severity level. For instance, an operator may configure the

algorithm to prioritize an occurrence of DoS attack instead of Portsweep (see variable *opK*).

Algorithm 5.1 starts by analysing the network traffic inside a sliding window provided by the IDS. This procedure seeks alarms and classifies them according to the *Alarms Database* and then assigns a level of severity. The ω function calculates the severity based on the number of alarms and the operator's knowledge. The ω function is based on a regression model that is built from the network traffic behaviour [Chatterjee and Hadi, 2015]. The *Alarm Generator* presents an interval tag that specifies a period of time to wait before sending similar alarms. All information about the attacks and the size of the interval to be monitored is given by the IDS. This is an important aspect that ensures the generalization of the *Triple-Similarity Mechanism* for working with other IDSs.

Algorithm 5.1. Severity Adjustment of Alarms

Input: Network traffic, $nTraffic$
Operator knowledge, $opK[\,]$
Output: Severity for alarms, $severity[\,]$

```
1:  Start
2:      procedure GetSeverity(nTraffic, opK[ ])
3:          var slidingWindow = idsWindow(nTraffic)
4:          for each element e in slidingWindow do
5:              if (hasAlarm(e)) then
6:                  var vType = classify(e)
7:                  var nAlarms[vType] + +
8:              end if
9:          end for
10:         for each element e in nAlarms[ ] do
11:             var severity[e] = ω(nAlarms[e], opK[e])
12:         end for
13:         return severity[ ]
14:     end procedure
15: End
```

Without this control mechanism, events of this nature may occur often and cause many alarms to be generated. Moreover, they may correspond recurrently to the same anomaly or attack. To avoid this, alarms will only be sent for the first event and after that, each occurrence with the same features represents an increase in the severity according to the severity level defined by the operator for different types of attacks. This prevents the network and the server from being flooded with redundant alarms as we will now demonstrate in the evaluation.

5.4 Evaluation and Discussion

To evaluate the effectiveness of the solution, we conducted a case study in which an Intrusion Detection System generates alarms based on a distributed mechan-

ism that combines a Support Vector Machine model with features extracted from a Poisson Moving Average predictor, as described in [Dalmazo et al., 2016a]. We have considered the DARPA dataset as the case study for evaluation.

5.4.1 Experimental Environment Setup

The experiments were conducted on a 64-bit standard personal computer with Intel Quad Core i5 Processor with 8Gb of RAM running at 2.70GHz. The operating system was Ubuntu 16.04 LTS. Furthermore, the programming language used for implementing the Triple-Similarity Mechanism was C++ (gcc 4.8.4 c11 version) and several shell scripts to deal with the data.

5.4.2 DARPA Dataset

For the evaluation of results, we resort to the DARPA Dataset. Further details about this dataset have been provided in Section 4.4.2. In the context of this evaluation, the goal is not evaluate the efficiency of the IDS but decreasing the amount of alarms generated by it. In light of this, we use the second week for the *training phase* because it contains details about the attacks such as timestamp, ports, duration and IP address. In order to make the dataset more realistic, we organized many of the attacks so that the resulting data set consisted of 10% attacks and 90% normal traffic.

The following list describes some attacks included in training data that have been posted on the Lincoln Laboratory web site. All the selected attacks are apt to generate anomalies in the network traffic.

- **Guest**: Try to guess a password via telnet for a guest account (Brute force attack, as described in Subsection 2.2.2.2)
- **Portsweep**: Surveillance sweep through many ports to determine which services are supported on a single host
- **Ipsweep**: Surveillance sweep performing either a port sweep or ping on multiple host addresses
- **Land**: Denial of Service where a remote host is sent a UDP packet with the same source and destination
- **Back**: Denial of Service attack against an apache webserver where a client requests a URL containing many backslashes
- **Syslog**: Denial of Service for the syslog service connects to port 514 with unresolvable source IP
- **Teardrop**: Denial of service where mis-fragmented UDP packets force a server to reboot

Currently, many providers have been suffering from Distributed DoS attacks, which, for instance, can cause many alarms to be generated [Nur and Tozal,

2018]. The next section presents the results from applying T-SyM in order to reduce the number of alarms generated.

5.4.3 Results

Table 5.3 shows a real Denial of Service attack against a web server. It is possible to observe that the attacker (172.16.114.50) floods the server (135.13.216.191) through port 80 trying to overthrow the services provided. According to the IDS, each line (1, 2, 3, 4, 5 and 7) generates an alarm. However, all the alarms constitute a single attack. This exemplifies a relevant problem for alarm management, the large number of alarms generated.

The evaluation is based on a comparison between the alarms generated for an IDS applied to the DARPA dataset. As general results, we observe that for all attacks fewer alarms were generated after using the *Triple-Similarity Mechanism*, as illustrated by Table 5.2. The Portsweep attack has shown the highest gain in comparison with other attacks (90%). In other words, the Portsweep attack triggered 574 alarms when using just the IDS. The number of alarms was about 10 times lower than expected when we combine the IDS and T-SyM. The DoS and Ipsweep attacks present similar results, 78.47% and 79.35% fewer alarms, respectively. Finally, our mechanism has decreased the number of alarms from 169 to 44 for the brute force attack.

Table 5.2: Comparison between alarms generated with and without the Similarity Mechanism

Type of attack	**Number of alarms**		**Improvement (%)**
	IDS	**IDS + T-SyM**	
Brute Force	169	44	73.96%
Portsweep	574	57	90.07%
Ipsweep	155	32	79.35%
DoS	641	138	78.47%

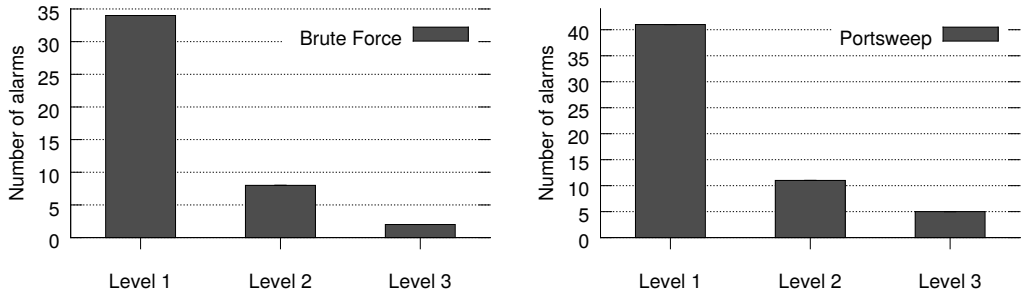
Along with the aggregation of true positive alarms reported above, our scheme is also able to aggregate false positives, therefore reducing their impact on the IDS performance. In particular, the IDS originally reported an amount of 2632 false alarms. After using T-SyM, by aggregating false alarms, their number decreased to 490, corresponding to an improvement of 81.4%.

The IDS is subject to failures. For example, a missing alarm could occur in the interval between two (or more) attacks that are performed at the same time. However, the problem of missing alarms is the exclusive responsibility of the IDS in question. Once the IDS generates alarms, they can be aggregated by the Triple-Similarity Mechanism. Therefore, the process of aggregation will never generate a missing alarm.

Table 5.3: Example of alarms in the data set

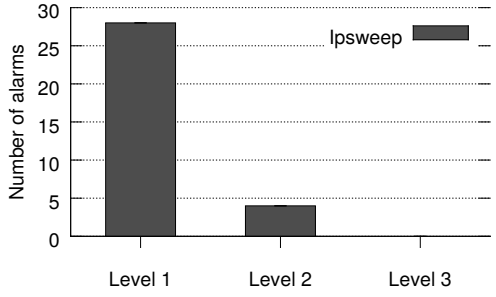
N	***Timestamp***	***Src IP***	***Src Port***	***Dst IP***	***Dst Port***	***Packet size***	***Number packets***	***Time interval***	**Δ-*variation***	***Attack type***	***Alarm***
1	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	124	back	1
2	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	103	back	1
3	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	61	back	1
4	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	59	back	1
5	920974021	172.16.114.50	80	135.13.216.191	29514	84	20	1	35	back	1
6	920974021	135.13.216.191	29514	172.16.114.50	80	32	59	1	34	-	0
7	920974021	172.16.114.50	80	135.13.216.191	29514	84	21	1	27	back	1

This case study applies Algorithm 5.1 to calculate the severity level for alarms. This algorithm resorts to a function ω for assessing the severity level of alarms based on a regression model fed with the history of attacks in the *Alarms Database*. We approximate the ω function to a logarithm base 4, meaning that an alarm at level 3 (the greatest risk for the system in this example) should aggregate, at least, 64 alarms from the IDS. Each level corresponds to a different class of alarm priority in increasing order of risk. In this case, actions can be generated and automatically executed in response to a specific alarm level. Figure 5.3 illustrates the severity level for the alarms. In general, most of the alarms are regarded as level 1 (about 66% of the alarms), 26% are level 2, whilst level 3 alarms represent 8% of the total.

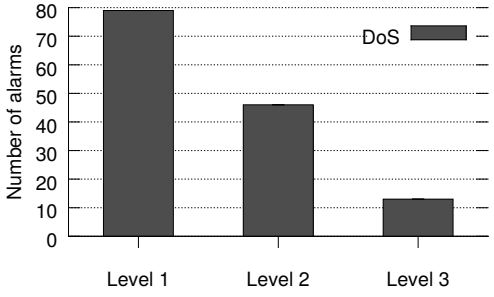


(a) Severity for Brute force attacks

(b) Severity for Portsweep attacks



(c) Severity for Ipsweep attacks



(d) Severity for DoS attacks

Figure 5.3: Evaluation of the severity for alarms

Moreover, the resulting dataset regarding the second week for the *training phase* has more than 400MB of data. After using our mechanism, the total amount of data was 78.3MB. In particular, our mechanism was able to decrease the number of alarms generated by aggregating similar alarms from the IDS. In addition, it is efficient when recognizing the attacks from less information.

It is worth noting that the T-SyM is not evaluating the effectiveness of the IDS to detect attacks nor the diversity of the labelled attacks contained within the DARPA dataset. Although the DARPA dataset was not designed for the cloud, it has a wide range of attacks that make it possible to detect threats from multiple hosts. In this context, using this dataset does not limit the proposal presented here. In fact, the DARPA dataset provides all the conditions necessary to validate our mechanism regarding distributed attacks, commonly present in

the cloud. Therefore, this scenario expresses a case study for a mechanism designed to aggregate alarms in a cloud environment.

Lastly, the T-SyM has proved to be efficient at aggregating alarms that carry similar features. In comparison with the alarms generated only by the IDS, our similarity approach produces fewer alarms but with higher levels of severity. This makes the network traffic monitoring of the cloud providers faster and more effective.

5.5 Summary of the Chapter

In this chapter, the main issues generated by Intrusion Detection Systems for cloud computing are presented. For instance, the huge number of alarms generated over time and how this impacts on the number of control messages between virtual machines and servers. To address these problems, the Triple-Similarity Mechanism (T-SyM), a systematic approach for aggregating similar alarms in the context of the cloud network traffic and an algorithm to assign severity level for alarms were proposed.

From the observation of the results, we can see that the mechanism was able to (*i*) reduce the generation of alarms by from 73% to 90% and; (*ii*) decrease the network data traffic to manage IDS and its associated transfer costs by more than 80%. Moreover, aggregating similar alarms produces fewer alarms but with higher levels of severity, supporting the network traffic monitoring of the cloud providers.

The outcomes of this chapter include the following submission:

- Dalmazo, Bruno L. and João P. Vilela and Marilia Curado, **"Triple-Similarity Mechanism for Alarm Management in the Cloud"**, Computers & Security, vol. 78, pp. 33-42, Elsevier, 2018. Impact factor: 2.65

Chapter 6

Conclusions and Future Work

> Truth is ever to be found in simplicity, and not in the multiplicity and confusion of things.
>
> *(Isaac Newton)*

Contents

This final chapter provides an overview around lack of security in cloud computing environment, the open issues that were addressed and the contributions to the advancement of the state of the art in this topic. Future research opportunities and directions are also outlined.

6.1 Synthesis of the Thesis

Cloud computing presents an impressive potential to provide rapid access to flexible and low cost IT resources on the fly, over the Internet. However, these benefits are subject to be harmed by the failure to guarantee an appropriate level of security when using cloud services, resulting in higher costs and potential loss of business.

Chapter 2 presented the general background required for understanding cloud computing and the solutions proposed in this thesis. More specifically, a description and characterization of the cloud computing environment was presented. Also, the characterization of network traffic in the cloud and the main issues that may harm its operation were introduced. Finally, an overview of IDSs focused on virtualized environments and similarity concepts in order to aggregate alarms were presented.

One of the most important instruments used for mitigating potential issues in the cloud is network traffic predictor. Taking this into consideration, Chapter

3 presented the state of the art followed by a taxonomy for network traffic prediction models. Moreover, an analysis mechanism that provided a standardized approach for evaluating network traffic predictors based on global and local data analysis was introduced. The outcomes of this mechanism enabled the performance comparison of several predictors in the cloud, particularly in terms of accuracy, historical dependency, time and computational overhead.

Chapter 4 discussed several IDSs designed for the cloud environment and an approach to detect anomalies in the cloud scenario was proposed. This approach differs from previous anomaly detection techniques since it relies on a mechanism that combines a Support Vector Machine model with features extracted from a Poisson Moving Average predictor. In doing this, the best level of detection rate and the second best number of false negative rates were achieved in comparison with other approaches in the literature.

Chapter 5 introduced the main issues caused by Intrusion Detection Systems for cloud computing, namely, the huge number of alarms produced over time and how this impacts the number of control messages between virtual machines and servers. In order to address these problems, the Triple-Similarity Mechanism was proposed, providing a systematic approach for aggregating similar alarms in the context of the cloud network traffic and an algorithm to assign severity level for alarms.

6.2 Contributions

The main contributions of this thesis are as follows. Firstly, a network traffic prediction model was proposed, that is suitable for the highly dynamic cloud computing environment. Secondly, an approach for extracting features based on the network traffic prediction model jointly with a Support Vector Machine in order to detect anomalies in the cloud network traffic. Finally, a similarity approach to aggregate alarms that may correspond to the same attack for minimizing generation of alarms, thus decreasing the network data traffic and its associated transfer costs.

More specifically, the following contributions can be identified:

Contribution 1

The Poisson Moving Average approach presented in section 3.2.5 was used to determine the probable minimum and maximum number of transactions that can occur within a given time period, from a series of discrete values. Besides providing the best prediction results with respect to the literature, the Poisson Moving Average has maintained the same computing complexity of the predictor models based on local analysis assessed in this work.

Contribution 2

The Dynamic Window Size Algorithm, introduced in section 3.3.3.2, depicted this contribution. In order to reduce the complexity of predicting

network traffic, time-bounded past information is considered by means of a sliding window of a size defined by the Dynamic Window Size Algorithm. All the predictors based on local analysis presented a considerable improvement after using this algorithm.

Contribution 3
The Feature Extraction Approach proposed in section 4.3.2 used multiple temporal layers of data for feature extraction so that it can express data in a compact representation by removing redundancy. It involves reducing the amount of information, therefore enabling its processing by the Support Vector Machine.

Contribution 4
The Anomaly Detection Mechanism, presented in section 4.3, described this contribution. The goal of this mechanism was to provide an efficient method to detect anomalies for the cloud-based network traffic. This mechanism combines a Support Vector Machine model with features extracted from a Poisson Moving Average predictor.

Contribution 5
The Triple-Similarity Mechanism, introduced in section 5.3, represents a systematic approach for aggregating similar alarms in the context of the cloud network traffic. From this mechanism was possible to reduce the generation of alarms, decreasing the network data traffic to manage IDS and its associated transfer costs.

Contribution 6
The Severity Adjustment of Alarms Algortihm, presented in section 5.3.4, has proved to be efficient for aggregating alarms inside a sliding window provided by the IDS. In comparison with alarms generated only by the IDS, after using this algorithm, fewer alarms with more severity were produced.

Next subsection proposes further research directions in the fields addressed in this thesis.

6.3 Future Work

Throughout this thesis, several approaches and mechanisms were proposed, showing improved results with respect to existing works in the literature. Nevertheless, there are still several aspects that need further work and could be addressed in the future including, but not limited to, evaluating the prediction models in other scenarios (with offline datasets or in real networks), extending the anomaly detection model so that it can cover other areas not initially envisaged (for instance, policies for reacting to an attack) and implementing these mechanisms in a real environment including the launching of real attacks against the network.

More specifically, approaches regarding prediction models can be enhanced by using artificial neural networks. For instance, an artificial neural network can

be fed with the data of the traffic network including the prediction. This kind of approach will strengthen the force exerted in the interaction between past and current values. In doing this, the prediction model could keep its generalization, but at the same time, become more suitable for a specific network baseline.

Additionally, the assessment of the current approaches for detecting anomalies in the cloud is limited to simulations and outdated offline datasets. In this context, experimental testbeds are welcome to support the new mechanisms evaluation in a more realistic scenario facing real attacks. In addition, the anomaly detection systems should consider an environment where they need to compete for resources against other applications. One of the main challenges is how to configure these mechanisms in an automated way to provide the best performance while not using unnecessary network resources.

Finally, the sudden increase of devices in the network presented by new technologies such as Internet of Things and 5G represents serious security and management challenges. In light of this, collaborative approaches will be prioritized in order to avoid the exhaustion of resources and degrade the performance of the whole system. Furthermore, techniques to minimize redundant information in order to meet sustainable computing with energy-aware are needed, as demonstrated in this thesis.

Bibliography

[Aceto et al., 2013] Aceto, G., Botta, A., de Donato, W., and Pescape, A. (2013). Cloud monitoring: A survey. *Computer Networks*, 57(9):2093 – 2115.

[Akesson and Toivonen, 2006] Akesson, B. M. and Toivonen, H. T. (2006). A neural network model predictive controller. *Journal of Process Control*, 16(9):937 – 946.

[Ali et al., 2015] Ali, M., Khan, S. U., and Vasilakos, A. V. (2015). Security in cloud computing: Opportunities and challenges. *Information Sciences*, 305:357–383.

[Alshaer, 2015] Alshaer, H. (2015). An overview of network virtualizationan overview of network virtualization and cloud network as a service. *International Journal of Network Management*, 25(1):1–30.

[Armbrust et al., 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

[Arshad et al., 2013] Arshad, J., Townend, P., and Xu, J. (2013). A novel intrusion severity analysis approach for clouds. *Future Generation Computer Systems*, 29(1):416–428. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[Auld et al., 2007] Auld, T., Moore, A., and Gull, S. (2007). Bayesian neural networks for internet traffic classification. *Neural Networks, IEEE Transactions on*, 18(1):223–239.

[Ballani et al., 2011] Ballani, H., Costa, P., Karagiannis, T., and Rowstron, A. (2011). Towards predictable datacenter networks. In *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM'11)*, volume 41, pages 242–253.

[Barford et al., 2002] Barford, P., Kline, J., Plonka, D., and Ron, A. (2002). A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, IMW '02, pages 71–82, New York, NY, USA. ACM.

[Baumast, 2013] Baumast, A. (2013). *Carbon Disclosure Project. Encyclopedia of corporate social responsibility*, volume 21. Springer Berlin Heidelberg.

[Benferhat et al., 2013] Benferhat, S., Boudjelida, A., Tabia, K., and Drias, H. (2013). An intrusion detection and alert correlation approach based on re-

vising probabilistic classifiers using expert knowledge. *Applied Intelligence*, 38(4):520–540.

[Beng et al., 2014] Beng, L. Y., Ramadass, S., Manickam, S., and Fun, T. S. (2014). A survey of intrusion alert correlation and its design considerations. *IETE Technical Review*, 31(3):233–240.

[Benson et al., 2010] Benson, T., Akella, A., and Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 267–280, New York, NY, USA. ACM.

[Bermolen and Rossi, 2009] Bermolen, P. and Rossi, D. (2009). Support vector regression for link load prediction. *Computer Networks*, 53(2):191 – 201. QoS Aspects in Next-Generation Networks.

[Buttle, 2015] Buttle, F. (2015). *Customer Relationship Management: Concepts and Technologies*. Abingdon, Oxon, New York, New York: Routledge, 3 edition.

[Buyya et al., 2010] Buyya, R., Broberg, J., and Goscinski, A. M. (2010). *Cloud computing: Principles and paradigms*, volume 87. John Wiley & Sons.

[Center, 1998] Center, C. C. (1998). IP Denial-of-Service attacks. *CERT Advisory CA-1997-28, Computer Emergency Response Team*.

[Cha, 2007] Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical models and Methods in Applied Sciences*, 1(4):300–307.

[Chang and Tsai, 2009] Chang, B. and Tsai, H. (2009). Improving network traffic analysis by foreseeing data-packet-flow with hybrid fuzzy-based model prediction. *Expert Systems with Applications. Tarrytown, NY, USA*, 36(3):6960–6965.

[Chatfield and Yar, 1988] Chatfield, C. and Yar, M. (1988). Holt-winters forecasting: some practical issues. *The Statistician*, pages 129–140.

[Chatterjee and Hadi, 2015] Chatterjee, S. and Hadi, A. S. (2015). *Regression analysis by example*. John Wiley & Sons.

[Chen et al., 2005] Chen, W.-H., Hsu, S.-H., and Shen, H.-P. (2005). Application of SVM and ANN for intrusion detection. *Computers & Operations Research*, 32(10):2617–2634.

[Chen et al., 2012] Chen, Y., Yang, B., and Meng, Q. (2012). Small-time scale network traffic prediction based on flexible neural tree. *Applied Soft Computing*, 12(1):274 – 279.

[Chopra et al., 2005] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1.

[Chunlin and Layuan, 2014] Chunlin, L. and Layuan, L. (2014). Multi-layer resource management in cloud computing. *Journal of network and systems management*, 22(1):100–120.

[Cilibrasi and Vitanyi, 2007] Cilibrasi, R. L. and Vitanyi, P. M. B. (2007). The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383.

[Cormen et al., 2001] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., et al. (2001). *Introduction to algorithms*, volume 2. MIT press Cambridge.

[Cortez et al., 2006] Cortez, P., Rio, M., Rocha, M., and Sousa, P. (2006). Internet traffic forecasting using neural networks. In *International Joint Conference on Neural Networks, 2006. IJCNN '06.*, pages 2635–2642.

[Cuppens and Miège, 2002] Cuppens, F. and Miège, A. (2002). Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy, 2002.*, pages 202–215. IEEE.

[Dainotti et al., 2008] Dainotti, A., De Donato, W., Pescape, A., and Salvo Rossi, P. (2008). Classification of network traffic via packet-level hidden markov models. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. New Orleans, Louisiana*, pages 1–5.

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

[Dalmazo et al., 2011] Dalmazo, B., Cordeiro, W., Rabelo, L., Wickboldt, J., Lunardi, R., dos Santos, R., Gaspary, L., Granville, L., Bartolini, C., and Hickey, M. (2011). Leveraging it project lifecycle data to predict support costs. In *IFIP/IEEE International Symposium on Integrated Network Management (IM'2011)*, pages 249–256.

[Dalmazo et al., 2013] Dalmazo, B. L., Vilela, J. P., and Curado, M. (2013). Predicting traffic in the cloud: A statistical approach. In *Third International Conference on Cloud and Green Computing (CGC'13), 2013*, pages 121–126.

[Dalmazo et al., 2014] Dalmazo, B. L., Vilela, J. P., and Curado, M. (2014). Online traffic prediction in the cloud: A dynamic window approach. In *The 2nd International Conference on Future Internet of Things and Cloud (FiCloud'2014)*, pages 9–14.

[Dalmazo et al., 2015] Dalmazo, B. L., Vilela, J. P., and Curado, M. (2015). A svm model based on network traffic prediction for detecting anomalies. In *21th edition of the Portuguese Conference on Pattern Recognition*.

[Dalmazo et al., 2016a] Dalmazo, B. L., Vilela, J. P., and Curado, M. (2016a). Online traffic prediction in the cloud. *International Journal of Network Management*, 26(4):269–285.

[Dalmazo et al., 2017a] Dalmazo, B. L., Vilela, J. P., and Curado, M. (2017a). Performance analysis of network traffic predictors in the cloud. *Journal of Network and Systems Management*, 25(2):290–320.

[Dalmazo et al., 2017b] Dalmazo, B. L., Vilela, J. P., and Curado, M. (2017b). Security and trustworthiness in cloud computing. In *Meeting with Science and Technology in Portugal*.

[Dalmazo et al., 2018] Dalmazo, B. L., Vilela, J. P., and Curado, M. (2018). Triple-similarity mechanism for alarm management in the cloud. *Computers & Security - Elsevier*, 78:33–42.

[Dalmazo et al., 2016b] Dalmazo, B. L., Vilela, J. P., Simoes, P., and Curado, M. (2016b). Expedite feature extraction for enhanced cloud anomaly detection. In *IEEE/IFIP Network Operations and Management Symposium (NOMS'16)*, pages 1215–1220.

[Debar et al., 1999] Debar, H., Dacier, M., and Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822.

[Deng et al., 2012] Deng, N., Tian, Y., and Zhang, C. (2012). *Support vector machines: optimization based theory, algorithms, and extensions*. CRC Press.

[Dhage and Meshram, 2012] Dhage, S. N. and Meshram, B. (2012). Intrusion detection system in cloud computing environment. *International Journal of Cloud Computing*, 1(2):261–282.

[Di Pietro and Mancini, 2008] Di Pietro, R. and Mancini, L. V. (2008). *Intrusion detection systems*, volume 38. Springer Science & Business Media.

[dos Santos et al., 2013] dos Santos, R. L., Wickboldt, J. A., Dalmazo, B. L., Granville, L. Z., Gaspary, L. P., and Lunardi, R. C. (2013). Identifying the root cause of failures in it changes: Novel strategies and trade-offs. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 118–125.

[Drago et al., 2012] Drago, I., Mellia, M., Munafò, M. M., Sperotto, A., Sadre, R., and Pras, A. (2012). Inside Dropbox: Understanding Personal Cloud Storage Services. In *Proceedings of the 12th ACM SIGCOMM Conference on Internet Measurement. Berlin, Germany.*, IMC'12, pages 481–494.

[Elshoush and Osman, 2011] Elshoush, H. T. and Osman, I. M. (2011). Alert correlation in collaborative intelligent intrusion detection systems - a survey. *Applied Soft Computing*, 11(7):4349 – 4365. Soft Computing for Information System Security.

[Elshoush and Osman, 2012] Elshoush, H. T. and Osman, I. M. (2012). An improved framework for intrusion alert correlation. In *Proceedings of the World Congress on Engineering*, volume 1, pages 1–6.

[Erman et al., 2006] Erman, J., Mahanti, A., and Arlitt, M. (2006). Qrp05-4: Internet traffic identification using machine learning. In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pages 1–6.

[Estevez-Tapiador et al., 2004] Estevez-Tapiador, J. M., Garcia-Teodoro, P., and Diaz-Verdejo, J. E. (2004). Anomaly detection methods in wired net-

works: a survey and taxonomy. *Computer Communications*, 27(16):1569 – 1584.

[Feng and Shu, 2005] Feng, H. and Shu, Y. (2005). Study on network traffic prediction techniques. In *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing*, volume 2, pages 1041–1044. IEEE.

[Fox et al., 2009] Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., and Stoica, I. (2009). Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 28(13):25.

[Freedman, 2009] Freedman, D. A. (2009). *Statistical models: theory and practice*. cambridge university press.

[Fu, 2011] Fu, S. (2011). Performance metric selection for autonomic anomaly detection on cloud computing systems. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5.

[Ganeshkumar and Pandeeswari, 2015] Ganeshkumar, P. and Pandeeswari, N. (2015). Adaptive neuro-fuzzy-based anomaly detection system in cloud. *International Journal of Fuzzy Systems*, pages 1–12.

[Gardiner et al., 1985] Gardiner, C. W. et al. (1985). *Handbook of stochastic methods*, volume 3. Springer Berlin.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability*, volume 174. Freeman San Francisco.

[Gilmer, 2011] Gilmer, E. M. (2011). Is There a Silver Lining for the Environment in Cloud Computing? *The New York Times*, 10 August.

[Haines et al., 2001] Haines, J., Rossey, L., Lippmann, R., and Cunningham, R. (2001). Extending the darpa off-line intrusion detection evaluations. In *DARPA Information Survivability Conference & Exposition II (DISCEX'01)*, volume 1, pages 35–45.

[Hajji, 2005] Hajji, H. (2005). Statistical analysis of network traffic for adaptive faults detection. *IEEE Transactions on Neural Networks*, 16(5):1053–1063.

[Hick et al., 2007] Hick, P., Aben, E., Claffy, K., and Polterock, J. (2007). The CAIDA DDoS Attack 2007 Dataset.

[Hongying and Li, 2013] Hongying, J. and Li, L. (2013). Dynamic network traffic flow prediction model based on modified quantum-behaved particle swarm optimization. *Journal of Networks*, 8(10):2332–2339.

[Hoque et al., 2014] Hoque, N., Bhuyan, M. H., Baishya, R., Bhattacharyya, D., and Kalita, J. (2014). Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications*, 40(0):307–324.

[Horng et al., 2011] Horng, S.-J., Su, M.-Y., Chen, Y.-H., Kao, T.-W., Chen, R.-J., Lai, J.-L., and Perkasa, C. D. (2011). A novel intrusion detection

system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, 38(1):306 – 313.

[Hubballi and Suryanarayanan, 2014] Hubballi, N. and Suryanarayanan, V. (2014). False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*, 49:1–17.

[Hudic et al., 2017] Hudic, A., Smith, P., and Weippl, E. R. (2017). Security assurance assessment methodology for hybrid clouds. *Computers & Security*, 70(Supplement C):723 – 743.

[Hwang et al., 2009] Hwang, K., Kulkareni, S., and Hu, Y. (2009). Cloud security with virtualized defense and reputation-based trust mangement. In *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009. DASC '09.*, pages 717–722.

[Hyndman and Khandakar, 2008] Hyndman, R. J. and Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 26(3):1–22.

[Jennings and Stadler, 2015] Jennings, B. and Stadler, R. (2015). Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, 23(3):567–619.

[Jensen et al., 2009] Jensen, M., Schwenk, J., Gruschka, N., and Iacono, L. (2009). On technical security issues in cloud computing. In *IEEE International Conference on Cloud Computing, 2009. CLOUD '09.*, pages 109–116.

[Jin, 2005] Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12.

[Joshi et al., 2012] Joshi, B., Vijayan, A., and Joshi, B. (2012). Securing cloud computing environment against ddos attacks. In *International Conference on Computer Communication and Informatics (ICCCI).*, pages 1–5.

[Joshi and Hadi, 2015] Joshi, M. and Hadi, T. H. (2015). A review of network traffic analysis and prediction techniques. *arXiv preprint arXiv:1507.05722*.

[Kandukuri et al., 2009] Kandukuri, B., Paturi, V., and Rakshit, A. (2009). Cloud security issues. In *IEEE International Conference on Services Computing, 2009. SCC '09.*, pages 517–520.

[Khalimonenko et al., 2017] Khalimonenko, A., Kupreev, O., and Ilganaev, K. (2017). DDoS attacks in Q3 2017. Securelist.com. {Online resource} Available at: https://securelist.com/ddos-attacks-in-q3-2017/83041/. [Accessed 20/03/18].

[Kholidy and Baiardi, 2012] Kholidy, H. and Baiardi, F. (2012). CIDS: A framework for intrusion detection in cloud systems. In *Ninth International Conference on Information Technology: New Generations (ITNG), 2012*, pages 379–385.

[Kind et al., 2009] Kind, A., Stoecklin, M., and Dimitropoulos, X. (2009).

Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management*, 6(2):110–121.

[Kissel, 2011] Kissel, R. (2011). *Glossary of key information security terms*. DIANE Publishing.

[Klinker, 2011] Klinker, F. (2011). Exponential moving average versus moving exponential average. *Mathematische Semesterberichte*, 58(1):97–107.

[Kotz and Essien, 2005] Kotz, D. and Essien, K. (2005). Analysis of a campus-wide wireless network. *Wireless Networks*, 11(1-2):115–133.

[Krunz and Makowski, 1998] Krunz, M. and Makowski, A. (1998). Modeling video traffic using m/g/ infin; input processes: a compromise between markovian and lrd models. *IEEE Journal on Selected Areas in Communications*, 16(5):733–748.

[Kuo and Wu, 2011] Kuo, W.-K. and Wu, K.-W. (2011). Traffic prediction and QoS transmission of real-time live VBR videos in WLANs. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 7(4):36.

[Kwon et al., 2011] Kwon, H., Kim, T., Yu, S. J., and Kim, H. K. (2011). Self-similarity based lightweight intrusion detection method for cloud computing. In *Asian Conference on Intelligent Information and Database Systems*, volume 6592, pages 353–362. Springer Berlin Heidelberg.

[Lee et al., 2011] Lee, J.-H., Park, M.-W., Eom, J.-H., and Chung, T.-M. (2011). Multi-level intrusion detection system and log management in cloud computing. In *13th International Conference on Advanced Communication Technology (ICACT), 2011*, pages 552–555.

[Lee et al., 2012] Lee, W., Chen, C., Chen, K., Chen, T., and Liu, C. (2012). A comparative study on the forecast of fresh food sales using logistic regression, moving average and bpnn methods. *Journal of Marine Science and Technology*, 20(2):142–152.

[Leydesdorff, 2008] Leydesdorff, L. (2008). On the normalization and visualization of author co-citation data: Salton's cosine versus the jaccard index. *Journal of the American Society for Information Science and Technology*, 59(1):77–85.

[Li et al., 2012] Li, A., Han, Y., Zhou, B., Han, W., and Jia, Y. (2012). Detecting Hidden Anomalies Using Sketch for High-speed Network Data Stream Monitoring. *Applied Mathematics*, 6(3):759–765.

[Li and Liu, 2010] Li, H. and Liu, D. (2010). Research on intelligent intrusion prevention system based on snort. In *International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010*, volume 1, pages 251–253.

[Li et al., 2009] Li, K., Zhou, W., Li, P., Hai, J., and Liu, J. (2009). Distin-

guishing ddos attacks from flash crowds using probability metrics. In *Third International Conference on Network and System Security, 2009*, pages 9–17.

[Li and Lim, 2008] Li, M. and Lim, S. (2008). Modeling network traffic using generalized cauchy process. *Physica A: Statistical Mechanics and its Applications*, 387(11):2584–2594.

[Li and Moore, 2007] Li, W. and Moore, A. W. (2007). A machine learning approach for efficient traffic classification. In *15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 310–317.

[Li et al., 2010] Li, Z., Xia, G., Gao, H., Tang, Y., Chen, Y., Liu, B., Jiang, J., and Lv, Y. (2010). Netshield: Massive semantics-based vulnerability signature matching for high-speed networks. *SIGCOMM Computer Communication Review*, 40(4):279–290.

[Lim et al., 2000] Lim, T.-S., Loh, W.-Y., and Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228.

[Liu et al., 2012] Liu, Y., Tseng, K.-K., and Pan, J.-S. (2012). Statistical based waveform classification for cloud intrusion detection. In *2012 International Conference on Computing, Measurement, Control and Sensor Network (CMCSN)*, pages 225–228.

[Lo et al., 2010] Lo, C.-C., Huang, C.-C., and Ku, J. (2010). A cooperative intrusion detection system framework for cloud computing networks. In *39th International Conference on Parallel Processing Workshops (ICPPW'10)*, pages 280–284.

[Loiseau et al., 2010] Loiseau, P., Gonçalves, P., Dewaele, G., Borgnat, P., Abry, P., and Primet, P. V.-B. (2010). Investigating self-similarity and heavy-tailed distributions on a large-scale experimental facility. *IEEE/ACM Transactions on Networking (TON)*, 18(4):1261–1274.

[Lu et al., 2014] Lu, X., Yu, Z., Guo, B., and Zhou, X. (2014). Predicting the content dissemination trends by repost behavior modeling in mobile social networks. *Journal of Network and Computer Applications*, 42(0):197 – 207.

[Makridakis et al., 2008] Makridakis, S., Wheelwright, S. C., and Hyndman, R. J. (2008). *Forecasting methods and applications*. John Wiley & Sons.

[Mallissery et al., 2011] Mallissery, S., Prabhu, J., and Ganiga, R. (2011). Survey on intrusion detection methods. In *3rd International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2011)*, pages 224–228. IET.

[McLeod and Zhang, 2008] McLeod, A. and Zhang, Y. (2008). Faster {ARMA} maximum likelihood estimation. *Computational Statistics & Data Analysis*, 52(4):2166–2176.

[Mell and Grance, 2011] Mell, P. and Grance, T. (2011). The NIST definition of cloud computing. *National Institute of Standards and Technology*.

[Modi et al., 2013] Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., and Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1):42–57.

[Mohler and Mihalcea, 2009] Mohler, M. and Mihalcea, R. (2009). Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 567–575. Association for Computational Linguistics.

[Monahan, 2005] Monahan, J. F. (2005). Some algorithms for the conditional mean vector and covariance matrix. *Journal of Statistical Software*, 16(i08).

[Mulay et al., 2010] Mulay, S. A., Devale, P., and Garje, G. (2010). Intrusion detection system using support vector machine and decision tree. *International Journal of Computer Applications*, 3(3):40–43.

[Nefoussi et al., 2014] Nefoussi, J. J., Dalmazo, B. L., and Lunardi, R. (2014). Pysoneta um sistema baseado em python para visualizacao de dados do trafego de rede. In *12th edition Escola Regional de Redes de Computadores, 2014*.

[Networking, CISCO Global Cloud Index, 2018] Networking, CISCO Global Cloud Index (2018). Cisco global cloud index: Forecast and methodology, 2016-2021 white paper.

[Nguyen and Armitage, 2008] Nguyen, T. and Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *Communications Surveys Tutorials, IEEE*, 10(4):56–76.

[Nur and Tozal, 2018] Nur, A. Y. and Tozal, M. E. (2018). Record route ip traceback: Combating dos attacks and the variants. *Computers & Security*, 72(Supplement C):13 – 25.

[Owezarski et al., 2013] Owezarski, P., Lobo, J., and Medhi, D. (2013). Network and service management for cloud computing and data centers: a report on cnsm 2012. *Journal of network and systems management*, 21(4):707–712.

[Palmieri et al., 2013] Palmieri, F., Fiore, U., and Castiglione, A. (2013). A distributed approach to network anomaly detection based on independent component analysis. *Concurrency and Computation: Practice and Experience*.

[Papadopouli et al., 2006] Papadopouli, M., Raftopoulos, E., and Shen, H. (2006). Evaluation of short-term traffic forecasting algorithms in wireless networks. In *2nd Conference on Next Generation Internet Design and Engineering (NGI'06)*, pages 8–109. IEEE.

[Parikh and Chen, 2008] Parikh, D. and Chen, T. (2008). Data fusion and cost minimization for intrusion detection. *IEEE Transactions on Information Forensics and Security*, 3(3):381–389.

[Patel et al., 2013] Patel, A., Taghavi, M., Bakhtiyari, K., and Junior, J. C. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. *Journal of Network and Computer Applications*, 36(1):25–41.

[Plonka and Barford, 2009] Plonka, D. and Barford, P. (2009). Network anomaly confirmation, diagnosis and remediation. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 128–135.

[Prangchumpol, 2013] Prangchumpol, D. (2013). A network traffic prediction algorithm based on data mining technique. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 79, page 1196. World Academy of Science, Engineering and Technology (WASET).

[Rahmani et al., 2009] Rahmani, M., Tappayuthpijarn, K., Krebs, B., Steinbach, E., and Bogenberger, R. (Nov. 2009). Traffic shaping for resource-efficient in-vehicle communication. *IEEE Transactions on Industrial Informatics*, 5(4):414–428.

[Rittinghouse and Ransome, 2016] Rittinghouse, J. W. and Ransome, J. F. (2016). *Cloud computing: implementation, management, and security*. CRC press.

[Sabahi and Movaghar, 2008] Sabahi, F. and Movaghar, A. (2008). Intrusion detection: A survey. In *3rd International Conference on Systems and Networks Communications (ICSNC'08)*, pages 23–26. IEEE.

[Salah et al., 2016] Salah, K., Elbadawi, K., and Boutaba, R. (2016). An analytical model for estimating cloud resources of elastic services. *Journal of Network and Systems Management*, 24(2):285–308.

[Sang and Li, 2002] Sang, A. and Li, S.-q. (2002). A predictability analysis of network traffic. *Computer Networks*, 39(4):329–345.

[Shirey, 2000] Shirey, R. (2000). RFC 2828: Internet security glossary. *The Internet Society*.

[Shon and Moon, 2007] Shon, T. and Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799 – 3821.

[Sobh, 2006] Sobh, T. (2006). Wired and wireless intrusion detection system: Classifications, good characteristics and state-of-the-art. *Computer Standards Interfaces*, 28(6):670–694.

[Song and Li, 2008] Song, H. and Li, G. (2008). Tourism demand modelling and forecasting—a review of recent research. *Tourism Management*, 29(2):203–220.

[Sperotto et al., 2010] Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., and Stiller, B. (2010). An overview of ip flow-based intrusion detection. *Communications Surveys Tutorials, IEEE*, 12(3):343–356.

[Subashini and Kavitha, 2011] Subashini, S. and Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1–11.

[Team, 2012] Team, R. C. (2012). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, 2012.

[Torres et al., 2005] Torres, J. L., Garcia, A., De Blas, M., and De Francisco, A. (2005). Forecast of hourly average wind speed with {ARMA} models in navarre (spain). *Solar Energy*, 79(1):65–77.

[Vieira et al., 2010] Vieira, K., Schulter, A., Westphall, C., and Westphall, C. (2010). Intrusion detection for grid and cloud computing. *IT Professional*, 12(4):38–43.

[Vokorokos and Balaz, 2010] Vokorokos, L. and Balaz, A. (2010). Host-based intrusion detection system. In *14th International Conference on Intelligent Engineering Systems (INES)*., pages 43–47. IEEE.

[Wang et al., 2010] Wang, C., Talwar, V., Schwan, K., and Ranganathan, P. (2010). Online detection of utility cloud anomalies using metric distributions. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 96–103.

[Wei, 1994] Wei, W. W.-S. (1994). *Time series analysis*. Addison-Wesley publ.

[Weigend and Gershenfeld, 1994] Weigend, A. S. and Gershenfeld, N. A., editors (1994). *Time series prediction: Forecasting the future and understanding the past*. Westview Press.

[Weingartner et al., 2015] Weingartner, R., Brascher, G. B., and Westphall, C. B. (2015). Cloud resource management: A survey on forecasting and profiling models. *Journal of Network and Computer Applications*, 47(0):99 – 106.

[Whaiduzzaman et al., 2014] Whaiduzzaman, M., Sookhak, M., Gani, A., and Buyya, R. (2014). A survey on vehicular cloud computing. *Journal of Network and Computer Applications*, 40(0):325 – 344.

[Wilamowski, 2009] Wilamowski, B. (2009). Neural network architectures and learning algorithms. *Industrial Electronics Magazine, IEEE*, 3(4):56–63.

[Xiong et al., 2014] Xiong, W., Hu, H., Xiong, N., Yang, L. T., Peng, W.-C., Wang, X., and Qu, Y. (2014). Anomaly secure detection methods by analyzing dynamic characteristics of the network traffic in cloud communications. *Information Sciences*, 258:403–415.

[Yadav and Balakrishnan, 2014] Yadav, R. K. and Balakrishnan, M. (2014). Comparative evaluation of arima and anfis for modeling of wireless network traffic time series. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):15.

[Yan et al., 2016] Yan, Q., Yu, F. R., Gong, Q., and Li, J. (2016). Software-defined networking (sdn) and distributed denial of service (ddos) attacks in

cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys Tutorials*, 18(1):602–622.

[Yan-hui and Tao, hina] Yan-hui, P. and Tao, W. (2009. Jiangsu, China). Network traffic emulation based on representative network behavior and protocol. In *1st International Conference on Information Science and Engineering (ICISE)*, pages 1777–1780. IEEE.

[Yin et al., 2005] Yin, H., Lin, C., Sebastien, B., Li, B., and Min, G. (2005). Network traffic prediction based on a new time series model. *International Journal of Communication Systems*, 18(8):711–729.

[Zare Moayedi and Masnadi-Shirazi, 2008] Zare Moayedi, H. and Masnadi-Shirazi, M. (2008). Arima model for network traffic prediction and anomaly detection. In *International Symposium on Information Technology, 2008. ITSim 2008.*, volume 4, pages 1–6.

[Zhang and Qi, 2005] Zhang, G. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, 160(2):501 – 514. Decision Support Systems in the Internet Age.

[Zhang et al., 2013] Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., and Guan, Y. (2013). Network traffic classification using correlation information. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):104–117.

[Zhang et al., 2010] Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.

[Zhani et al., 2009] Zhani, M. F., Elbiaze, H., and Kamoun, F. (2009). Analysis and prediction of real network traffic. *Journal of Networks*, 4(9):855–865.

[Zhao, 2009] Zhao, H. (2009). Multiscale analysis and prediction of network traffic. In *IEEE 28th International on Performance Computing and Communications Conference (IPCCC).*, pages 388–393. IEEE.

[Zhou et al., 2006] Zhou, B., He, D., and Sun, Z. (2006). Traffic modeling and prediction using arima/garch model. In *Modeling and Simulation Tools for Emerging Telecommunication Networks*, pages 101–121. Springer.

[Zhu and Sastry, 2010] Zhu, B. and Sastry, S. (2010). Scada-specific intrusion detection/prevention systems: a survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems, Stockholm, Sweden*.

[Zissis and Lekkas, 2012] Zissis, D. and Lekkas, D. (2012). Addressing cloud computing security issues. *Future Generation Computer Systems*, 28(3):583 – 592.

Appendices

A Proof for Proposition 1

If we know the minimum and maximum range, *e.g.* from y_a to y_b, we are able to represent its average m by:

$$m = \frac{q_a y_a + q_b y_b}{q_a + q_b}, \tag{A.1}$$

where q_a and q_b are the quantity of y_a and y_b, respectively. Then, if we consider the average and these extreme values as referred before to estimate the maximum variance σ^2_{max} into a sliding window, it may be expressed for:

$$\sigma^2_{max} = \frac{q_a(m - y_a)^2 + q_b(y_b - m)^2}{q_a + q_b},$$

$$\sigma^2_{max} = \frac{q_a(m^2 - 2my_a + y_a^2) + q_b(m^2 - 2my_b + y_b^2)}{q_a + q_b},$$

$$\sigma^2_{max} = \frac{(q_a + q_b)m^2}{q_a + q_b} - \frac{2(q_a y_a + q_b y_b)m}{q_a + q_b} + \frac{q_a y_a^2 + q_b y_b^2}{q_a + q_b}. \tag{A.2}$$

Simplifying the first term in Equation A.2 and substituting the second term by Equation A.1 into it, we achieve:

$$\sigma^2_{max} = m^2 - 2m^2 + \frac{q_a y_a^2 + q_b y_b^2}{q_a + q_b}. \tag{A.3}$$

Now, isolating the term $q_a y_a$ from the Equation A.1 we have:

$$q_a y_a = m(q_a + q_b) - q_b y_b. \tag{A.4}$$

And similarly:

$$q_b y_b = m(q_a + q_b) - q_a y_a. \tag{A.5}$$

Using these two equations (A.4 and A.5) into the Equation A.3, we have:

$$\sigma^2_{max} = -m^2 + \frac{y_a(m(q_a + q_b) - q_b y_b) + y_b(m(q_a + q_b) - q_a y_a)}{q_a + q_b}.$$

Evidencing the term $q_a + q_b$ of the equation,

$$\sigma^2_{max} = -m^2 + \frac{m(q_a + q_b)(y_a + y_b) - (q_a + q_b)(y_a y_b)}{q_a + q_b},$$

$$\sigma^2_{max} = -m^2 + m(y_a + y_b) - y_a y_b.$$

Evidencing the term $y_b - m$,

$$\sigma^2_{max} = m(y_b - m) - y_a(y_b - m). \tag{A.6}$$

So, we may represent the σ^2_{max} just acknowledging the minimum, the maximum and the average of the data inside the sliding window. In addition, the Equation A.6 is equivalent to the Equation 3.9. This finally leads to the results presented in Proposition 1.