UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Master Thesis

Network traffic footprint analysis in the RedIRIS academic network

Author:
Felipe Mata Marcos
Telecommunication Engineer

Supervisor:
Prof. Javier Aracil Rico

Madrid, 2009

MASTER THESIS:	Network traffic footprint analysis in the RedIRIS academic network
AUTHOR:	Felipe Mata Marcos
SUPERVISOR:	Prof. Javier Aracil Rico

The committee for the defense of this master thesis is composed by:

PRESIDENT:	Prof. Javier Aracil
VOCALS:	Dr. Jorge López de Vergara
	Dr. José Alberto Hernández

Contents

List of Figures

List of Tables

Summary

This master thesis covers the first two years of the researching work with network measurements of the author. This researching has focused on the search of network invariants in such measurements, hoping that these invariants may help to obtain relevant information about the network status. The organization of this document faithfully follows the researching temporal line during these two years.

In the first part, the analysis with packet traces is described. Such packet traces were downloaded from public repositories like CAIDA, and were processed at the flow level, obtaining the more relevant statistics at packet and connection level. These statistics were then applied to traffic classification by means of clustering techniques. The target of such classification was to determine the applications that produce the traffic, so the quality of service requirements for each kind of traffic could be assessed. Another study using the same dataset was the analysis of the throughput of the flows. We carry-out an study of the distribution of the mean value of the throughput, and statistical techniques to compute its confidence intervals were applied. Our hypothesis supposes that, under optimal working conditions, these confidence intervals must overlap amongst them because the majority of the flows would reach a similar value for the mean throughput.

However, we observed that such confidence intervals do not overlap, but instead they followed a daily pattern opposite to the well-known traffic daily pattern. This motivated us to analyze the daily traffic pattern, using real measurements from the Spanish National Research and Education Network RedIRIS. The main finding of this study was that the RedIRIS daily traffic pattern was close to other ones published from other networks, but only for working days. Therefore, we use the invariance of the daily traffic pattern of the RedIRIS network to design a multivariate normal model for the measurements over one day. Such model was validated by means of normality tests, showing very good results.

Finally, we applied this model to develop an online algorithm for automatically detecting change points in the links' load. We first assessed the performance of the algorithm with synthetically generated datasets, and then we applied our algorithm to the RedIRIS network measurements. The results show that our algorithm can be useful to reduce the operational expenditures of a network operator, given that the output of our algorithm (in the form of alerts when a change is detected) can prevent the network manager to visually inspect all the time series generated through the network monitors of the operator.

Resumen

La presente tesis de máster comprende el trabajo de dos años de investigación con medidas de red. Esta investigación se ha enfocado a la búsqueda de invariantes en dichas medidas, que permitan obtener información relevante del estado de la red. La organización de este documento sigue fielmente la línea temporal de la investigación realizada.

En la primera parte, se describe el análisis de trazas de paquetes, descargadas de repositorios públicos como CAIDA, las cuales fueron procesadas a nivel de flujo, obteniéndose las estadísticas más relevantes a nivel de paquete y conexión. Estas estadísticas fueron utilizadas para realizar clasificación de tráfico, mediante la aplicación de técnicas de agrupamiento (*clustering* en inglés). El objetivo de esta clasificación era determinar las aplicaciones que producían el tráfico, de manera que se pudieran aplicar medidas de calidad de servicio específicas a cada tipo de tráfico para determinar el estado del enlace. Otra opción explorada partiendo del mismo conjunto de datos, fue el análisis de la tasa de transmisión de los flujos presentes en el enlace analizado. Sobre esta tasa se estudió la distribución de su valor medio, aplicándose técnicas estadísticas para determinar intervalos de confianza alrededor de ese valor medio. Nuestra hipótesis suponía que, bajo condiciones óptimas de funcionamiento, los intervalos de confianza debían solaparse, ya que la mayoría de los flujos obtendrían un valor medio de la tasa similar.

Sin embargo, observamos que los intervalos de confianza seguían un patrón de comportamiento diario opuesto al patrón observado en el tráfico de red. Esto nos llevó a analizar el patrón diario del tráfico, tomando como datos medidas de la red académica española RedIRIS. El principal hallazgo de este estudio fue que el patrón diario de tráfico de RedIRIS es similar a otros publicados sobre otras redes, pero solo para los días laborables. Por tanto, aprovechamos la invariancia del patrón de tráfico de RedIRIS para diseñar un modelo normal multivariante para las medidas de un día. Este modelo ha sido validado con tests de normalidad, mostrando buenos resultados.

Finalmente, hemos aplicado este modelo para diseñar un algoritmo para la detección automática de cambios en la carga de un enlace, cuyo rendimiento ha sido evaluado con datos sintéticamente generados. Una vez comprobado su validez, se ha aplicado el algoritmo a medidas reales de la red RedIRIS, graficándose los resultados obtenidos. Estos resultados demuestran que nuestro algoritmo puede ser útil para reducir costes de operación de un operador de red, puesto que el uso de nuestro algoritmo evita la necesidad de monitorizar continuamente los resultados de las medidas realizadas, siendo únicamente necesario supervisar los enlaces en caso de que el algoritmo genere alguna alerta.

Acknowledgments

It is difficult to write technical papers clear and concisely (moreover if you have to write them in English!), but it is more difficult to properly write and acknowledgment, being fair with all the people that helped you, either by directly working side by side, or by supporting and understanding your situation externally. Therefore, I would like to explicitly thank those people, whose help has mainly impacted my researching career, especially in this hard early stage, apologizing to those ones not mentioned. This work is dedicated to all of them.

Firstly, I would like to thank my family for their support and understanding every time, although I have not been able to spend as much time with them as I desire. Their protection and upbringing have made up the man that I am nowadays (well, not completely, they are only responsible for those good things sometimes I do).

Although my friends' contribution to this work is arguable (sorry dudes, but you are better for wasting time ;-]!), they also merit to be acknowledged, because taking the most of the spare time helps to be productive during working hours. Maybe since I started the PhD my free time is reduced and I am not able to see you frequently, but it does not mean our relationship is worse, it just changed. Nobody could snatch me the memories of the good moments we have spent together.

A special acknowledgment is dedicated to my supervisor, Javier Aracil, for his close collaboration and advises, the fruitfulness of this work and the forthcoming ones is mainly your duty. Thank you for allowing me to start my researching career here. In the same way, I would like to thank my colleagues from the Networking Research Group: José Luis García, Víctor López, Jorge López de Vergara, Sergio López, Iván González, Alfredo Salvador, Bas Huiszoon, Luis de Pedro, Javier Ramos, Pedro Santiago, Jaime Garnica, and those that are no longer here: José Alberto Hernández and Walter Fuertes. Thank you for all this time at the laboratory and for making the lab a nice place to work. This also definitely includes my other colleagues from the laboratory Gustavo Sutter, Elías Todorovich, Juan González and Eduardo Boemo. All my work would not have been possible without the support of the Universidad Autónoma de Madrid and the Departamento de Informática of the Escuela Politécnica Superior.

In addition, I would also like to express my gratitude to RedIRIS for providing us with the traffic measurements that have been fundamental to this thesis, to the Spanish Ministry of Education and Science that has funded this research under the F.P.U. fellowship program, and the project TRAMMS that has helped me to grow as a researcher.

Last, but not the least, Cristina, I would like to dedicate this work to you. Your company, support and understanding are invaluable. Thank you for being by my side, we have to celebrate this! ;)

Acronyms

AC Autonomous Community.

AE All Equal.

AM Ante Meridiem.

API Application Programming Interface.

ARIMA Auto Regressive Integrated Moving Average.

BLINC BLINd Classification.

bps bytes per second.

CAIDA The Cooperative Association for Internet Data Analysis.

CDDP CD Database Protocol.

CDF Cumulative Distribution Function.

CEST Central European Summer Time.

CLT Central Limit Theorem.

CoS Class of Service.

DITL Day in the Life of the Internet.

DNS Domain Name System.

DoS Denial of Service.

DPI Deep Packet Inspection.

DRAM Dynamic Random Access Memory.

ECDF Empirical Cumulative Distribution Function.

EM Expectation Maximization.

FP False Positives.

FPR False Positives Ratio.

FTP File Transfer Protocol.

GB Gigabyte.

GMM Gaussian Mixture Model.

GPL General Public License.

HMM Hidden Markov Model.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IANA Internet Assigned Numbers Authority.

IP Internet Protocol.

ISP Internet Service Provider.

IXP Internet eXchange Point.

JB Jarque-Bera.

KE Kernel Estimation.

KS Kolmogorov-Smirnov.

LDA Linear Discriminant Analysis.

MBFP Multivariate Behrens-Fisher Problem.

Mbps Megabits per second.

MI Monthly Increments.

MIB Management Information Base.

MRTG Multi Router Traffic Grapher.

MTU Maximum Transmission Unit.

NBC Naïve Bayesian Classifier.

NBKE Naïve Bayes Kernel Estimation.

NN Nearest Neighbors.

NREN National Research and Education Network.

OC Optical Carrier.

OID Object IDentifier.

OPEX Operational Expenditure.

P2P Peer to Peer.

pcap Packet Capture.

POP Point of Presence.

PPS Packets Per Second.

Q-Q Quantile-Quantile.

QI Quarterly Increments.

QoS Quality of Service.

R&E Research and Education.

RFC Request For Comments.

RMS Root Mean Square.

S&H Sample & Hold.

SCTP Stream Control Transmission Protocol.

SMTP Simple Mail Transfer Protocol.

SNMP Simple Network Management Protocol.

SRAM Static Random Access Memory.

TCP Transmission Control Protocol.

ToS Type of Service.

UAM Universidad Autónoma de Madrid.

UDP User Datagram Protocol.

US United States.

UTC Coordinated Universal Time.

WWW World Wide Web.

Chapter 1

Introduction

Network operators have always been aware of the importance of having detailed descriptions about what is happening in their networks. For this reason, there are a lot of measurement techniques existing in the literature (active and passive), most of them being implemented by network managers that allow them to tackle incidences in the network. For instance, network operators can track malicious traffic to prevent their users of being target of security attacks, assess Quality of Service (QoS) [vdBMvdM+06, MPM05] or bill high consuming clients [EV02]. This increasing interest on network measurements by network operators has been reflected in the research community. There have been a lot of contributions involving network measurements to characterize the Internet traffic [BM01, BC02, DPV06, NAR+04, RK96], or even to characterize specific applications [SFKT06, BS06, PGDM07, PM07, ZSGK08].

All these studies demonstrate the importance of network measurements for network research, however, collecting accurate network measurements have became an arduous task because links' speeds have increased at a larger rate than memory accesses' speeds [Rob00], making it unfeasible to monitor all the network traffic. This misfortune has motivated the development of new techniques to substitute the previously used ones, such as the application of sampling to network measurement [CPB93, Coc97, LG08]. Sampling allows longer measurement campaigns; however, it entails a reduction of the available information. Therefore, the applications of statistical inference and digital signal processing techniques have gained importance, allowing to obtain information of interest. One of the most common ways of obtaining this information is by extracting patterns or footprints that are easily detectable and characterize in an accurate manner the measured traffic [MC00], even measuring these footprints at different time resolutions [PTZD05]. Once the footprints are detected, statistical methodologies are applied to corroborate whether the conclusions obtained from them can be extrapolated or they are just a particular case of the study [KN02].

This master thesis presents the study of different measurement datasets captured with different measurement techniques, as described in Chapter 2, searching for invariants that can be considered as footprints. In a first step, we tried traffic classification based on packet and connection level statistics as footprints. A description of the state of the art in traffic classification and our contribution using parsimonious statistics is presented in Chapter 3. The results were not conclusive, so other approaches were investigated. These are described in Chapter 4, and involve the study of the throughput at the individual user and at the link's aggregation levels.

The throughput analysis conducted us to the concept of utilization, which is the percentage of the link's capacity that it is being used. The utilization showed invariant properties at the level of weeks, mainly due to the well-known daily and weekly pattern behaviors. This invariability allows us to take the utilization over one week without anomalies as a reference footprint that permits comparison to determine the status of a link, i.e. normal behavior or abnormal behavior requiring attendance. An in-depth study of these utilization footprints led us to a new model for the network traffic in an Internet link, using a multivariate normal distribution to represent the traffic of one day. This model is detailed in Chapter 5, jointly with a assessment of its validity. An application of this model is the development of an algorithm for detection of changes in the load of the Internet links (Chapter 6). This algorithm assumes the normality of the samples and applies a powerful statistical procedure to assess the validity of the changes detected by clustering techniques. Finally, Chapter 7 concludes this master thesis and outlines future steps continuing the work presented in Chapters 5 and 6.

Chapter 2

Network Measurements

Network managers are in charge, within other tasks, of keeping network performance under reasonable levels. For this reason, production networks are being monitored continuously, exporting the obtained measurements for further processing. However, the amount of network traffic is humongous, so it is very challenging to handle it in an efficient way. These challenges appear since traffic traversing network links at ever-increasing speeds has to be monitored in a timely fashion. For this reason, different kinds of network traffic monitors have been developed. In this chapter we describe the most common network monitoring tools and the characteristics of the measurement data that are obtained. These measurement data has been deeply analyzed in this study, so it is strongly necessary to understand their advantages and their drawbacks, e.g. the information that can or cannot be extracted from them, their computational costs, etc. The remaining of the chapter is structured as follows. Section 2.1 describe packet captures measurements. Following, the NetFlow records and the definition of flow are presented in Section 2.2. Finally, Section 2.3 describes the information available in Multi Router Traffic Grapher (MRTG) records, and how it is obtained.

2.1 Packet Captures

Packet capture is the process where each packet traversing a link is copied in output files, which are commonly referred as packet traces. This measurement process reproduces exactly the status of the link within the measurement period. The most common format for these packet traces is the one obtained through the Packet Capture (pcap) [JLM93] Application Programming Interface (API), which is used and supported by a variety of network sniffers and packet analyzers.

The advantage of packet captures is that all the available network information is included in the packet traces, i.e. both the payloads and headers. This, however, leads to an important drawback regarding storage requirements. As packet traces contain all the information within a packet, this means that the packet trace size will be equal to the number of bytes of the captured packets. As the speeds of networks are continuously increasing [Rob00], the size of the packet traces is growing at the same rate for a fixed measurement period. This fact makes long packet capture measurement campaigns unfeasible, and it is common to have them split in one hour intervals within one day.

Another negative aspect of packet traces is that packet traces of production networks are very hard to find. The reason for this is related to privacy concerns regarding the personal information that is sent and received in the Internet Protocol (IP) packets, including the IP addresses. Techniques to circumvent these legal aspects are mainly based on anonymization of IP addresses and removal of packet payloads. With these limitations included, there are few packet traces publicly available in the Internet. The anonymized traces used in this study are distributed under request by The Cooperative Association for Internet Data Analysis (CAIDA)[1] and come from Optical Carrier (OC)12 and OC48 Internet Backbone and Exchange Point Data links in the United States.

2.2 NetFlow Records

A flow is defined as a sequence of packets that share the same source and destination IP addresses, port numbers and transport protocol identification. The information that NetFlow stores for each flow entry in its memory includes traffic volume (in bytes and packets), port numbers, source and destination IP addresses, Type of Service (ToS), input and output interfaces indexes (as per Simple Network Management Protocol (SNMP) Management Information Base (MIB)), together with timestamps for the flow beginning and end (see [Cla04] for more detailed description of NetFlow records). All these flow summaries are gathered in a central repository located at the Universidad Autónoma de Madrid (UAM) campus, with an average input rate of 2 Megabits per second (Mbps).

NetFlow is a proprietary format developed by Cisco Systems that runs in their routers and it is implemented by other vendors as well. This protocol is used to monitor the traffic that traverses a router and to keep performance statistics. Cisco defines a flow as a unidirectional sequence of packets sharing all the following 7 values, commonly referred as 7-tuple: Source and Destination IP addresses, IP protocol, Source and Destination ports in case that the IP protocol is Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), Ingress interface and IP ToS.

NetFlow updates the NetFlow record for a flow when a new packet belonging to that flow is sampled, until a timeout counter expires, i.e. when no packets belonging to that flow are sampled for more than "timeout" units of time, or when it samples a packet that finalizes a TCP session, i.e. it samples a packet with either the FIN flag or the RST flag set. The NetFlow sampling method is a deterministic sampling method, i.e., for every N packets it sees, NetFlow samples the first packet and does nothing with the remaining ones.

The NetFlow record contains a wide variety of statistics about the flow, where the most important ones are the timestamps for the flow start and finishing times, number of bytes and packets observed in the flow (that are actually estimations of the real value by taking into account the sampling ratio), as well as the 7-tuple (see [Cla04] for more detailed description of NetFlow records).

Each router with NetFlow capabilities generates NetFlow records, which are exported from the router using UDP or Stream Control Transmission Protocol (SCTP) packets to a NetFlow collector. In the RedIRIS scenario of Figure 2.1, the autonomic routers are routers with NetFlow capabilities that export the NetFlow records to the NetFlow collector located at UAM's premises.

[1] http://www.caida.org

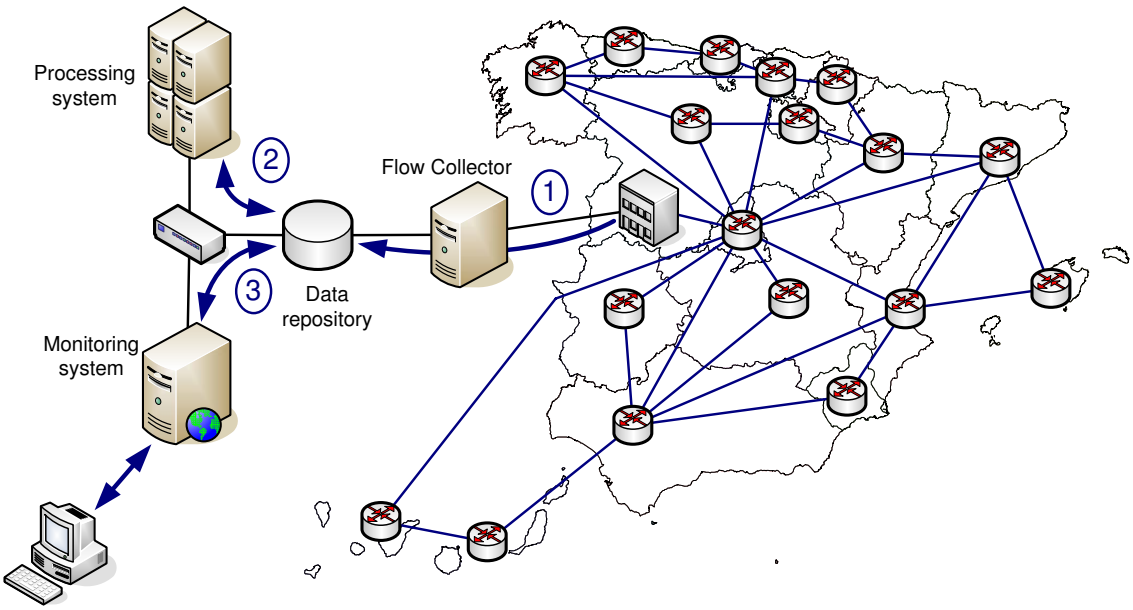

Figure 2.1: RedIRIS Points of Presence.

2.3 MRTG Records

MRTG [OR98] is a software tool distributed under GNU General Public License (GPL) freely available from the MRTG web page[2]. In its origins it was developed as a software to monitor and measure traffic load on network links, graphing the information and showing statistics as maximum, minimum and mean values, but it has evolved to allow the user to visualize almost any kind of information. It is written in Perl, and is available for several operating systems, including Windows, Linux and Mac.

It uses SNMP to send requests to the monitored device. SNMP is an application layer protocol that facilitates the exchange of information between network devices (where a SNMP agent must be running) using MIBs to define hierarchically what information is available to be monitored.

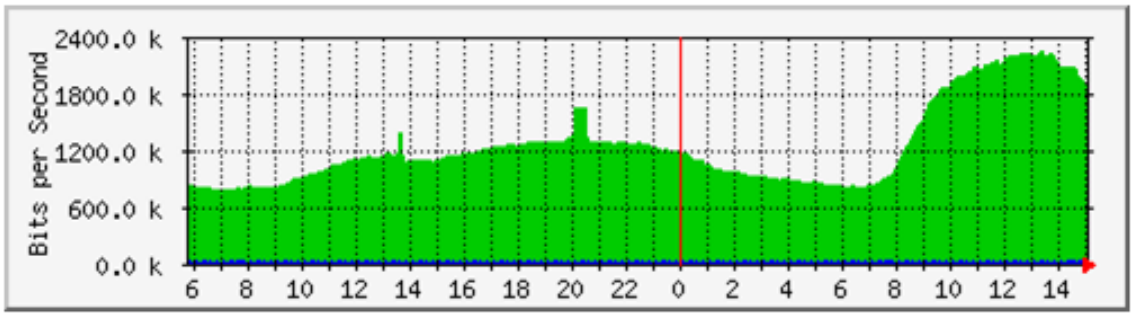

Figure 2.2: Sample one-day MRTG monitoring.

The requests that MRTG sends to a device contain the Object IDentifier (OID) of the resource that it wants to get information about. The SNMP agent of the device looks up the OID in its MIB

[2] http://oss.oetiker.ch/mrtg

and response the MRTG with the corresponding data encapsulated in SNMP protocol. MRTG then gathers all the information received in an incremental database and creates a HyperText Markup Language (HTML) document containing graphs of the received information, as shown in Figure 2.2.

MRTG measures two values per target, the input value and the output value. The input value is plotted as a solid green area and the output one as a blue line, as can be seen in the figure. It collects the data every five minutes for daily graphs, and greater time spans for weekly, monthly and yearly graphs. Furthermroe, MRTG features automatic scaling of the Y-axis to fit the graph to the information area and it also reports the maximum, average and current values for both input and output data, as is shown in Table 2.1.

Table 2.1: Sample statistics for the input and output of the target

	Max	Average	Current
In	2233.0 kb/s (22.3%)	1230.8 kb/s (12.3%)	1894.8 kb/s (18.9%)
Out	880.0 b/s (0.0%)	16.0 b/s (0.0%)	312.0 b/s (0.0%)

Chapter 3

Flow Classification Based on Measurable Parameters

There is a lot of effort from the networking research community in traffic classification. Traffic classification aims to match a measured flow to the application that has produced it by means of packet and flow-level information. The motivations for traffic classification are mainly related with QoS objectives or capacity planning tasks. Thus, either the traffic classification is used to differentiate traffic with different QoS priorities (i.e. with a different Class of Service (CoS))or instead traffic is classified according to bandwidth consumption and flow duration. In the former case, different QoS metrics and thresholds based on their quality requirements are applied, whereas in the latter, the user behavior is characterized to foresee the trends in link usage, on attempts to overtake possible shortages due to congestions issues that may force the network to drop packets and increase the delays in packet delivering. In what follows, a description of the state of the art concerning traffic classification is summarized in Section 3.1. A deeper study on this techniques is available in [MA08]. Next, our contribution to traffic classification is described in Section 3.2. Finally, Section 3.3 concludes the chapter.

3.1 State of the Art

The state of the art in traffic classification can be divided into classic traffic classification techniques and innovative traffic classification techniques, as follows.

3.1.1 Classic Traffic Classification Techniques

Classic traffic classification techniques make use of flow and packet level information to characterize different kinds of traffic. The resulting classification is coarse and can be commonly applied on-line.

Flow-level Statistics

Flow-level classification aggregates packets traversing a link by the 5-tuple that describes its headers (as described in Section 2.2 of Chapter 2) and computes statistics of the aggregates as flow duration, bandwidth consumption, transferred bytes, number of packets per flow, the means and

the variances of these metrics, etc. This is no more than the available information from NetFlow records. A limitation is that flow-collection may sometimes aggregate packets that belong to multiple application-level connections into a single flow, which distorts the flow-level features.

Dragonflies and tortoises The classification of traffic into dragonflies and tortoises is based on the duration of the flows. This classification was first introduced by Brownlee & Claffy in [BC02], where they defined dragonflies as very short duration flows, lasting up to 2 seconds, and tortoises as long-running flows, being active for more than 15 minutes. A flow is considered to time out once a TCP RST packet or a pair of TCP FIN packets is seen through the link. This assumption only works for TCP flows and can fail due to packet loss or asymmetric routes. For this reason, a timeout interval is set so that if no packets are observed for that dynamically specified timeout interval, the flow is considered to be timed out (the same criteria is used by NetFlow).

Mice and Elephants Another flow-level classification study classifies network traffic into the categories of mice and elephants. This classification arises from the observation that a very small percentage of flows consumes the majority of the available bandwidth, what is commonly referred as "the elephants and mice phenomena". The classification assigns large flows (i.e. with a large amount of bytes) to the elephant class, and small flows to the mice class. Commonly, the classification schemes are based on a separation threshold that elephants have to exceed [PTB+02].

Packet-level Statistics

Packet-level classification keeps statistics of packets that are not related with aggregation based on parameters of theirs headers. Features in this level are simple to compute and packet-level sampling, which is widely used in network data collection, has little impact on them. They offer a characterization of the application that is independent of the notion of flows, connections or other higher level aggregations. Examples of this statistics are mean packet size or time series of this data, from which can be derived a number of features, as correlations over time of this values.

Port Based Classification Packet-level header inspection has been used to associate network traffic with the application that produces it based on TCP or UDP port numbers. The port numbers are divided into three ranges: the Well Known Ports (0-1.023), the Registered Ports (1.024-49.151) and the Dynamic Ports (49.152-65.535). All the packets sent within either a TCP connection or UDP session use the same pair of ports to identify the client and server sides. Therefore, theoretically, TCP or UDP server port numbers can be used to identify higher layer applications, by simply identifying which port is the server port and mapping this port to an application using the Internet Assigned Numbers Authority (IANA)[1] list of registered ports. However, port-based application classification has limitations. First, the mapping from ports to applications is not always well defined. For instance from [RSSD04], it turns out that

- Many TCP implementations use client ports in the registered port range. This might mistakenly classify the connection as belonging to the application associated with this port. Similarly, some applications use port numbers from the well-known ports to identify the client site of a session.

[1] http://www.iana.org/assignments/portnumbers

- An application may use ports other than its well-known ports to circumvent operating system access control restrictions, e.g., non-privileged users often run World Wide Web (WWW) servers on ports other than port 80, which is restricted to privileged users on most operating systems.
- There are some ambiguities in the port registrations, e.g. port 888 is used for CD Database Protocol (CDDP) and access-builder.
- In some cases, server ports are dynamically allocated as needed. For example, File Transfer Protocol (FTP) allows the dynamic negotiation of the server port used for the data transfer. This server port is negotiated on an initial TCP connection which is established using the well-known FTP control port.
- Trojans and other security (e.g. Denial of Service (DoS)) attacks generate large volume of bogus traffic which should not be associated with the applications of the port numbers those attacks use.
- The use of traffic control techniques, like firewalls to block unauthorized and/or unknown applications from using a network, has spawned many work-arounds which make port-based application authentication harder. For example port 80 is being used by a variety of non-web applications to circumvent firewalls which do not filter port 80 traffic. In fact, available implementations of IP over Hypertext Transfer Protocol (HTTP) allow the tunneling of all applications through TCP port 80.
- Ports are not defined by IANA for all applications, e.g. Peer to Peer (P2P) applications such as eMule or BitTorrent.

Intra-flow/connection Features

There are very interesting features that network monitors might wish to collect, which are based on the notion of a flow or TCP connection, but require statistics about the packets within each flow. One example is the statistics of the inter-arrival times between packets in flows, e.g. this requires data collected at a packet level, but then grouped into flows. Other statistics included in this group are loss rates, latencies, packet size distribution for a given application, etc.

The authors of [DPV06] use intra-flow/connection statistics to characterize HTTP and Simple Mail Transfer Protocol (SMTP) traffic. They collect all the packets destined to and sent from ports 80 (HTTP) and 25 (SMTP) and compute packets, bytes and inter-arrival times distributions in both directions (from client to server and from server to client). They show that characterizations at this level for this protocol have invariant properties, in terms of spatial, i.e. same behavior at different links, and temporal, i.e. same behavior at different observation instants.

3.1.2 Alternative Classification Methods

In this section we describe innovative techniques applied to traffic classification. Such techniques are well-known methods, as for example Bayesian analysis techniques or Hidden Markov Model (HMM), which have been widely uses in other areas but not in Internet traffic classification, or new developed ideas that work well when applied to traffic classification.

Well-known Methods Applied to Traffic Classification

We start describing well known methods that have been applied to classify traffic. The classification techniques that we will describe make use of HMM, Bayesian analysis, Nearest Neighbors (NN) and Linear Discriminant Analysis (LDA).

Hidden Markov Model traffic classification In [OSST04] the authors use a two state (high mean state and low mean state) HMM approach similar to those used nowadays in speech recognition systems to classify Internet traffic. In each state $j; j \in \{1, \ldots, N\}$, flow rate is assumed to follow a Gaussian distribution $\mathcal{N}(\mu_j, \sigma_j)$ with mean μ_j and variance σ_j. The transition matrix $\mathbf{\Gamma}$ between the Markov chain states is given by $p_{n,m}$, i.e. the probability of stay in state m when in the previous instant it was in state n.

The parameters of the model $(\mu_j, \sigma_j, pn, m; j, n, m \in \{1, \ldots, N\})$ are chosen using a maximum likelihood criteria, i.e. the parameters are chosen to maximize the probability that the observed sequence came out of a HMM with these parameters, using the Expectation Maximization (EM) algorithm.

The first step in the classification is to extract features for each flow i over a time window t of size l. The second step is to classify the flows based on the features extracted in the previous step. For this purpose [OSST04] assumes that the features are distributed following a Gaussian Mixture Model (GMM) of dimension l with a predetermined number of classes K. Once the GMM is calibrated, the membership probabilities π_{ik}^t, i.e. the probability that a given flow i characterized by a feature vector belongs to the class k, can be derived. They use these probabilities to classify flows into two and three classes.

The two classes' classification was not able to give meaningful results. Classes are mixed together and there is no clear separation between the two classes. In the three classes' classification, flows that have a very low probability of remaining in the high state (dragonflies) are separated from those that have large value of high state mean value and high probability of remaining in the high state (elephants) and the ones that are in the middle (mice).

Bayesian Analysis Traffic Classification The authors of [MZ05] use Bayesian analysis techniques to classify traffic in ten different classes. These classes and an example of each class are the following: BULK (ftp), DATABASE (oracle), INTERACTIVE (ssh), MAIL (smtp), SERVICES (dns, ntp), WWW (http), P2P (KaZaA), ATTACK (worms), GAMES (Counter-Strike) and MULTIMEDIA (Real player). The discriminators they use as input parameters are for example flow duration, TCP port, packet inter-arrival time and its moments, payload size and its moments, Fourier transform of the packet inter-arrival time, etc.

They propose two different algorithms, a simpler one named Naïve Bayesian Classifier (NBC), and an improved version with kernel estimation. The NBC tries to calculate the probability of belonging to a class $c_j; j \in \{1, \ldots, M\}$ given an observation of the discriminators for a flow y. This probability is denoted by $p(c_j \mid y)$, and is computed by applying the Bayes Rule:

$$p(c_j \mid y) = \frac{p(c_j) f(y \mid c_j)}{\sum_{c_j} p(c_j) f(y \mid c_j)}, \tag{3.1.1}$$

where $p(c_j)$ denotes the probability of obtaining class c_j independently of the observed data (prior distribution), $f(y \mid c_j)$ is the distribution function (or the probability of y given c_j) and the denominator acts as a normalizing constant. To estimate $f(y \mid c_j)$, $j = 1, \ldots, k$, a training set $\mathbf{x}$ is used, assuming that the discriminators are independent with a Gaussian behavior.

The Naïve Bayes Kernel Estimation (NBKE) is similar to the Naïve Bayes method algorithmically. The only difference is that the estimate of the real density $f(\cdot \mid c_j)$ is given by

$$\hat{f}(t \mid c_j) = \frac{1}{n_{c_j} h} \sum_{x_i \,:\, C(x_i)=c_j} K(\frac{t - x_i}{h}), \tag{3.1.2}$$

where h is called the kernel bandwidth and $K(t)$ is any kernel, where kernel is defined as any non-negative function such that $\int_{-\infty}^{\infty} K(t)dt = 1$.

Despite NBKE shows to behave better than the simple NBC, it has greater computational costs that can make the simple NBC more appealing. The results presented by [MZ05] on the application of these methods to traffic classification show that NBC has a poor accuracy (about 65%), but the use of Kernel Estimation (KE) and other improvements (preprocessing to remove redundant and irrelevant variables) raise this accuracy to more than 90%. The accuracy was defined as the ratio between the number of flows that were classified correctly and the total number of flows.

They also computed a per-class measure of trust that indicates how much reliable is the classification. With this measure, [MZ05] shows that MAIL and WWW categories are very well classified, having a trust measure greater than 90% in the NBC, but the remaining applications were not classified satisfactorily. With the enhancement of KE, as well as the MAIL and WWW, SERVICES and DATABASE categories were successfully classified, and BULK and MULTIMEDIA categories obtained an acceptable trust level (about 77%).

Nearest Neighbor and Linear Discriminant Analysis Traffic Classification This subsection is devoted to describe the work presented in [RSSD04]. They search for discriminators that can characterize applications, so a clustering algorithm can be done based on these discriminators to classify flows in different classes. They define four different classes which are Interactive, which contains traffic that is required by a user to perform multiple real-time interactions with a remote system (e.g. remote login sessions); Bulk data transfer, which contains traffic that is required to transfer large data volumes over the network without any real time constraints (e.g. FTP); Streaming, which contains multimedia traffic with real-time constraints (e.g. video conferencing); and Transactional, which contains traffic that is used in a small number of request response pairs which can be combined to represent a transaction (e.g. Domain Name System (DNS)).

To test the goodness of the proposed method, [RSSD04] goes one step further than [MZ05] and, instead of classifying traces manually, they classify a few representative applications for each class, so all the applications not selected as representative are clustered with the representative with closer features. The reference selected applications are the following: Telnet for Interactive class; FTP-data and Kazaa for Bulk data transfer; RealMedia streaming for Streaming class; and DNS and Hypertext Transfer Protocol Secure (HTTPS) for transactional class.

The methods proposed by [RSSD04] are simple, but commonly used for classification: NN and LDA. The NN method is based on the assumption that the class of a new data point is the class of the point which is closest using the Euclidean distance. This method can be generalized to k-NN

to enhance it robustness, where the k nearest neighbors 'vote' on the class of the observation. k-NN work well with low-dimensional data, but are less effective on high-dimensional samples. The LDA method uses the posterior probability of belonging to class n, using the Bayes rule and assuming that each class g has a Gaussian distribution with a given mean μ_g and with the same intra-class covariance Σ for each class. With these assumptions, the log of the ratio between the probabilities of belonging to two different classes can be simplified to a linear function which is called the linear discriminant functions.

To apply these methods, [RSSD04] evaluated the following easily obtainable features: the average packet size, flow duration, bytes per flow, packets per flow, and Root Mean Square (RMS) packet size. Of these, the most valuable pair was the average packet size and flow duration, and [RSSD04] considers such characteristics to classify the reference applications mentioned above.

The results show that the classification does not separate Streaming and Bulk data transfer classes adequately, so a new feature to distinguish these classes is introduced. This feature is referred as the inter-arrival variability metric and is the mean of the ratio between the variance and the mean of the inter-arrival times for the packets belonging to a flow. It was also found by the authors that some streaming traffic ended up with a long gap followed by a few packets. This behavior makes it worse the ability to separate classes of the presented methods, so to avoid it [RSSD04] proposes to ignore the final 10 packets from each flow. The use of this metric jointly with the average packet size gives a product space where these two classes are linearly separated.

Innovative Traffic Classification Techniques

In this section an innovative method presented in [KPF05] is described, namely BLINd Classification (BLINC). It has been selected due to the relevance of the ideas presented in it. This method relies on the observation and identification of patterns of host behavior.

BLINC BLINC [KPF05] uses a multilevel approach to traffic classification where the patterns of host behavior are analyzed at three levels of increasing detail: (*i*) the social, (*ii*) the functional and (*iii*) the application level. BLINC has no access to packet payload and port numbers neither has additional information other than what current flow collectors provide.

Firstly, packet traces are gathered to test the BLINC classifier. These traces are deep packet inspected to classify all the applications, so the goodness of the BLINC technique can be evaluated. This payload-based classification looks for specific bit strings and inspects well-known ports. These bit strings are identified either from Request For Comments (RFC) and public documents in case of well-documented protocols or by reverse engineering. Finally, the payload is searched for this bit strings and initial protocol handshakes that allow classifying by application type. An IP, port number table that contains all the flows already identified is used to speed up the classification. With these assumptions, [KPF05] classifies more than 50% of the flows in their traces. The flows that were not classified corresponded to non-payload and unknown flows.

The non-payload flows account for almost one third of all flows in the traces, however as they had not payload, they only account for a small percentage of the bytes (about 2%). These flows seem to be failed TCP connections from worms that tried to attack hosts. The unknown could be due to new applications that were not taken into account in the bit string creation, but this will

be one of the strengths of BLINC, that it may be able to classify new applications into categories that fit with their characteristics.

The main differences between BLINC and classical approaches to traffic classification are that BLINC does not treat each flow as a distinct entity, but it focuses on the source and destination hosts of the flows; it operates on flow records and requires no information about the timing or the size of individual packets; and it is insensitive to network dynamics such as congestion or path changes, that can potentially affect statistical methodologies which rely on inter-arrival times between the packets in a flow.

BLINC gathers host-related information reflecting transport layer behavior and then they associate the host behavior with one or more application types therefore classifying the flows. The host behavior is studied across three levels: at the social level, the popularity of a host is defined as the number of other hosts it communicates with; at the functional level, the behavior of a host in terms of its functional role in the network is captured, i.e. whether it is a provider or consumer of a service, or whether it participates in collaborative communications; finally, at the application level, transport layer interactions between hosts are captured with the intention of identifying the origin application.

The classification results of [KPF05] are presented with two different metrics. The first metric is the completeness, which measures the percentage of the traffic classified by BLINC. This ratio is computed at flow level and at byte level. The completeness over the three traces used to test BLINC lay between 80-95%. The other metric to test the goodness is the accuracy, which measures the percentage of the classified traffic by BLINC that is correctly labeled. BLINC results show that it has a high accuracy, being it higher than 95%.

3.2 Contribution

This section is devoted to describe our contribution to traffic classification. We performed application classification based on packet level and intra-flow/connection statistics through clustering techniques. A brief description of the objectives of the classification and the statistics of interest is presented in Section 3.2.1. Our method obtained such statistics from packet captures publicly available without payload from CAIDA. These datasets are described in Section 3.2.2. Next, Section 3.2.3 presents the results obtained through the clustering technique. Finally, Section 3.2.4 shows the results of the same clustering technique after applying the Sample & Hold (S&H) technique [EV03] to focus on the most contributing flows.

3.2.1 Description of the Contribution

Our contribution aims to classify traffic according to two main different groups in order to use this membership information and then apply specific QoS measurements. Such groups are Real Time interactive applications, like Skype [Lim], which have stringent QoS requirements, and Bulk data transfer applications, like FTP, whose QoS requirements are not so demanding. We follow an approach similar to that of [RSSD04], but started with only two applications groups to facilitate the classification paradigm. We assume that all the packets sent and received within a flow as defined in Section 2.2 belong to the same application. Therefore, we measure in the traffic traces

described in Section 3.2.2 six packet and connection level statistics for each flow. Such statistics are described in Table 3.1. As a flow is a bidirectional transaction of packets, we take different statistics from each direction, as there may be relevant differences between them depending on which end started the communication.

Table 3.1: Packet and connection level statistics used for flow classification

Statistic	Description
Mean packet size	Average packet size for all the packets sharing the same flow descriptor
Flow duration	Difference between the timestamp of the last packet and the first packet belonging to the same flow. Flow ending conditions are the same used by NetFlow
Number of packets	Total number of packets sent within the flow
Flow size	Total number of bytes sent within the flow
Mean packet inter-arrival time	Average time between arrival of contiguous packets within the same flow
Packet rate	Number of packets per second sent within the flow

Our contribution then groups flows according to distances between the obtained statistics using k-means. We briefly describe the k-means procedure in Appendix A. The main reason not to select NN as clustering technique, following the method presented in [RSSD04], is that NN needs a dataset labeled in advance, from which the representatives of the classes are obtained. This dataset should therefore have been manually inspected and classified. This is a very challenging task, moreover if the dataset has no payload, as are the datasets used in our study. The unfeasibility to manually classify a packet trace prevented us from applying NN, therefore selecting k-means, which does not need a train set classified beforehand.

3.2.2 Description of the Packet Traces

For the extraction of the statistics described in Section 3.2.1 we analyzed two data collections publicly available from CAIDA. These data collections were captured in different points and time instants. Following we describe each of the collections.

CAIDA OC48 Traces

CAIDA made available three collections of traces captured within 2002 and 2003. From these collections, we selected the most recent one that was captured in Apr 24, 2003 at 9 Ante Meridiem (AM) Central European Summer Time (CEST) on an OC48[2] link and lasted for an hour. This collection was composed by 12 anonymized packet header traces of 5 minutes long for each direction. The monitored link is a west coast peering link for a large Internet Service Provider (ISP). These traces consist of packet headers found in the first 48 bytes of packets, with IP addresses anonymized with the prefix-preserving Crypto-PAn [FXAM04] library. These traces do not include non-IPv4 traffic. Packet timestamps are likely precise to the microsecond. The size of the traces for both directions is 13 Gigabytes (GBs) and contains 203 million of packets.

[2] up to 2488.32 Mbps

AMPATH OC12 Traces

One of the kinds of applications that we wanted to classify with our clustering technique was these with stringent QoS requirements, namely real time applications. One of such applications is Skype [Lim], whose classification has attracted a lot of researchers due to its complexity ([EP06, PGDM07, PM07, RMM08, SFKT06]) and its broad utilization. However, the publication date of Skype software was August 2003, i.e. after the capturing time of CAIDA OC48 traces. This motivated us to stop analyzing the CAIDA OC48 traces and focus on newly traces, although the information from the CAIDA OC48 traces was still valuable to classify the remaining class of applications of interest.

We found newly traces available from CAIDA again, captured on 2007. This collection contains anonymized pcap packet header traces collected on both directions of an OC12[3] link at the AMPATH International Internet eXchange Point (IXP) in Miami, Florida. This OC12 link carries traffic between United States (US) Research and Education (R&E) networks and R&E networks in South and Central America. These traces were collected as part of the Day in the Life of the Internet (DITL) project [Cla06]. They cover the full 2 days of DITL-2007-01-09 which started midnight 2007-01-09 Coordinated Universal Time (UTC) and ended midnight 2007-01-11 UTC, and consist of over 850 million IPv4 packet headers in hourly files.

3.2.3 Results of the Traffic Classification Through Clustering Techniques

In this section we present the results of our contribution. These results are in the form of scatter plots of two and three dimensions, being these dimensions some of the statistics of Table 3.1 and each plotted marker an analyzed flow from the packet traces. We performed a visual inspection of the results, starting by graphing the statistics by pairs. Figure 3.1 is an example of these graphs, where we have represented the duration of the flows versus the mean packet length of theirs packets. These statistics were measured from most loaded hour trace of the AMPATH OC12 traces (therefore, the maximum duration of the flows is 3600 seconds, which explains the accumulation of flows in that duration). We represent TCP flows with a red $\times$ and UDP flows with a blue $\circ$.

As can be seen in the figure, there is a concentration of flows with small mean packet length. This length is around 1500 bytes, and explains the concentration above it because it is the value for the Maximum Transmission Unit (MTU) of Ethernet V2 [MD90] and nearly all the implementations of IP over Ethernet use this frame format. As the amount of flows under this mean packet length is very large, we zoom in this region in Figure 3.2.

The number of flows in Figure 3.2 is humongous and highly dense in short durations, thus preventing us from obtaining meaningful clusters. The situation throughout the remaining combination pairs of statistics is quite similar, being also difficult to obtain clusters on them. For that reason, we graphed the statistics by trios (Figure 3.3).

In that figure, we have removed flows with mean packet size greater than 1500 bytes and packet rates that exceed 5000 Packets Per Second (PPS) for the same reason as in Figure 3.1. There we can see an agglomeration of flows with of short duration, small mean packet size but very

[3] up to 622.08 Mbps

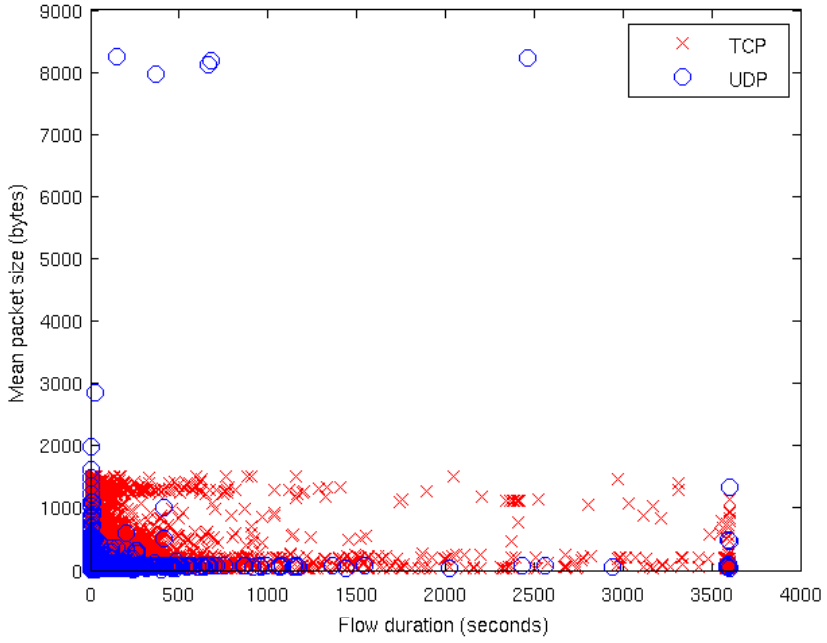

Figure 3.1: Scatter plot of flow duration versus mean packet size.

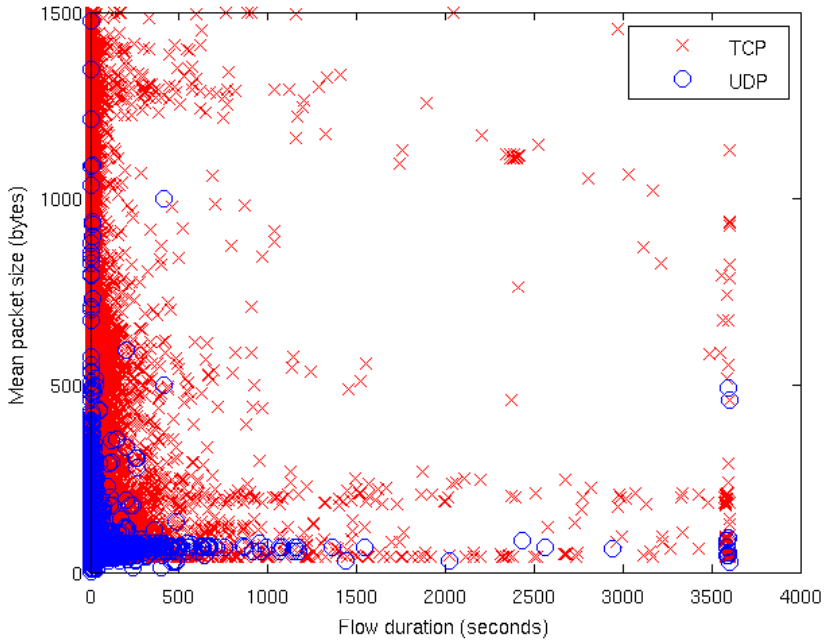

Figure 3.2: Scatter plot of flow duration versus mean packet size focused on flows with mean packet length smaller than 1500 bytes.

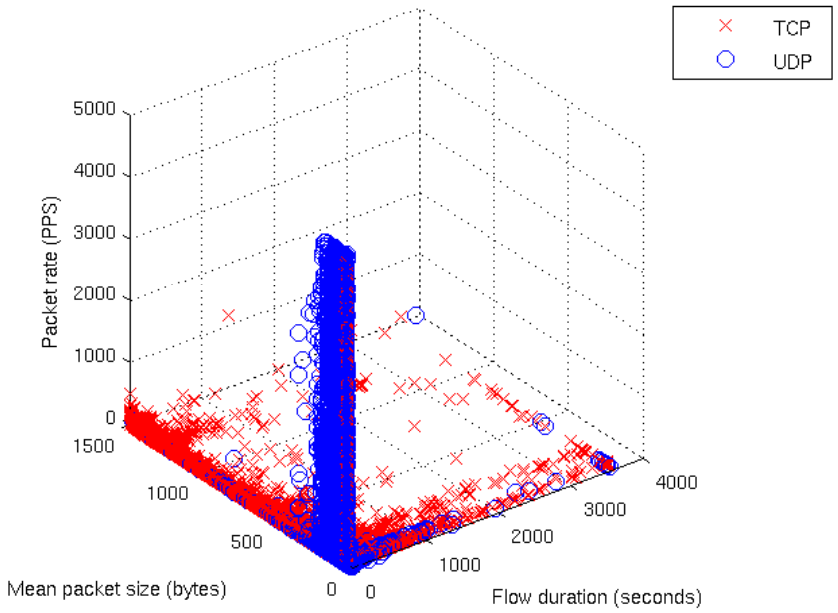

Figure 3.3: Three dimensional scatter plot of flow duration, mean packet size and packet rate focused on flows with mean packet size smaller than 1500 bytes and packet rate smaller than 5000 packets per second.

large packet rate (the column around the origin in Figure 3.3). These agglomeration could be thought as a cluster, but it is not of interest for our objective, as the huge amount of flows in that region entails an application mix that hampers the classification into the classes defined in Section 3.2.1. In addition, the classification of these flows has little attractiveness, because you cannot take actions to improve those flows QoS in a timely fashion, i.e. before the flows finish.

3.2.4 Results of the Traffic Classification Focused on the Most Contributing Flows

The reasons described at the end of the previous section motivated us to apply the S&H methodology (see Appendix B for a description of the technique) to focus on the most contributing flows, i.e. those with more than 0.01% of the capacity of the link[4]. In addition, we also remove the short lived flows during less than 120 seconds although S&H did not remove them (note that S&H results are not exact neither deterministic), due to its lack of importance. We show these results in Figure 3.4.

As can be seen in the figure, S&H has removed the column of short lived with high packet rate flows concentrated around the origin, and now there are not flows with packet rates greater than 800 PPS (in this figure we have not focused on this region, just there are not flows with higher rates). Although the amount of flows removed from the S&H technique is very large, we still do not find any remarkable cluster where applications of the same class are concentrated.

[4]As we are analyzing hour length traces, the capacity is the OC12 link speed multiplied by length of the traces: Capacity = 622.08 Mpbs x 3600 sec $\approx$ 2 Tb.

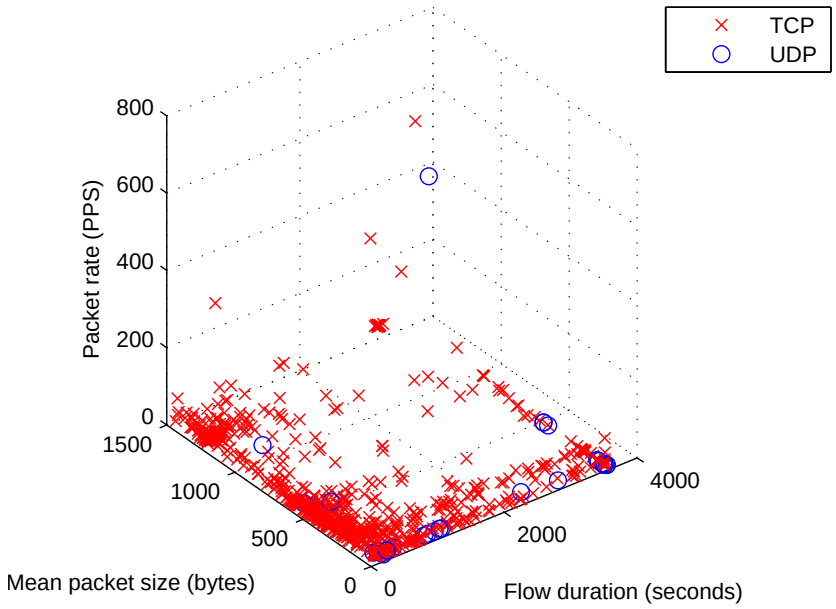

Figure 3.4: Three dimensional scatter plot of flow duration, mean packet size and packet rate focused on flows with mean packet size smaller than 1500 bytes after applying SH.

3.3 Summary and Conclusions

We have presented in this chapter a summary of the state of the art in traffic classification. This research field is receiving a lot of attention from the research community, but the approaches they are taken are moving to Deep Packet Inspection (DPI), where the payload of the captured packets is inspected on attempts to find certain bit sequences commonly referred as signatures. Although DPI techniques have shown to obtain better results than non-DPI methodologies, the DPI approach is very demanding in terms of hardware requirements, and its online application is very challenging. However, a traffic classification approach based on packet and connection level parameters is less demanding and can be easily deployed online. Unfortunately, the results of our approach showed in this chapter are not sufficiently promising to justify continuing with this line of research.

Chapter 4

Throughput Analysis

In our search for traffic footprints, we decided to analyze the throughput of the flows. Throughput is defined as the average rate of successful packet delivery over a communication network, i.e. the speed at which the receiving end of a communication is getting error-free packets. At first glance, the throughputs of the flows traversing a link seem to be a reasonable statistic to determine the links' status, as they might reveal low values where there are anomalies or the link is under-dimensioned, or in the contrary they should reach a high stable value maintained during most of the flow duration. Therefore, we proceeded to analyze the throughput of the flows contained in the packet traces described in Section 3.2.2.

First of all, we analyzed the mean throughput of the flows and graphed their histograms and CDFs. The results of this analysis are presented in Section 4.1. The distribution of the throughput does not resemble any common distribution, but it seems to be a mixture of distributions. As making inference with this underlying distribution is very challenging, we decided to assume normality and compute hourly confidence intervals for the mean value of the throughput. The results and motivations for the computation of these confidence intervals are presented in Section 4.2. These results were not as expected, so we computed instantaneous values for the throughput and moved to a time series analysis of these values (Section 4.3). Finally, we summarize the throughput analysis presented in this section with the conclusions obtained from the results in Section 4.4.

4.1 Histograms and CDFs of the Throughput

We have computed for each flow observed in the analyzed packet traces its average throughput during its lifetime. This analysis was performed on an hourly basis (e.g. taking only into account the flows active between 12:00-13:00 to obtain the corresponding histograms for that hour interval, therefore computing the throughput of the flow during that hour in case the flow is longer than one hour) and in a daily basis (i.e. taking into account all the flows observed in the whole day). We treat the mean value of the throughput for each flow as a realization, and compute the corresponding histograms and empirical CDFs. We remove from this sample all the flows with less than 50 packets, in order to remove small flows that are not of interest for QoS actions and can introduce a bias in the distribution, and make a distinction whether the protocol is TCP or UDP, then making three different plots for each study (one for TCP, other for UDP and another

one without making the distinction, hereafter total).

We represented the results for the total when measuring the whole day, however, in that representation we can only distinguish a huge first bin, whereas the frequencies of the remaining bins are nearly negligible. This happen because there are little flows with a high mean throughput, whereas the majority of the flows have small throughput values compared with the observed maximum. Therefore, we remove those flows with high value and represent again the histograms. Those are depicted in Figure 4.1, Figure 4.2 and Figure 4.3 respectively for total, TCP and UDP.

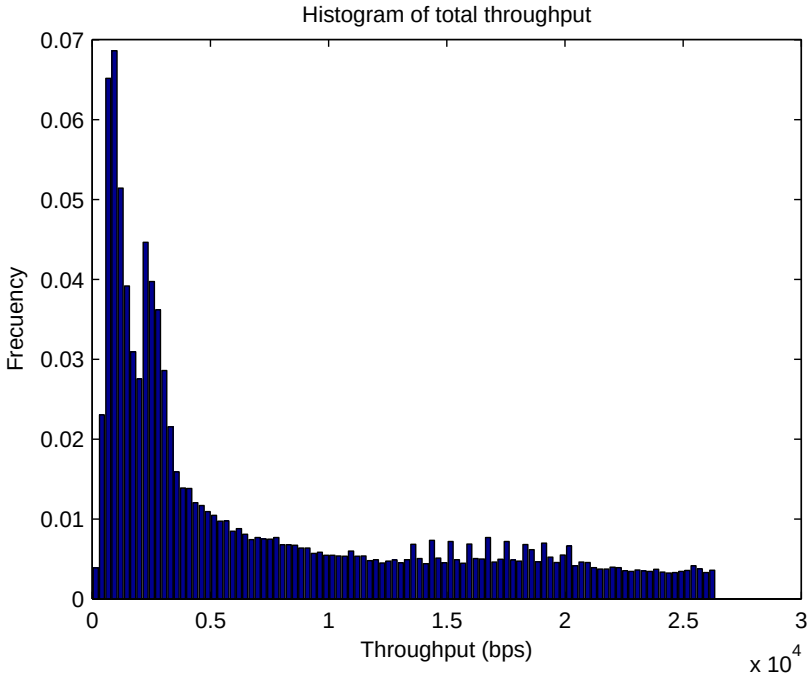

Figure 4.1: Histogram of the mean value of the throughput for TCP and UDP flows, representing flows with more than 50 packets and average throughput smaller than $2.6 \cdot 10^4$ bps.

As can be seen in the figures, the distribution of the mean throughput is quite similar between TCP and UDP (therefore also for its combination), but unfortunately it does not resemble any well-known distribution. This can be also confirmed with the Cumulative Distribution Function (CDF) of the total traffic shown in Figure 4.4 (we do not present the analogous figures for only TCP and UDP traffic as they are close similar to Figure 4.4).

The analysis of the mean throughput on an hourly basis evidenced two remarkable results. On the one hand, the histograms for the two protocols are quite different, as shown in Figure 4.5 and Figure 4.6 for TCP and UDP, respectively. We have selected the analysis interval 10:00-11:00 as we found it representative of the whole analysis by hour intervals. The great disparity between those figures is quite surprising, because this was not reflected when we analyzed the whole day (compare with Figure 4.2 and Figure 4.3). This disparity makes clear the differences between both protocols, where in TCP, due to slow start and other congestion avoidance mechanism [Ste97], a great number of the flows have low mean throughput values, thus having high density in the first bin of the histogram (16%). These mechanisms are not used in UDP, so the main peak in the histogram is found at greater mean throughput values (around 175 bytes per second (bps)) and

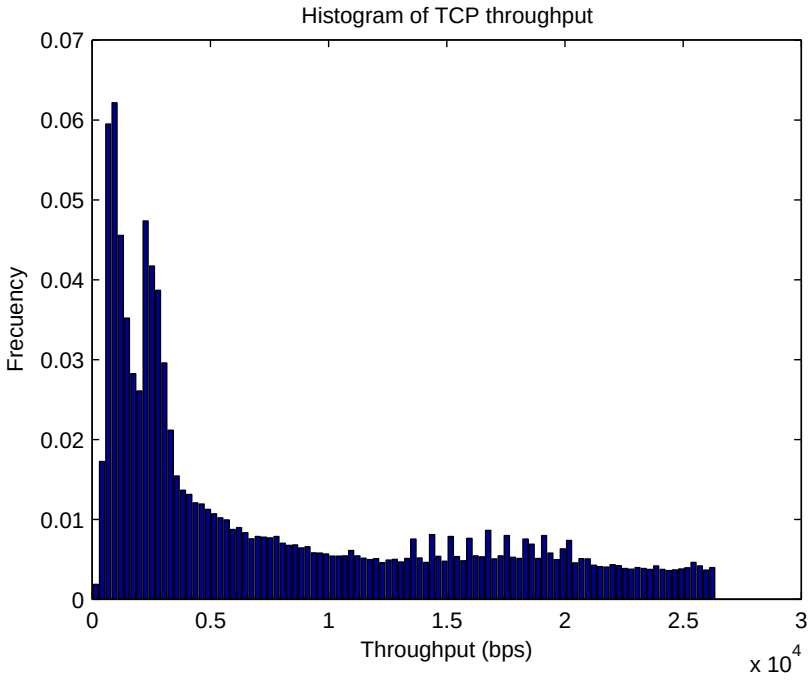

Figure 4.2: Histogram of the mean value of the throughput for TCP flows, representing flows with more than 50 packets and average throughput smaller than $2.6 \cdot 10^4$ bps.

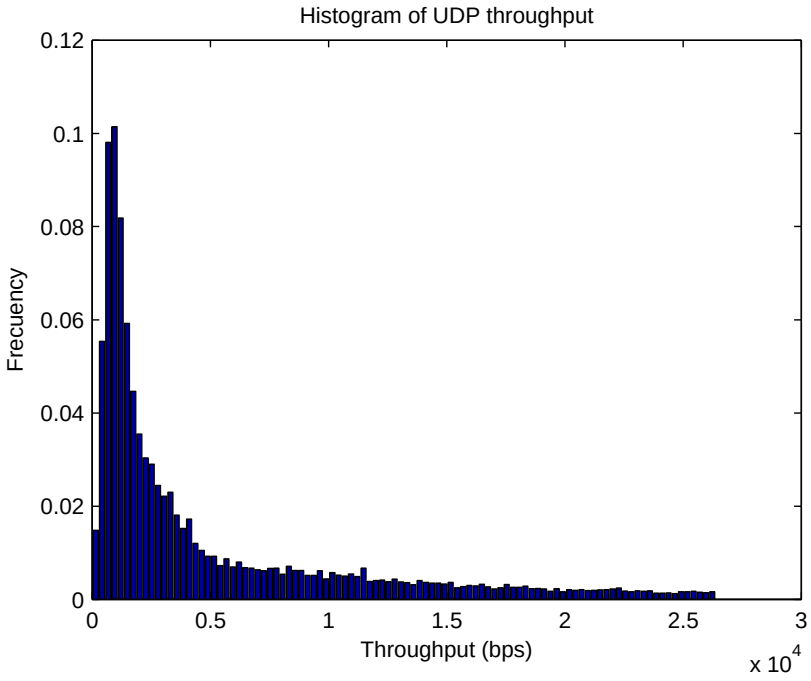

Figure 4.3: Histogram of the mean value of the throughput for UDP flows, representing flows with more than 50 packets and average throughput smaller than $2.6 \cdot 10^4$ bps.

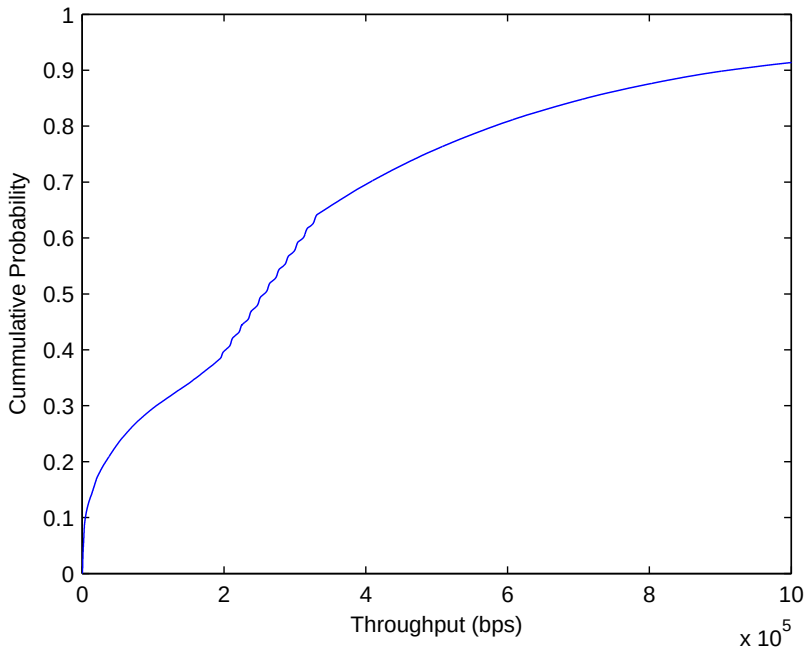

Figure 4.4: ECDF of the mean value of the throughput for the aggregate of all the traffic within the analyzed day, representing flows with more than 50 packets and average throughput smaller than 10^6 bps.

has lower density (10%). In addition, as the number of UDP flows is considerably smaller than the number of TCP flows, we found in the UDP histogram empty bins, meaning that there are not flows with that mean throughput value.

On the other hand, we found that the TCP histogram shapes the histogram of the total traffic within the link (Figure 4.7). This behavior could be hinted in the histograms for the whole day (Figures **??** and 4.1), although the hourly analysis sheds light on this fact and removes any possible doubt.

The graphs for the remaining hours not presented here, in addition to videos representing the evolution of the hourly Empirical Cumulative Distribution Functions (ECDFs) can be found in the following link
`http://www.eps.uam.es/~fmata/Publications/Files/throughput%20analysis.rar`.

4.2 Confidence Intervals for the Mean Values of the Throughput

The understanding of the shapes of the distribution of the mean values of the throughput let us analyze further the obtained results. If the results would have evidenced that the mean value of the throughput follows some of the well-known distributions, we could have applied specific statistic inference (after testing that the mean value of the throughput actually follows that supposed distribution). However, from the analysis of the previous section, we cannot conclude that the mean value of the throughput follows any well know distribution. Thus, we have to suppose a

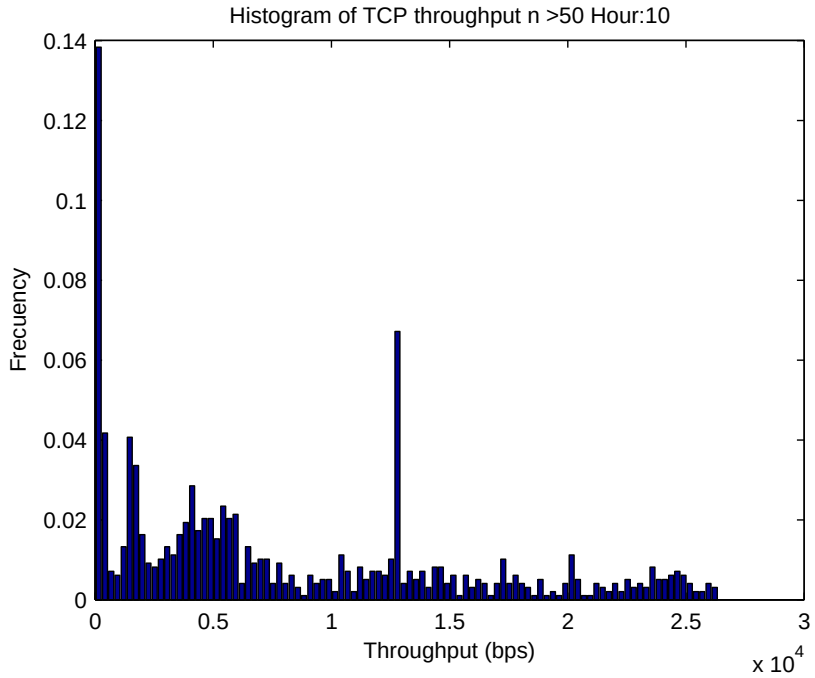

Figure 4.5: Histogram of the mean value of the throughput for TCP flows, representing flows with more than 50 packets and average throughput smaller than $2.6 \cdot 10^4$ bps for the time interval 10:00-11:00.

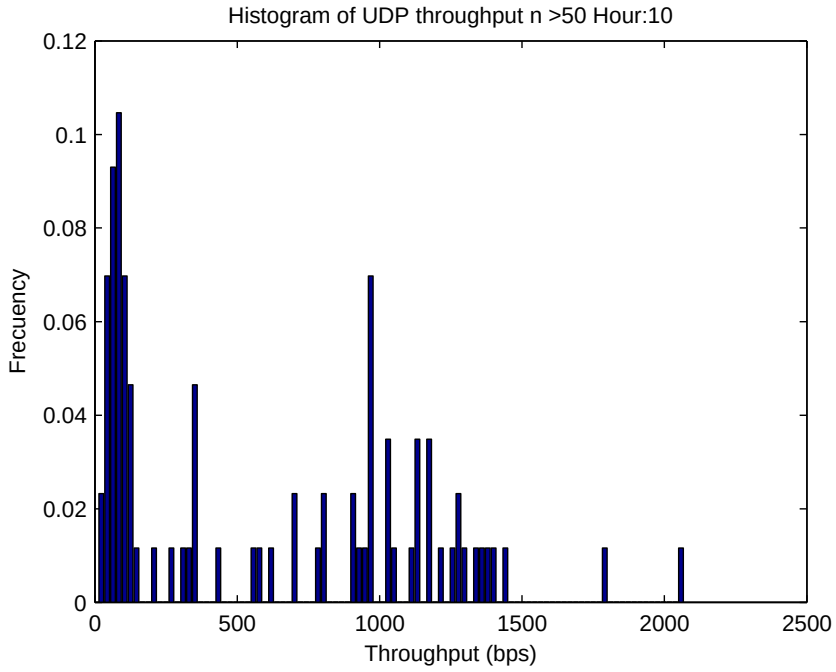

Figure 4.6: Histogram of the mean value of the throughput for UDP flows, representing flows with more than 50 packets and average throughput smaller than $2.6 \cdot 10^4$ bps for the time interval 10:00-11:00.

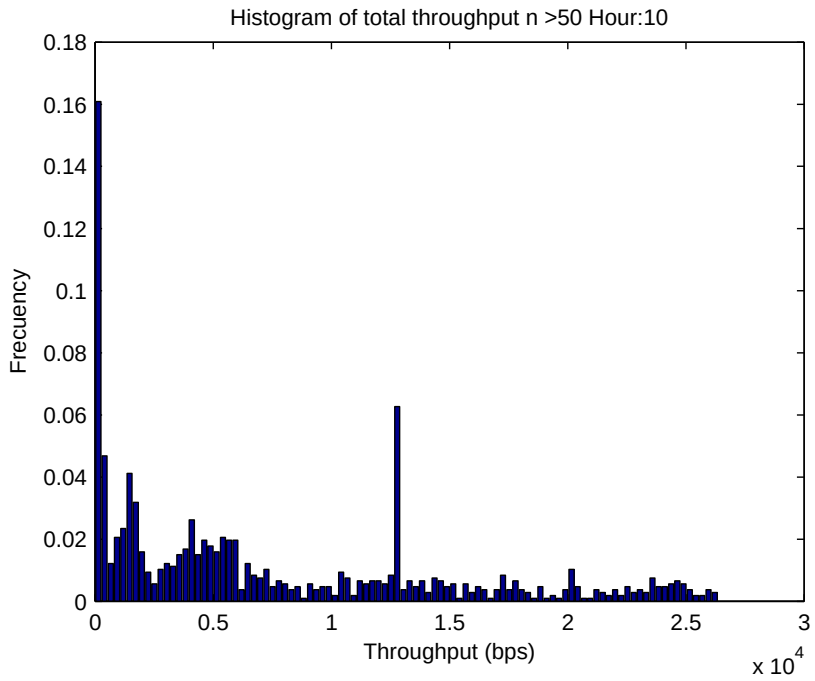

Figure 4.7: Histogram of the mean value of the throughput both for TCP and UDP flows, representing flows with more than 50 packets and average throughput smaller than $2.6 \cdot 10^4$ bps for the time interval 10:00-11:00.

reasonable distribution for the mean value of the throughput, and then apply inference assuming such distribution. Therefore, we supposed a normal distribution for the mean value of the throughput, and applied the corresponding calculation of confidence intervals for the mean assuming this distribution. The normality supposition is reasonable because the throughput may be caused to many independent effects: congestions control, packet loss, number of packets per flow, etc. In addition, the mean value is a weighted sum of the instantaneous values of the throughput. Such instantaneous values of the throughput can be assumed to follow the same (unknown) distribution. Therefore, applying the Central Limit Theorem (CLT), we can conclude that in the limit this weighted sum will converge in distribution to a normal distribution. As we are only taken a small realization, we cannot rigorously apply the CLT. However, the knowledge that in the limit the mean values of the throughput will follow a normal distribution let us assume reasonably the normality assumption and apply the confidence interval computation assuming this distribution.

The $1-\alpha$ confidence interval $I_{1-\alpha}$ for the mean of a normal distribution when its variance is unknown is given by the following equation

$$I_{1-\alpha} = (\bar{x} \mp t_{n-1;\alpha/2}\frac{s}{\sqrt{n}}), \tag{4.2.1}$$

where $\bar{x}$ is the sample mean of the observations, s is the square root of the sample variance, n is the number of observations and $t_{n-1;\alpha/2}$ is the $1-\alpha/2$ percentile of a Student's t-distribution with $n-1$ degrees of freedom.

We applied equation to our hourly based analysis of the throughput and computed the hourly confidence interval for the mean of the mean values of the throughput. The obtained results are

presented in Figure 4.8 for both TCP and UDP flows. We have plot the throughput in the x axis and the hour in the y axis. The hour -1 is equivalent to the time interval 23:00-00:00 of the previous day, the hour 0 is equivalent to the time interval 00:00-01:00 of the analyzed day, and so forth.

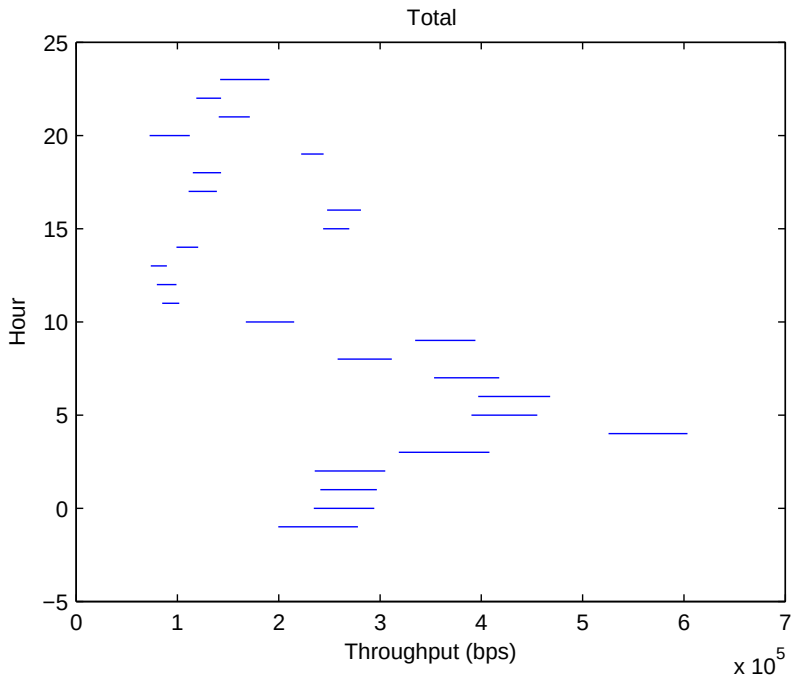

Figure 4.8: Hourly confidence intervals for the mean of the mean value of throughput for both TCP and UDP flows.

The analysis of the hourly confidence intervals was motivated by the following reasoning. If there are no problems in the link, the flows that are sending information will reach high stable values for its throughput, and these high stable values under good performance will be similar between different flows. Therefore, if we compute confidence intervals for the means values they should overlap because all of them should contain the high stable value of throughput, which should have been reached by the majority of the flows due to the good performance of the link.

However, as can be seen in the figure, these confidence intervals do not overlap. The confidence intervals during the night-time are centered at larger values of the throughput than during the daytime. Some studies have shown that there is a clear traffic pattern within the duration of a day, where the traffic load of a link starts growing at 08:00 and decays after 18:00 [TMW97]. With both results in mind, we can conclude that during daytime there is a huge number of users, which share the link's capacity and therefore their communications achieve lower throughput values. On the contrary, during night-time the number of users is reduced dramatically, because they send smaller amounts of traffic at higher rates. This leads to think that when there are fewer flows in the link, they achieve a higher value of the throughput because there is less competition for the resources and the congestion avoidance mechanisms do not apply. In conclusion, the abovementioned high stable value for the throughput is not reached in all the hour time intervals, or at least it is not the same value. Therefore, our approach for assessing link performance based on mean value of

the throughput for all the flows failed to obtain acceptable results.

4.3 Time Series Analysis of the Throughput

The results of the previous section were not conclusive enough to infer the QoS status of the links. The problem was that the overall behavior of the flows was not invariant (i.e. its behavior does not change with time) so the conclusions over a time interval cannot be extrapolated to other intervals. Therefore, we decided to inspect the flows individually, expecting that under good link's conditions its behavior is predictable (i.e. there is a slow start transient period where the rate is increasing and finally stabilizes at a high value).

To perform this analysis, we compute the instant throughput values every time a packet is received as the size of the packet divided by the inter-arrival time. Then, we remove the initial transient period because it masks the rest of the time series. Finally, we depict the corresponding time series, being the time axis the number of packets received instead of the time elapsed since the first received packet (thus we make all the time series plots comparable). In Figure 4.9 we show what we have defined as predictable behavior.

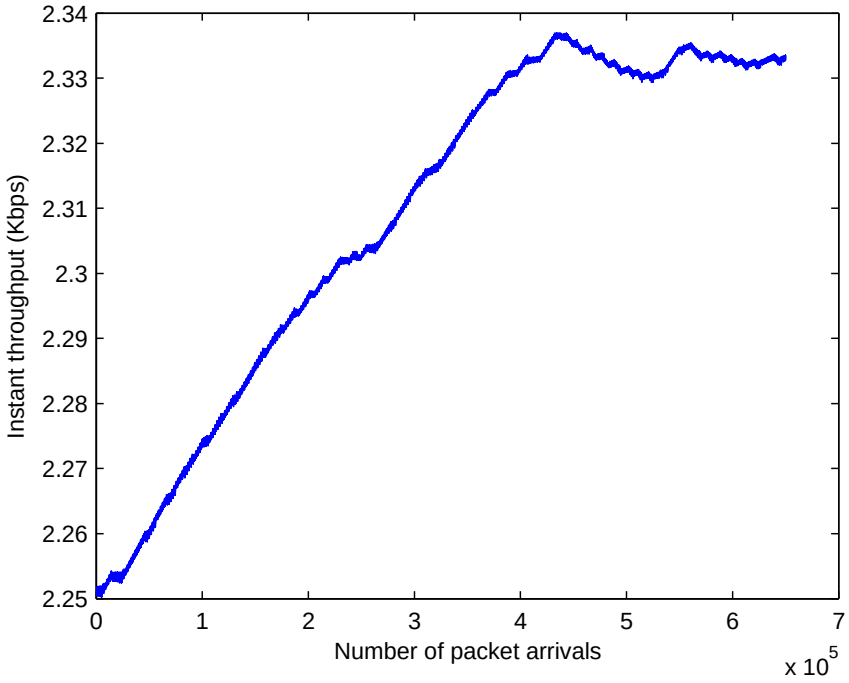

Figure 4.9: Time Series representation of the throughput of a flow with predictable behavior.

Unfortunately, this is not the usual behavior through the analyzed flows. There are flows that begin with a growing period, after which the throughput decreases drastically (Figure 4.10). Other flows never have a growing period, and their instant values of the throughput are always decreasing (Figure 4.11). On the contrary, there are some flows which throughput is always growing, meaning that they never reach a stable limit for the rate (Figure 4.12). Other kind of flows that never reach a stable limit for their rates is shown in Figure 4.13, where we can see that its throughput have several growing and decreasing periods with an overall growing trend.

Therefore, our initial assumption about predictable behavior is not satisfied by the majority of the flows, and we cannot use it to infer the link under analysis status.

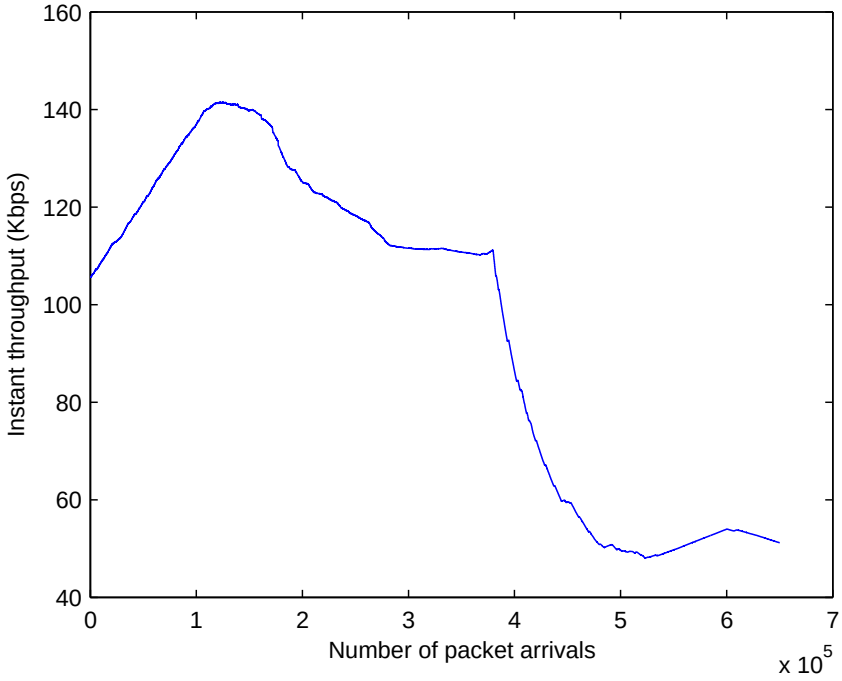

Figure 4.10: Time Series representation of the throughput of a flow which throughput decreases after a growing period.

4.4 Summary and Conclusions

In this chapter we have presented an analysis of the throughput for the flows in the AMPATH OC12 packet trace. First of all we analyzed the distributions of the mean value of the throughput at different time scales (one for the whole day and 24 for each hour of the day). We were unable to determine a well-known distribution to fit the obtained empirical ones. However, we present a reasoning based on the CLT to apply confidence intervals inference assuming a normal distribution. Unfortunately, the results were not as expected, and we could not infer the performance level of the link based on the obtained confidence intervals. Therefore, we decided to analyze individually the instantaneous values time series of the throughput, in hopes that a predictable behavior (i.e. a transient growing period after which the rate stabilizes) was achieved by the majority of the flows. However, the results shown that this predictable behavior is rarely obtained, being the common situation different combinations of growing and decreasing periods without reaching a stable value.

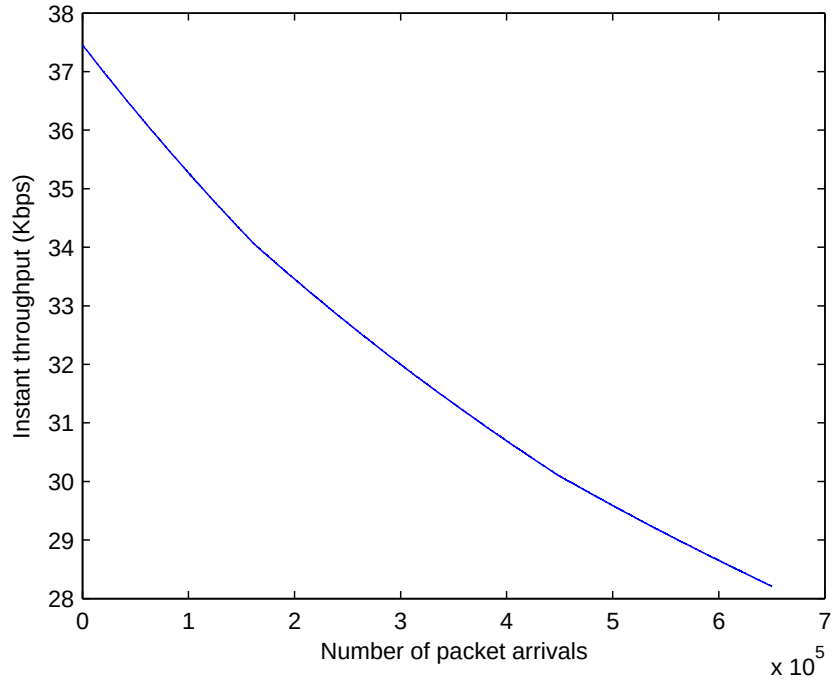

Figure 4.11: Time Series representation of the throughput of a flow with decreasing behavior.

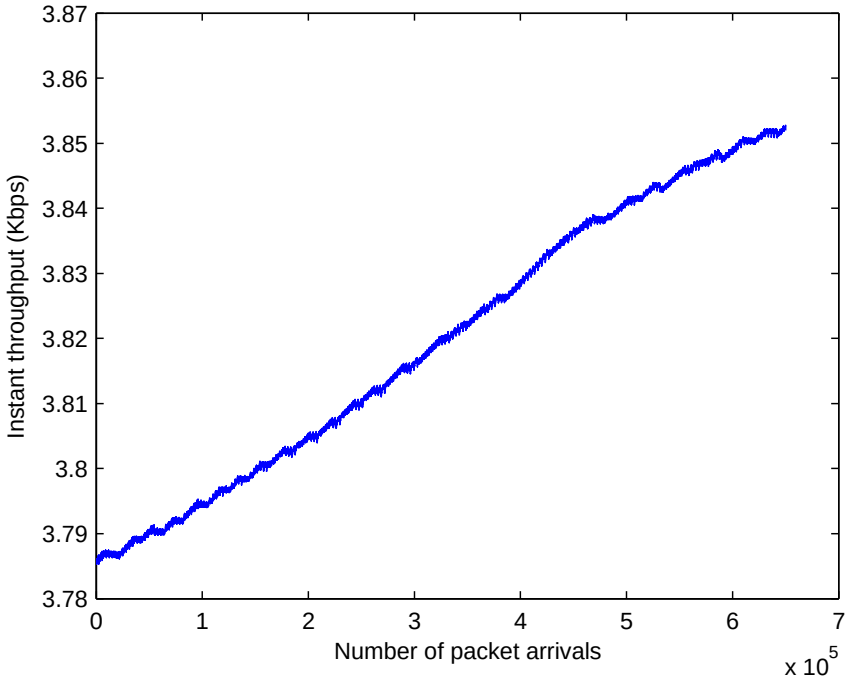

Figure 4.12: Time Series representation of the throughput of a flow with growing behavior.

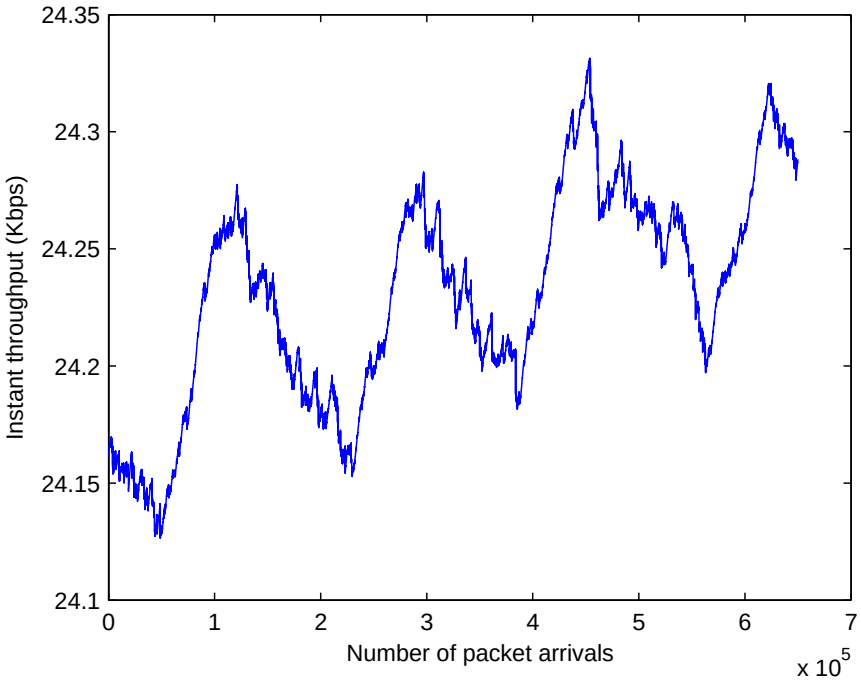

Figure 4.13: Time Series representation of the throuhgput of a flow with several transitions between growing and decreasing periods.

Chapter 5

Multivariate Normal Model for Daily Traffic

The results of the throughput analysis were not conclusive, so we were not able to apply them in order to detect anomalies or misbehavior in the network links. However, when we inspected the confidence intervals we observed that the center of the intervals follows a clear day-night pattern as the bandwidth consumption [TMW97]. Contrary to what could be expectable, the day-night pattern of the throughput has its peaks when the day-night pattern of the bandwidth has its off-peaks hours and vice versa. This led us to think that the day-night pattern was indeed an invariant, because there have been several reports using different datasets but obtaining similar day-night patterns, and therefore it was a suitable measure to obtain footprints. Motivated by this reasoning, we analyzed the utilization day-night pattern of MRTG RedIRIS measurements (described in Section 5.1). Note that the utilization is just the consumed bandwidth scaled by the total bandwidth. The results of this study are presented in Section 5.2. These results were conclusive enough, showing that the day-night pattern of different working days was similar, whereas it was very different when compared with the day-night pattern of the holidays (note that RedIRIS is an academic network) that actually were similar amongst themselves. Therefore, we developed a network traffic model bearing this in mind. This model divides the 24 hour period of a day into disjoint intervals, and obtains the average of the transfer rate within each interval. All the measures from one day are treated as a multivariate sample from a multivariate normal distribution. The details of the model are presented in Section 5.3. Finally, a validation of the model is presented in Section 5.4, after which Section 5.5 summarizes and concludes the chapter.

5.1 Description of the MRTG Measurements

The data used in this study are MRTG [OR98] records of different links within the Spanish National Research and Education Network (NREN) RedIRIS. There are MRTG records for the traffic traversing the incoming and outgoing interfaces of several Point of Presences (POPs) and the access routers of some universities within the RedIRIS network. In total, there are measurements for 23 different network devices that we treat as different links. The MRTG records are extracted

with a granularity of 5 minutes, i.e. every five minutes a new record is output. Each record has five different values. The first one is the UNIX time of the measurements, that will be used in the preprocessing step described in Section 5.3. The next ones are the average and maximum transfer rates, in bps since the last record, for both interfaces. As we know the values for the link capacities for all the measured links, we transform these measurements into utilization values, just dividing each record by the capacity in bps. With this time granularity, we have 288 records per day and direction. Our measurements span from the 2^{nd} of February 2007 to the 31^{st} of May 2008, which leads to 485 days worth of data per link.

5.2 Analysis of the Day-Night pattern of the Traffic Rates

In this section we analyze the MRTG measurements described in the previous section, graphing the values of selected days, in order to better understand the behavior of the daily and weekly patterns in the RedIRIS network. We first graphed the same day of the week for different weeks, hoping that the day-night pattern is more or less the same in all the days. The following graphs confirm our intuition. Figure 5.1 shows the incoming traffic for 6 consecutive Mondays (being each line a different day). As can be seen in the figure, the peak and off-peak pattern is close similar to the one described in [TMW97]. We see in addition that there are some days with bursty values. This could be due to measurement errors, so an averaging process is encouraged in order to reduce the impact of these peaks.

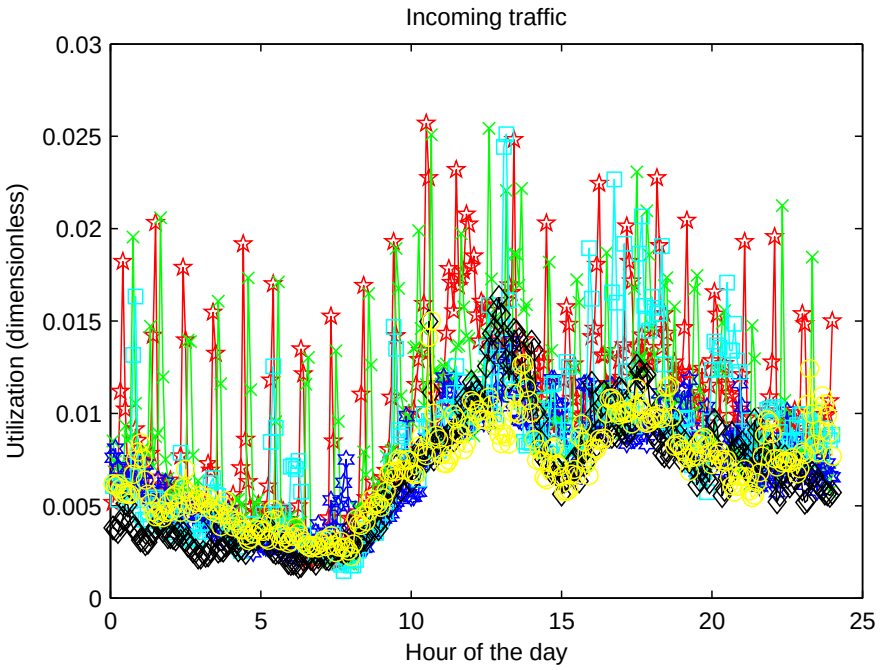

Figure 5.1: Time Series representation of the utilization of a RedIRIS link for several consecutive Mondays.

We repeated this representation for other working days (Figure 5.2 for outgoing traffic of Tuesdays and Figure 5.3 for outgoing Wednesdays) and non-working days (Figure 5.4 for incoming

traffic of Saturdays and Figure 5.5 for incoming traffic of Sundays). The other days or directions not shown in this document are available upon request to the author.

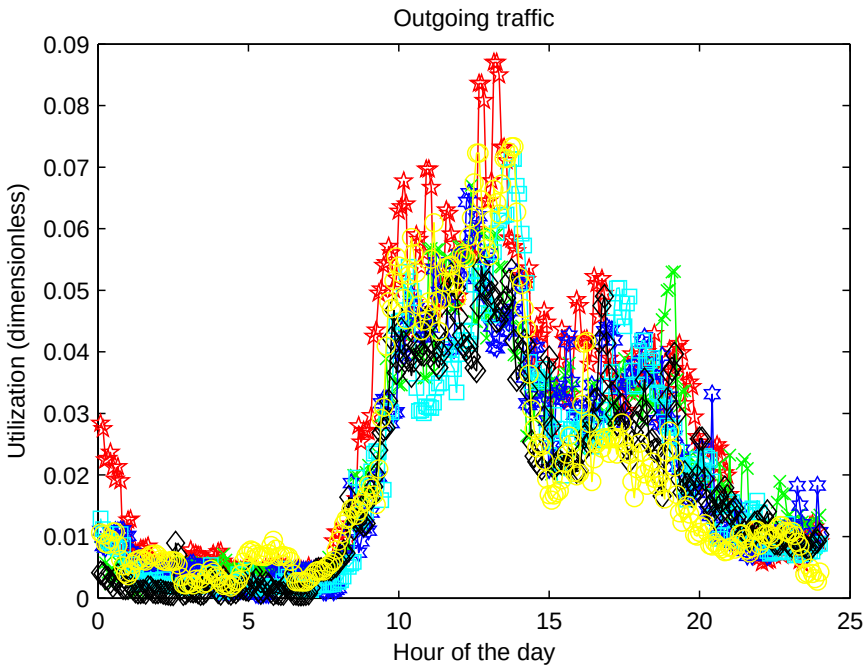

Figure 5.2: Time Series representation of the utilization of a RedIRIS link for several consecutive Tuesdays.

The figures for the working days are similar amongst them. All of them show the day-night pattern, having the peak and off-peak periods in the same intervals. On the contrary, the non-working days show a very different pattern. The utilization is almost flat during the whole day, although the bursty behavior is also obvious during the weekend. The differences are better shown in Figure 5.6, where we have graphed a whole week.

We can see in this figure that the utilization during the weekend (the bottom two time series) is considerably lower than the utilization during working hours in working days (the five top time series), but is nearly the same during the night-time in all the weekdays. This leads us to think that the traffic during night-time corresponds to applications that are left running and generating traffic without user interaction. We have also repeated this procedure but selecting days of different weeks, in order to avoid any possible correlation between days of the same week. The obtained results are close similar to those of Figure 5.6, and are not presented here for the sake of brevity. We therefore can conclude that the day-night pattern is similar between working days, but that there is no day-night pattern during weekends, where the traffic is nearly flat. This conclusion will be applied in the following section, where we present our model for the network traffic.

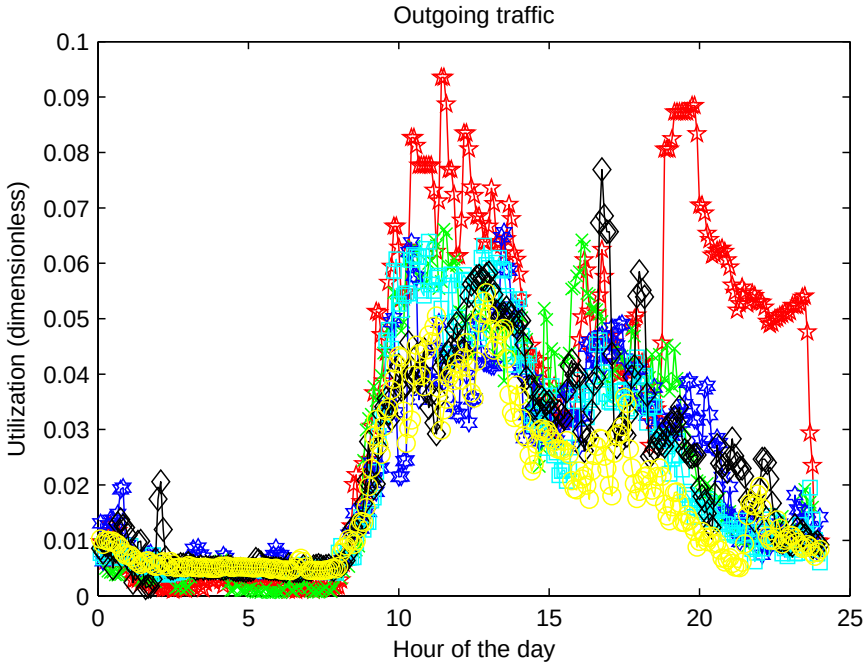

Figure 5.3: Time Series representation of the utilization of a RedIRIS link for several consecutive Wednesdays.

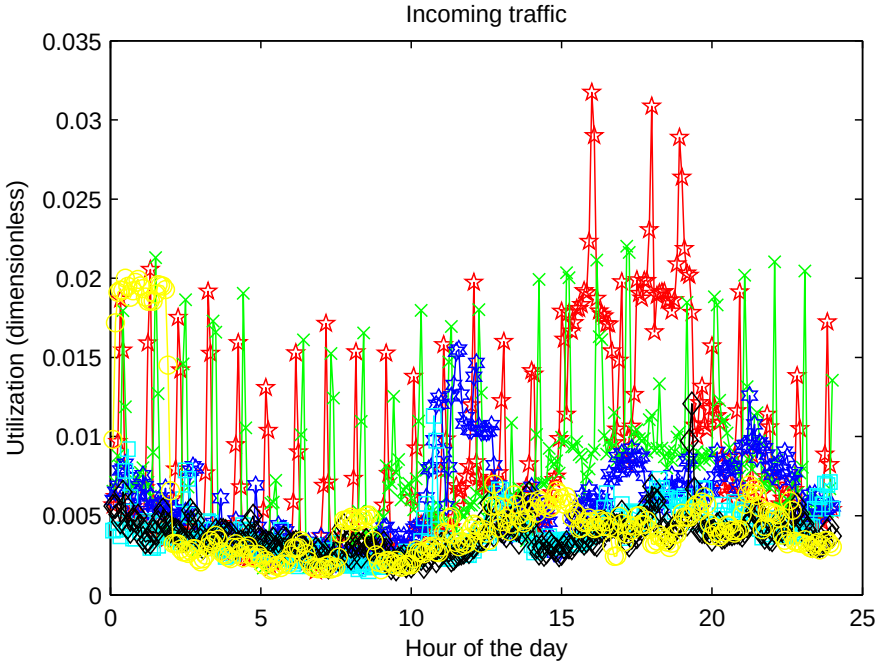

Figure 5.4: Time Series representation of the utilization of a RedIRIS link for several consecutive Saturdays.

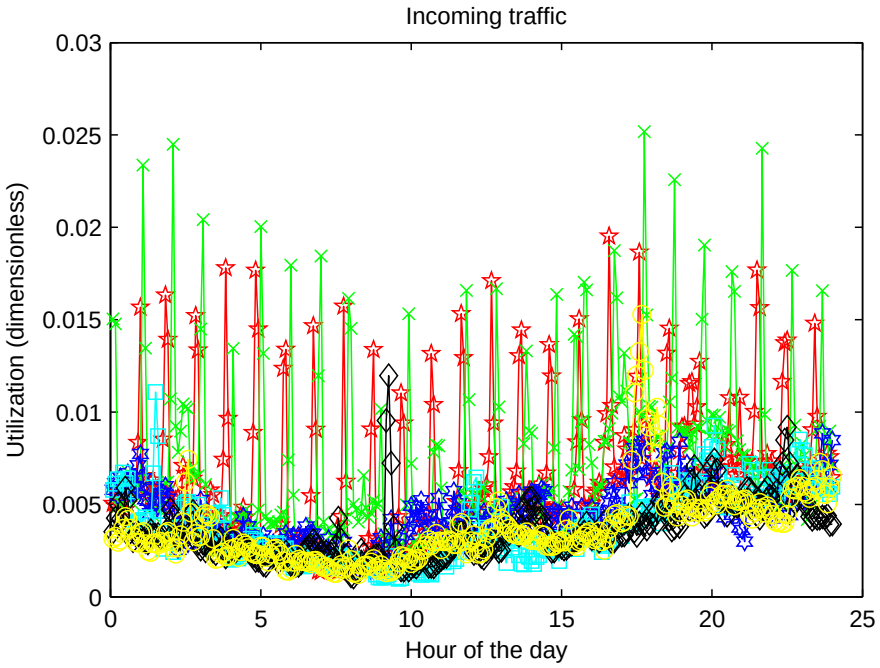

Figure 5.5: Time Series representation of the utilization of a RedIRIS link for several consecutive Sundays.

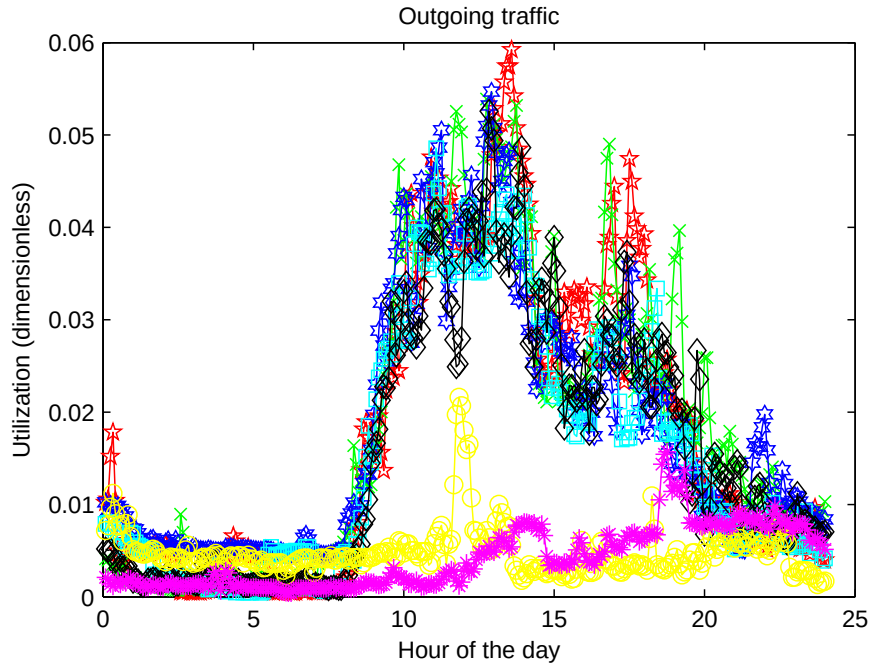

Figure 5.6: Time Series representation of the utilization of a RedIRIS link for a whole week.

5.3 Description of the Multivariate Normal Model

We describe in this section a new model for network traffic. This model takes into account the observed day-night pattern in the previous section, as follows. As we have shown that the day-night pattern of working days is close similar between them, both in shape and values, we decided to use a multivariate distribution to model the samples obtained from the MRTG records for each day (remember that each day corresponds to 1440 samples per direction). Therefore, the realizations during different days at the same time can be considered to follow the same (unknown) distribution. However, a 1440-variate distribution is not an adequate model, because its high dimensionality makes it difficult to work with.

In order to make the model more manageable, we average the MRTG values in 16 disjoint intervals of 90 minutes. The reasons to choose 90 minutes as the averaging period are manifold: first, there is a slim chance of missing data in the five minutes timescale, which is filtered out by averaging in 90 minute periods; second, the time of the measurements may not be the same in the different POPs due to clock synchronization issues. A timescale of 90 minutes is coarse enough to circumvent this problem (this reason is also pointed out by [PTZD05]); thirds, the averaging process prevents outliers and measurement errors to bias the results; last, but not the least, the assumption of normality for Internet traffic holds when there is enough temporal aggregation of the measurements [KN02, vdMMP06] (this normality is also reinforced by the averaging period thanks to the CLT). Therefore, in addition to simplifying the model, we obtain a reasonable distribution for the averaged samples.

However, as we have seen in the previous section, this model cannot be applied to working days and weekends at the same time, because their day-night pattern differs a lot. Therefore, we focus on working days, as their day-night pattern is of higher interest, and remove weekends from our sample. In addition, we remove potential abnormal data because we do not pursue to detect measurement anomalies. We summarize in what follows the data that is removed from our data set:

- The day to which the measurements are referred was Saturday or Sunday. We remove these days because the day-night pattern is very different to that of the working days.

- At least one of the 90 minutes intervals have no measurements. If so, we have no value for this period, and the day-sample will have missing values. In order to circumvent the possible problems that these missing values can produce in latter analysis, we remove the entire day from the data.

- Summer & Christmas Holidays. During summer and Christmas holidays the pattern of usage of the network resources is comparable to the pattern of weekends. Therefore, we also remove these days from the data. Summer holidays are considered since the 1^{st} of June till the 30^{th} of September. Christmas holidays are considered since 22^{th} of December till 7^{th} of January.

- National & Autonomous Community (AC) Holidays. The national holidays are removed from the data for all the links because those days the centers are closed. If there is network usage during those days, it comes from applications that people let running when they are

not present in their workplace, so they are meaningless for capacity planning tasks. For AC Holidays, those days are removed but only for the links that connect centers from that AC to the RedIRIS network, or links that aggregate traffic from any of those ACs. In contrast, for the links that aggregate traffic for all the ACs, these days are kept (for instance links connecting RedIRIS with the Internet).

- Exams periods. As we are mainly using traffic from universities, exams periods can affect the traffic patterns, because students do not use the network as they do when they have no exams. For this reason we remove as summer holidays the months of June and September. It is not clear what to do about February; the exams in this month do not usually take place in the same periods of time in our university, and we have no clue about how is this carried in other universities. By the time of writing these days are kept in the data.

After the preprocessing step, the data set contains more than 200 samples, each being a day that we model with a p-variate normal distribution, where $p = 16$. Note that this preprocessing step can be done in an online fashion because these days are known in advance. Finally, to facilitate the understanding of the relation of the number of the variable with the time period of the day to which it refers, these associations are presented in Table 5.1.

Table 5.1: Equivalence in time of the variables.

Number of the variable	Time interval	Number of the variable	Time interval
1	00.00-01:30	9	12:00-13:30
2	01:30-03:00	10	13:30-15:00
3	03.00-04:30	11	15:00-16:30
4	04:30-06:00	12	16:30-18:00
5	06.00-07:30	13	18:00-19:30
6	07:30-09:00	14	19:30-21:00
7	09.00-10:30	15	21:00-22:30
8	10:30-12:00	16	22:30-00:00

5.4 Validation of the Model

In order to validate the model, we have performed several verifications of the normality assumption for the 16 variables of our model. Therefore, we have applied several univariate normality tests. In addition to this, we have tested for multivariate normality. This is necessary because the fact that several variables have univariate normal distributions does not imply that its tuple has joint normal distribution [JW92]. In what follows, we present the normality tests applied for both univariate and p-variate distributions and the obtained results.

5.4.1 Univariate Normality Tests

We have applied three tests that are available in the statistic toolbox of Matlab for testing univariate normality. These tests are the Kolmogorov-Smirnov (KS), Lilliefors and the Jarque-Bera (JB) tests. A brief description of these tests is presented in Appendix C. We applied these tests independently to samples of the outgoing and incoming directions. First of all, we tested the

overall dataset for univariate normality. The conclusion for both directions in all the links tested is that the null hypothesis of normality must be rejected. This does not necessary mean that the normality assumption is not valid. The normality rejection can be due to variations with time of the distribution parameters. This possibility motivated us to perform the change point detection analysis that is presented in Chapter 6.

To circumvent the possibility that the normality assumption is rejected because its parameters are changing with time, we divide the whole dataset in disjoint subsets of contiguous day-samples, and perform the normality tests to these day-samples. The duration of these subsets is sufficiently large to perform the normality tests but smaller enough to avoid significant changes in the distribution's parameters. In total, we performed more than 300 normality tests per direction, and the percentages of rejections of the assumption of normality with a confidence level $\alpha = 0.01$ for each variable are presented in Table 5.2 for the incoming direction and in Table 5.3 for the outgoing direction.

Table 5.2: Percentage of rejections of the normality assumption per variable in the incoming direction.

Number of the variable	KS Test	Lilliefors Test	JB Test
1	0.66	18.42	16.78
2	1.32	26.64	20.07
3	3.29	30.26	21.71
4	3.95	31.25	23.36
5	3.95	30.26	24.67
6	3.62	21.05	21.05
7	0.66	18.75	18.09
8	0.33	15.46	19.08
9	0.66	17.76	18.75
10	0.66	18.42	21.38
11	0.66	17.76	12.5
12	0.99	16.78	13.49
13	0	18.09	13.49
14	1.32	16.12	16.45
15	0.66	19.08	15.79
16	0	15.13	16.45

The results of the KS tests are very good. However, as it is described in Appendix C.1, when the parameters are estimated from the sample, the results of the KS test tends to be conservative. Therefore, we should focus on the results from the Lilliefors and the JB tests. These are quite similar for all the variables, and its maximum value is near 30%. This means that for that variable, 30% of the times the test was performed, the null hypothesis of normality was rejected. As we are treating with real world measurements, it is expectable to have deviations from normality of some samples that are detected by the normality tests. Therefore, a rate of acceptance greater to 70% is a value large enough to assume the model is appropriate.

In addition to the normality tests, we have also inspected the normality visually. To do this, Quantile-Quantile (Q-Q) plots are commonly used. Q-Q plots are plots were the quantiles of the sample data are plotted versus the quantiles of the distribution the data are supposed to come from (i.e. a normal distribution). If the assumption of normality holds, the quantiles are nearly

Table 5.3: Percentage of rejections of the normality assumption per variable in the outgoing direction.

Number of the variable	KS Test	Lilliefors Test	JB Test
1	0.33	18.57	20.52
2	1.63	22.80	21.17
3	1.95	23.13	21.50
4	2.28	25.73	24.76
5	2.61	23.45	28.01
6	0.65	17.26	20.52
7	0.65	11.73	18.24
8	0.65	18.24	20.52
9	1.30	16.94	23.45
10	0.98	19.22	22.48
11	0	15.96	15.64
12	0.98	15.64	12.70
13	1.30	18.24	14.98
14	0.65	16.61	14.98
15	1.30	21.5	22.15
16	2.28	23.13	25.73

aligned in a straight line, although in the tails of a Q-Q plot small deviations from the line can be accepted. We present in Figure 5.7 and Figure 5.8 the Q-Q plots for two different variables of different directions from the same link where the normality assumption is doubtless exhibited.

5.4.2 Multivariate Normality Tests

Testing multivariate normality is harder than testing univariate normality. There are few analytical procedures for testing multivariate normality and usually the results are obtained through simulations and approximations. However, we can test for normality graphically, in a similar way as was done with the Q-Q plots. To do so, the Mahalanobis distance [Mah36] given by equation (5.4.1) is used.

$$D_j^2 = (x_j - \bar{x})^t S^{-1} (x_j - \bar{x}) \tag{5.4.1}$$

In equation (5.4.1), x_j is the j-th sample of the population, $\bar{x}$ is its sample mean and $\mathbf{S}$ its sample covariance matrix. If the sample comes from a normal distribution, the Mahalanobis distances follow a χ^2 distribution with degrees of freedom equal to the sample size n. Therefore, we can make Q-Q plots of these distances, which are referred as χ^2 plots [JW92]. If the points of this χ^2 plot follow a straight line with slope one through the origin the normality assumption cannot be rejected. We present in Figure 5.9 and Figure 5.10 the χ^2 plots for the incoming and outgoing directions, respectively, of the same link.

These plots evidence that also multivariate can be assumed from the samples of the model. However, we see that the normality assumption is better held by the samples in the incoming direction. This is reasonable, because the amount of traffic in the incoming direction is bigger than in the outgoing one. Therefore, there is more aggregation of traffic in the incoming direction and the normality assumption is better satisfied.

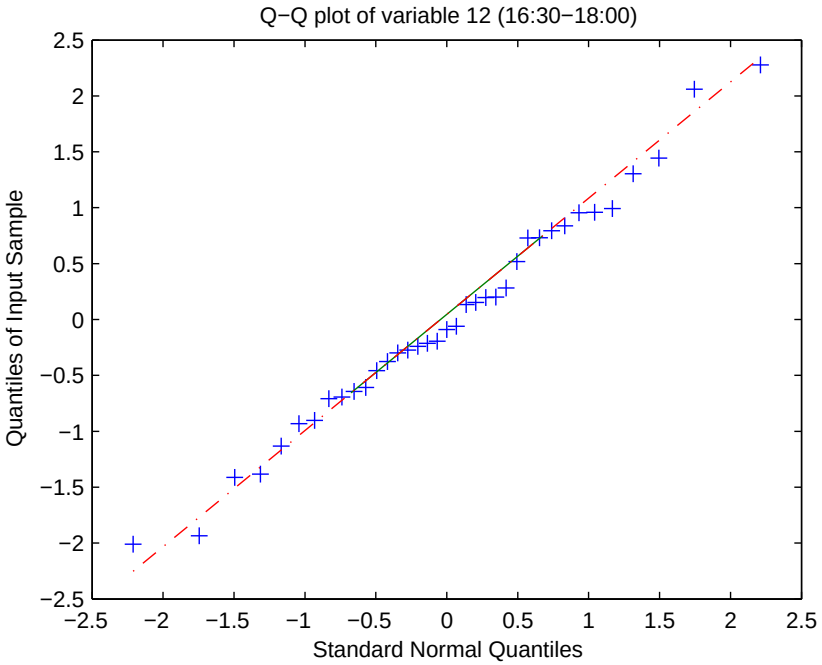

Figure 5.7: Q-Q plot for variable 12 in the incoming direction.

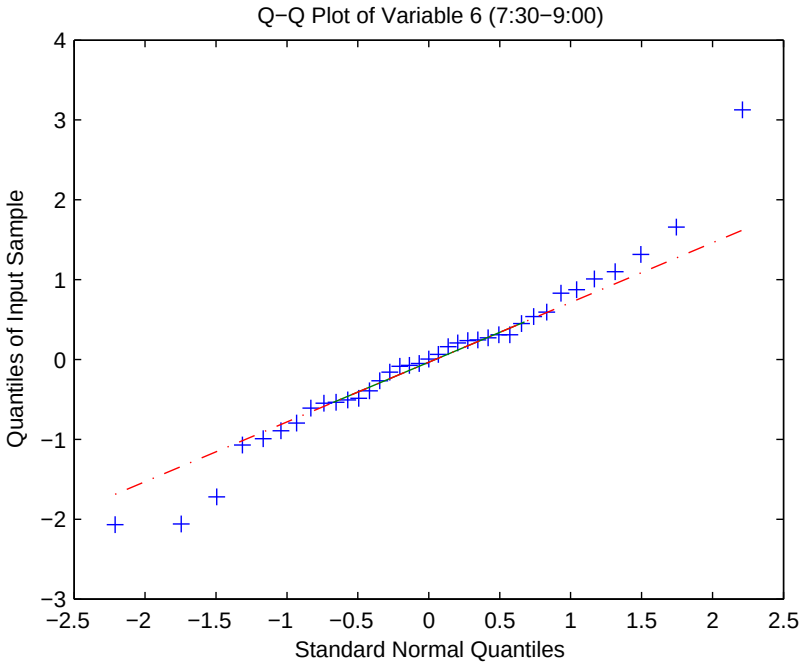

Figure 5.8: Q-Q plot for variable 6 in the outgoing direction.

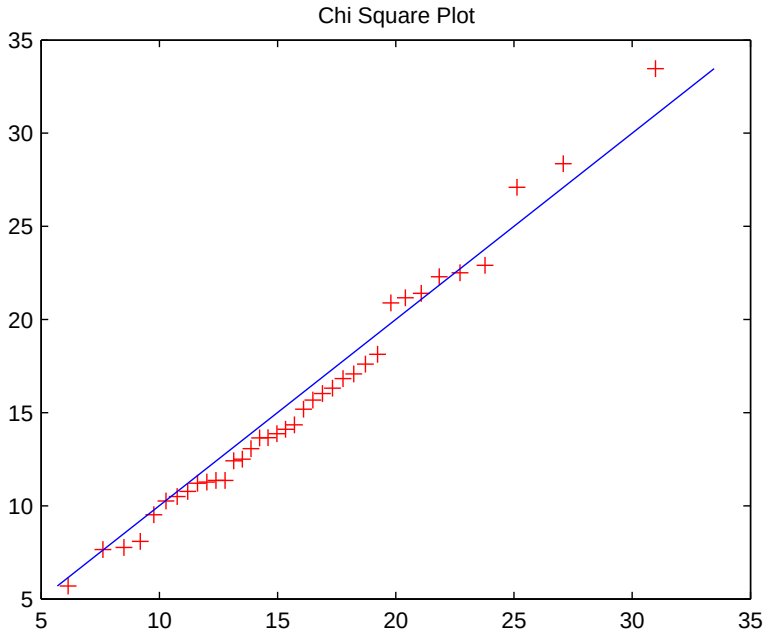

Figure 5.9: χ^2 plot for the incoming direction.

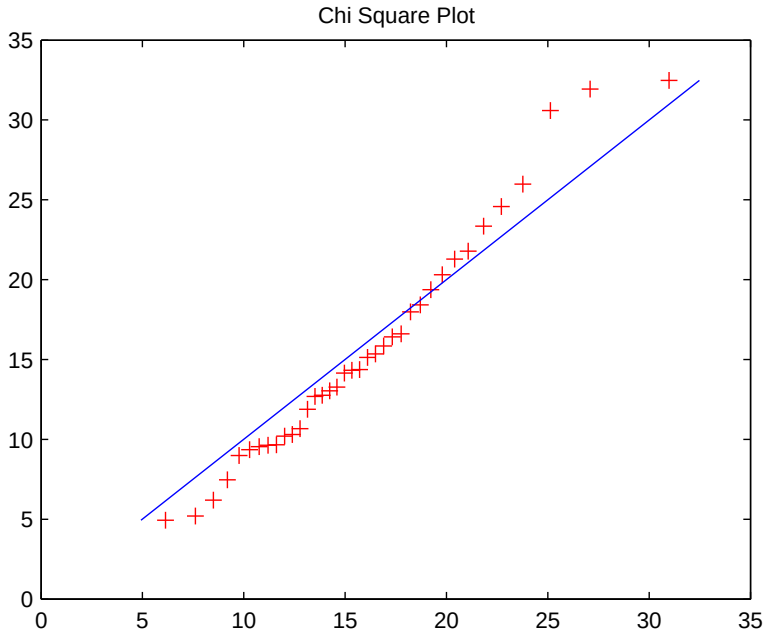

Figure 5.10: χ^2 plot for the outgoing direction.

5.5 Summary and Conclusions

In this chapter we have used a multivariate normal distribution to model the daily traffic. We have used 16 dimensions for our multivariate distribution as we have found this number to be a good compromise between simplicity and detail. The multidimensional nature of the model keeps track of the well-known daily patterns of traffic [TMW97]. We present the reasons that lead us to select a normal distribution for the model, which are based on formerly studies of normality of traffic [vdMMP06, KN02]. Then, we have performed an exhaustive validation of the model, testing for the multivariate normality assumption. However, testing for multivariate normality is not straightforward, and several procedures must be followed. First, univariate normality for all the dimensions should be tested independently. As these tests are applied to independent samples, there is no need to apply corrections for the significances of the test (like the Bonferroni correction). We have performed three different analytic tests for normality, and also assessed it by graphical methods. Once the univariate normality assumption is verified (see Table 5.2 and Table 5.3) it is necessary to test for joint normality, because the univariate normality does not imply joint multivariate normality. Therefore, we computed the Mahalanobis distances for the samples and performed χ^2 plots, that confirmed the multivariate normality assumption. Thus, we have a multivariate normal model which validity has been assessed with powerful and well-known statistical tests. The next step is therefore to apply this model to real network traffic, in order to obtain relevant information from network measurements. This application is described in the following chapter.

Chapter 6

Online Load Change Detection Algorithm

As was pointed out in the validation of the multivariate model, the fact that our hole dataset does not follow a normal distribution can be due changes in the parameters of the distribution. This intuition was reinforced with the univariate normality tests to subgroups of the datasets. This motivated us to develop an algorithm to automatically detect the change points in the datasets, so the results of the algorithm could be applied to network managing. This chapter is devoted to present this online load change detection algorithm, aimed to identify changes in traffic loads when monitoring Internet links. This online change detector was first introduced in [MAGD09] and produces an alert when a sustained and statistically significant change has been detected. Then, the network manager verifies the change and takes action if the change is truly relevant. First, the related works on automated change detection are reviewed in Section 6.1. The algorithm description is presented in Section 6.2. Then, we validate that the behavior of the algorithm with synthetically generated time series, showing the results in Section 6.3 and Section 6.4. This work appears in [MA10]. Following, we apply this algorithm to real network measurements in Section 6.5. Finally, Section 6.6 concludes the chapter.

6.1 Related Work

Change points are defined as the time positions in the original time series where the local trend is disrupted. Mostly, the problem of detecting change points has been tackled by segmenting the original time series data into portions where the parameters of the chosen model remain unchanged. The most naïve models used in segmentation of time series are linear models. With these segmentation models, the time series are divided into piecewise linear segments, and the change points are located in the time instants where the slope of the linear segment approximations changes. However, this kind of approach usually lacks in either good performance or scalability (i.e. it needs all the data in order to find the segments). In [KCHP01] a survey of the different approaches for piecewise linear segmenting is presented, analyzing the aforementioned drawbacks. In addition, the authors present a new algorithm that obtains good performance yet being online

(i.e., not needing all the time series to obtain results). To circumvent this weakness, Guralnik et al. present in [GS99] an algorithm that not only reports changes when the parameters of the model are no longer the same, but also when there is another model more suitable to fit the data (selected from the set of all algebraic polynomials). In addition, more complex models have been also applied to change point detection. For instance, Sharifzadeh et al. [SAS05] use wavelet footprints to detect change points with the same underlying idea of using a polynomial basis, although this approach has the advantage of scaling well to large datasets because of the compression property of wavelets.

However, these fitted polynomial algorithms (and also other model/parameter change detectors such as [PK02]) do not use any knowledge of the process that generate the time series. This means that the performance of change detectors can be enhanced for specific applications by properly applying domain knowledge. Therefore, we apply this domain knowledge modeling the samples with a p-variate normal distribution and focusing on changes in the mean, which are the most significant changes for capacity planning tasks of Internet links. Another main difference between our solution and other existing solutions in the literature is that the Behrens-Fisher procedure, which is applied in our algorithm to verify the change points, is equivalent to inspect for change points in p time series at one time (one for each variable), thus enhancing the change point detection.

6.2 Description of the Algorithm

Our online load change detection algorithm aims to identify sustained and statistically significant change points in network load measurements. Once detected, the change points are reported to the network manager, allowing him to be aware of potential anomalies. The network measurements of interest for the algorithm are load measurements, which can be easily obtained from MRTG [OR98]. We preprocess the measurements in order to obtain day-samples according to the multivariate model presented in Chapter 5.

Our methodology then applies k-means (with $k = 2$) and the Multivariate Behrens-Fisher Problem (MBFP) statistical test in an online fashion, as follows. Every time a one-day measurement is available, it is added to the sample set $\mathcal{S}$. If the cardinal of our sample set is large enough we apply k-means in order to obtain two suitable clusters, i.e. each one with at least 17 samples (we note that we are looking for sustained changes, defining it as change free regions larger than 16 days, so we need $\sharp\mathcal{S} \geq 34$). When we find two suitable clusters, we apply the MBFP statistical hypothesis testing procedure after testing for normality. The MBFP procedure addresses the statistical problem of testing whether the means of two normally distributed populations are the same (null hypothesis H_0), for the case of unknown covariance matrices (more details on the MBFP are presented in Appendix D). When the normality assumption does not hold (i.e. the normality tests reject the null hypothesis) the algorithm still goes on to the following step, and applies the MBFP test to the populations. However, the network manager is warned about this fact in order to not blindly trust the results of the algorithm. Finally, if the MBFP test rejects the null hypothesis, an alert is placed to the network manager that indicates a potential change point, and the oldest cluster is removed from the sample set. The flux diagram of Figure 6.1 summarizes the description of the algorithm.

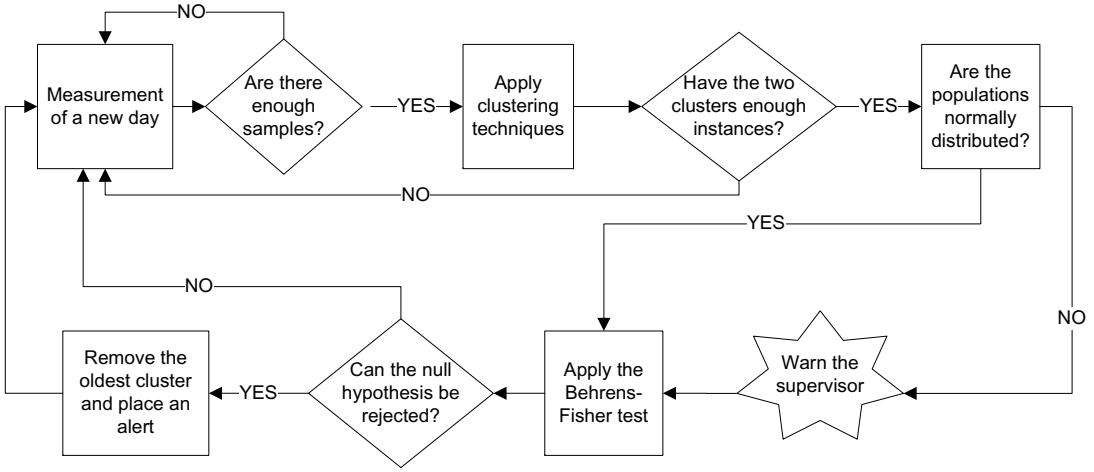

Figure 6.1: Flux diagram of the online algorithm.

6.3 Validation of the Algorithm's Performance

In order to assess the performance of the load change detection algorithm, we have tested it with synthetic data. These synthetic data allow us to verify whether the algorithm is detecting the changes properly. We can do so because we know beforehand where the changes are located. The synthetic datasets generated to test the algorithm can be classified into two different groups, depending on whether they have changes or not. In what follows we describe the datasets generated and show the results of the algorithm performance evaluation. The datasets are N 16-dimensional normal distributed vectors[1], with $N = 9000$, which is large enough to assess the validity of the obtained results (note that a sample of $N = 9000$ is equivalent to analyzing approximately 25 years of data in our algorithm).

6.3.1 Datasets with no changes

We have generated four datasets with no changes, i.e. having all the samples within the dataset the same mean vector. Even in this case, there is always the chance of detecting a change anyway, thus having False Positives (FP) alarms. These FP can be controlled with the significance level α, which is the probability of rejecting the null hypothesis (that is, detecting a change) even though there is no change in the data (Type I Error). The purpose of these datasets is to evaluate the FP rate under no changes, which asymptotically must approach the probability of Type I Error.

$$\begin{aligned} \mathrm{P}(\text{Type I Error}) &= \mathrm{P}(\text{reject } H_0 | H_0 \text{ is true}) = \alpha \\ &= \lim_{M \to \infty} \frac{\#\text{ of rejections}}{M}, \end{aligned} \tag{6.3.1}$$

where M is the total number of tests performed with datasets that fulfill H_0.

[1] all the vector components are independent of each other

Table 6.1: Datasets generated with no changes.

Dataset	Description
All Equal (AE)	All vector components have the same mean and variance
Means (M)	Each vector component has a different mean, but their variances are the same
Variances (V)	Each vector component has the same mean, but different variance
Means Variances (MV)	Each vector component has different mean and variance

Description of the datasets

The four datasets generated without changes in their means are obtained through four different affine transformations on four different random samples of N realizations distributed according to a standard 16-variate normal distribution. The applied transformations have been chosen in order to obtain four datasets with the characteristics that are summarized in Table 6.1.

Results

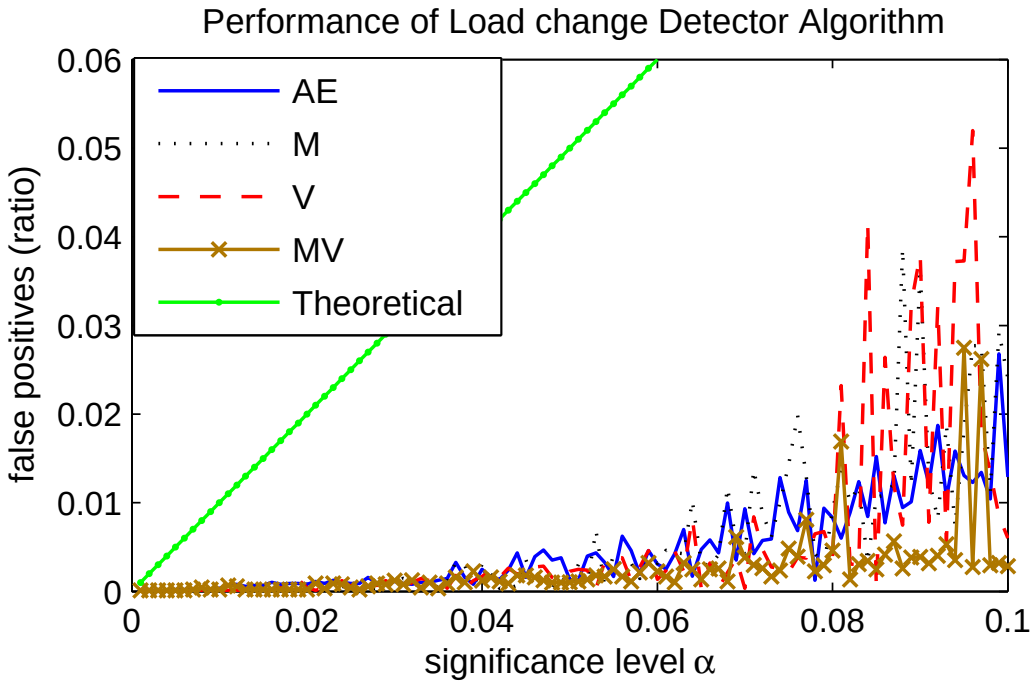

Figure 6.2: False positives ratio in datasets with no changes.

We have measured the False Positives Ratio (FPR) given by (6.3.1) for different significance levels α. The results are presented in Figure 6.2, which shows the FPR of each dataset versus the significance level used in the tests, also with the theoretical FPR (that equals α). The FPR remains almost negligible for significance levels smaller than $\alpha = 0.06$. Thus, we have a large interval of

possible significance levels with good performance. Significance levels above 0.06 experiment an increment in the FPR, but also in this region the FPR of the algorithm when applied to these datasets is smaller than the theoretical one. The differences in the performance of the algorithm for the four different datasets are not relevant, because these differences are mainly due to random number generation issues (we have confirmed this by applying different transformations to the same random generated sample).

6.3.2 Datasets with staggered increments

As the aim of the algorithm is to detect changes in the load, after confirming that there is a low ratio of FPs, a validation with controlled changes follows. Thus, we have generated two different datasets with staggered increments of duration one and three months, i.e. the distribution of the samples remain the same for one (three) month(s), and after that, the mean is increased. We note that this kind of growth is the most significant for the capacity planning task, because linear increments are easily tracked by classical time series analysis, so a forecast of upgrading times when the changes are linear is straightforward. This is accomplished by fitting a time series model to the data (for instance an Auto Regressive Integrated Moving Average (ARIMA) model [PTZD05]) and then predicting when the time series will be above a given threshold ([BD91]) where the QoS of the link might be compromised. Therefore, detecting staggered increments in a timely fashion is crucial for network operators, because the reduction in QoS delivered to its customers adversely affects the operator's reputation.

Description of the datasets

The growth rate for the monthly staggers is chosen such that effective annual growth is around 90%, which is in accordance with popular reports about the Internet traffic growth ([Odl03]). Thus, the monthly growth is approximately 6%. The quarterly growth has also been set to approximately 6%, on attempts to make the obtained results comparable, i.e. we have longer periods without changes in the quarterly growth dataset, but the size of the staggers (which are the relevant facts to detect changes) are the same in both time series. Finally, the theoretical number of changes that should be detected with the algorithm in the Monthly Increments (MI) dataset is 300 and in the Quarterly Increments (QI) dataset is 100.

Results

In Figure 6.3 we show the number of detected changes on the MI data as a function of the significance level of the performed tests. This figure shows very promising results. The number of detected changes is in the range 295-300, while the correct value is 300. In addition, the number of false negatives is small for all the significances tested.

Figure 6.4 presents the same information but for the QI data. Here the performance has been reduced. There is no significance level at which we detect exactly the same number of changes that are theoretically in the dataset. In addition, the false positives have enlarged, being now greater than 50. Above significance values greater than 0.06 we detect more than 300 changes, meaning that every theoretical change we alert for 3 detected changes. We will shed light on the causes of this misidentification in Section 6.4 by inspecting the results at a fixed significance level.

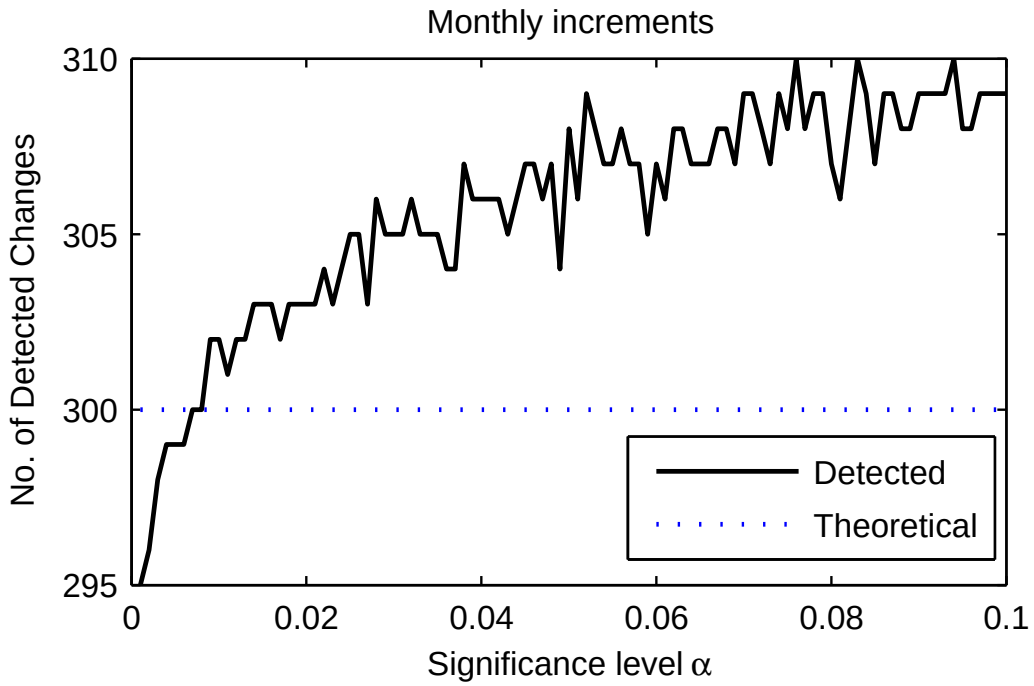

Figure 6.3: Detected changes in Monthly Increments dataset.

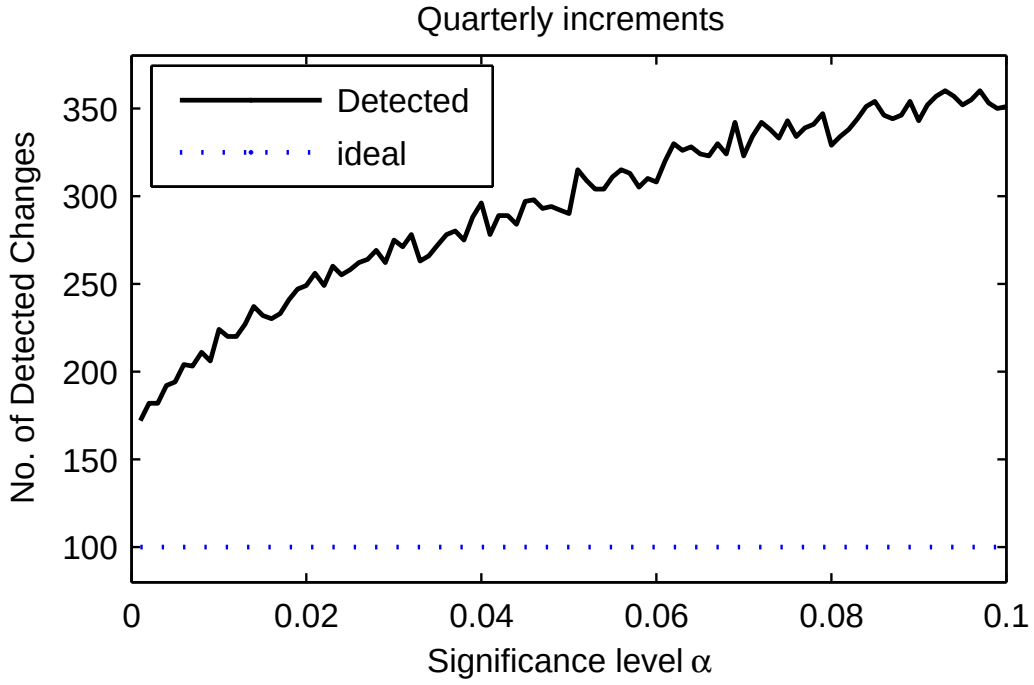

Figure 6.4: Detected changes in Quarterly Increments dataset.

6.4 Analysis of the Validation Results at Fixed Significance Level

In this section, we further inspect the synthetic data presented in the previous section, but with a fixed value for the significance level. The value selected for the significance level is $\alpha = 0.05$, as it is the most commonly used value. By making the significance level fixed we can apply analysis of the Hotelling's T^2 statistic presented in Appendix E. In addition, we can present graph plots of the clusters found and inspect the reported change points. On those graphs, we plot the values of the projection in one vector component, using different color-marker schemes to differentiate the change free regions according to the results of the algorithm. In addition, we mark with a straight line the mean of all the values within a change free region, making it easier to judge the validity of the reported change points. As the amount of points generated for each vector component is humongous, we will focus on certain regions of the plots that we have found to be relevant for the validation.

6.4.1 Datasets with no changes

This subsection is devoted to inspect the datasets generated with no changes. In what follows, we focus on the All Equal (AE) dataset, as we have found it to be representative of all the datasets generated with no changes.

In Figure 6.5, we show the change free regions found by the algorithm in the first 300 samples of the AE dataset. Although the samples are concentrated around the true mean (100), the algorithm detected some change points. This happens because we are applying a statistical test, whose confidence level can be interpreted as the rate of false positives in the limit. Therefore, although a perfect algorithm would have detected no changes in this dataset, it is a normal situation when applying statistical tests to have some FPs due to the confidence level.

The change points reported by the algorithm in this dataset can be due to the following reasons:

- The algorithm found one cluster with mean above the theoretical followed by a cluster with mean under the theoretical (or vice versa). This can be easily seen between the first two change free regions in Figure 6.5.
- The weighted sum of the differences in all the vector components is above $F_{p,N-p}^{1-\alpha}$ (Appendix E). To illustrate this fact, we present in Figure 6.6 the same zoom area for vector component 2. The differences between the last two change free regions on Figure 6.5 and Figure 6.6 (the dots ($\cdot$) around sample 200 and the circles ($\circ$) on its right) are very small, but the addition of these differences through all the variables motivates reporting a change point.

6.4.2 Datasets with staggered increments

As was described in Section 6.3.2, these datasets are designed to be invariant both in mean and variance for a fixed period of time after which the value of the mean is increased. Thus, in these regions without changes we are in the same case as in the AE dataset. We therefore inspect each stair of the dataset from the point of view used in Section 6.4.1.

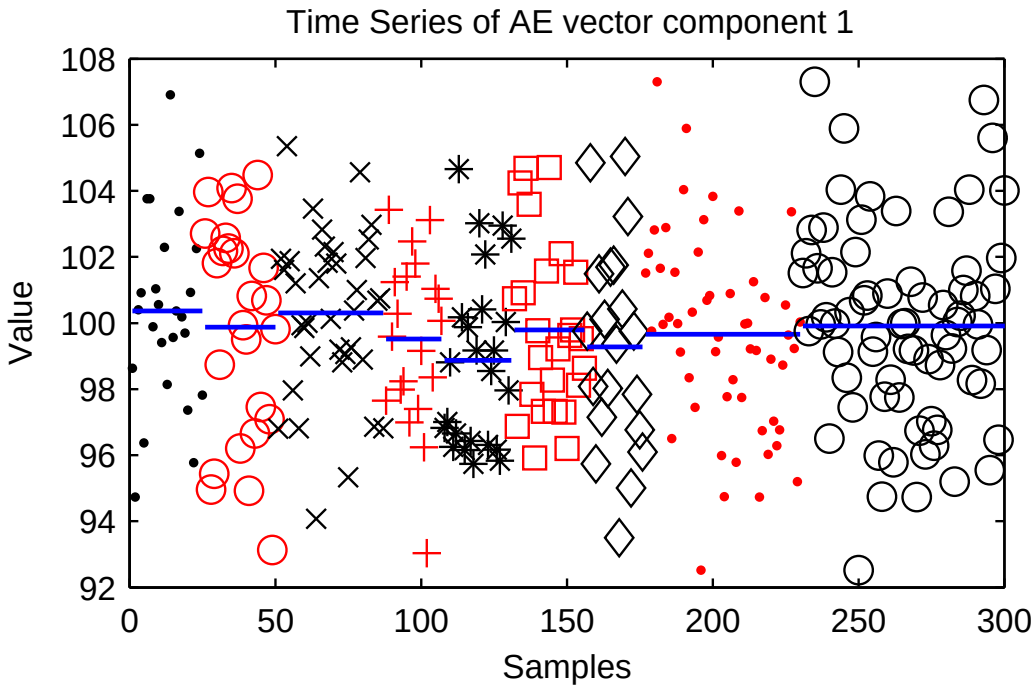

Figure 6.5: Time Series representation of the change free regions for the first 300 samples of the 1^{st} vector component of the AE dataset.

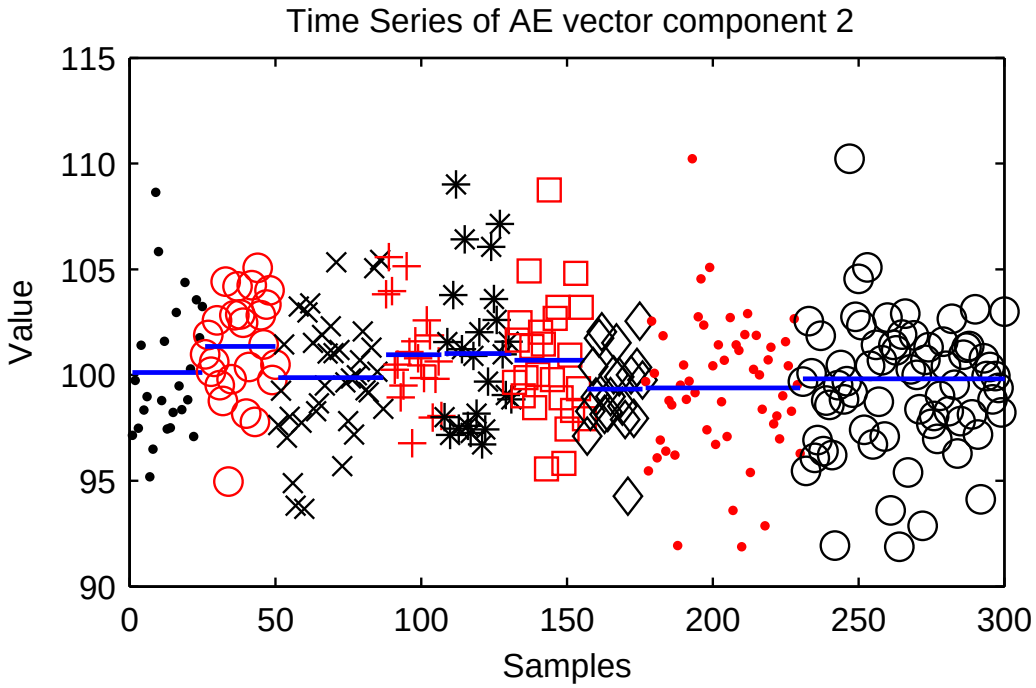

Figure 6.6: Time Series representation of the change free regions for the first 300 samples of the 2^{nd} vector component of the AE dataset.

Monthly increments dataset

The clusters in the final samples of this dataset (sample 8000 and above) are easily identified by the algorithm, as the differences between those clusters are big enough due to the increment by percentages in each theoretical change point. Thus, we will zoom in the beginning of the dataset and focus on the first samples (sample 120 and under). This region is depicted in Figure 6.7, where we have placed vertical lines in the time instants where the theoretical change points are located.

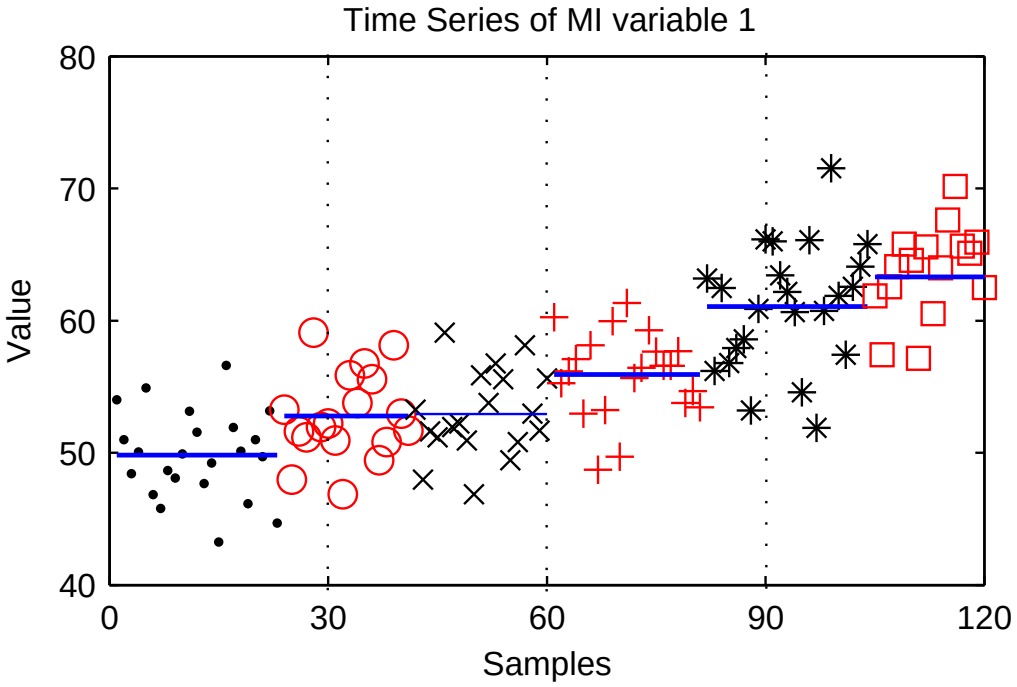

Figure 6.7: Zoom to the first 120 samples of the 1^{st} vector component of the MI dataset with delimitation lines for the theoretical change points.

As can be seen in the figure, the variance of the sample is big enough to make samples in different theoretical change free regions (therefore with different means) to be indistinguishable in some cases. For instance, take a look in the first change free region (under sample 30). The circle ($\circ$) samples in this region are generated with the same mean as the dot ($\cdot$) ones. However, these circle samples resemble more to those circle samples in the second change free region (between samples 30 and 60) than to the dot ones with the same theoretical mean. This is detected by the algorithm through the clustering technique, which divides the first region before the theoretical change. As the difference between the means is truly significant, the MBFP procedure detects it and a change point is reported between these clusters. That is what makes the algorithm to misinterpret the true change point between those regions, which we have confirmed to happen also in other instants of the dataset. This rationale explains all the false positives detected by the algorithm, that under small variance samples or with a more restrictive significance value would not have been detected. However, if we pay attention to the second change free region, we find that there are not significant differences between the two clusters found by the algorithm when

inspecting them visually. Remember form Section 6.4.1 that the detected change point between these two clusters is also due to the differences in the means of the remaining vector components, although apparently in this component there is no change.

Quarterly increments

In this section, we deal with the staggered synthetic data whose increments happen every three months (90 samples). For the same reason that in the MI dataset, we will zoom in to the first samples, because there the samples are more concentrated and it is difficult to assess the validity of the algorithm without this zoom. In Figure 6.8 we have zoomed in to the first 360 samples, and represented the change free regions found by the algorithm in conjunction with their means and the theoretical change points.

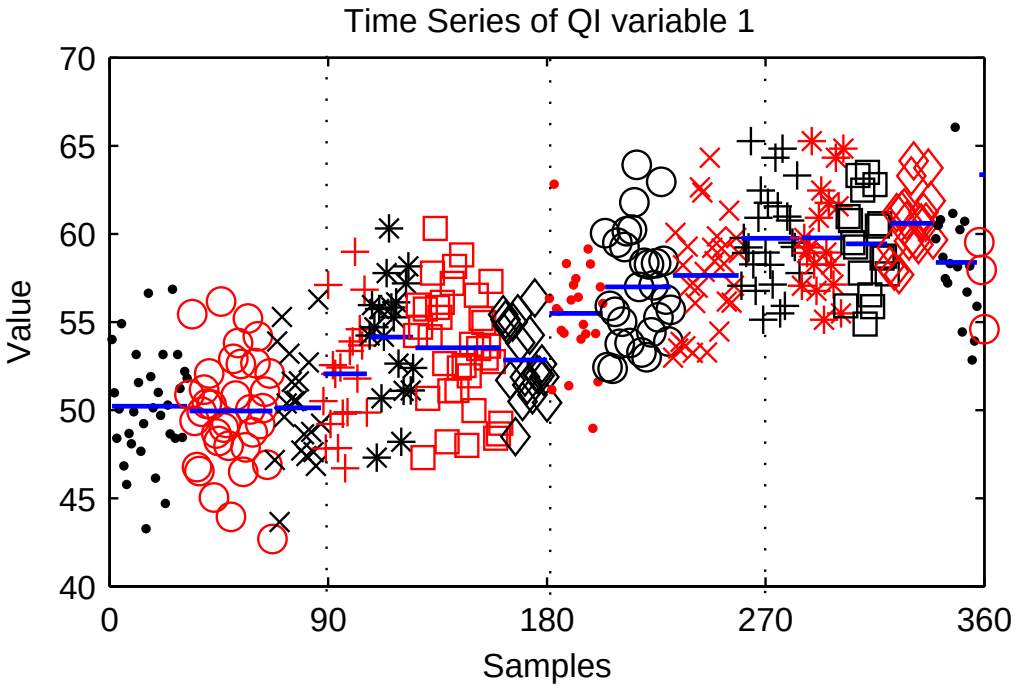

Figure 6.8: Zoom to the first 900 samples of the 1^{st} vector component of the QI dataset.

In that figure it can be easily seen that in each theoretical change free region, our algorithm reported several change points. The reason for the detection of these extra change points is the same pointed out in Section 6.4.1, as the extra change points are detected within a theoretical change free region, where the mean and the variance remain constant (same as AE dataset). On the other hand, there are some theoretical change points not reported by the algorithm (for instance the one in sample 270). The reason for the misidentification of some theoretical change points was described in the previous subsection for the MI dataset. As the samples have a relatively large variance (compared to their mean) in this region, this leads to samples of one theoretical change free region that resemble more to those of adjacent regions than to the samples on its own region. This similarity is detected by the clustering algorithm, and the fact that there is actually a difference between them is finally confirmed by the statistical procedure.

6.5 Change Point Analysis with Real Network Measurements

n this section we present the results of applying our methodology to real network measurements obtained in the RedIRIS network. These measurements follow the description given in Section 5.1, although in this case the measurement period is larger. We have now measurements from 18 links that spans from the 6^{th} of February of 2007 to the 10^{th} of March of 2009. Table 6.2 summarizes the number of tests performed and alerts generated by our algorithm when applied to this dataset. The second and fourth columns show the number of times the MBFP testing methodology is applied. This is the number of times that the clustering algorithm was able to form two clusters with enough size to apply the test. The third and fifth columns show the number of times an alert is generated, i.e. the null hypothesis of equality of means is not verified.

Table 6.2: Results of the online algorithm.

University link	Incoming direction		Outgoing direction	
	Number of tests	Number of alerts	Number of tests	Number of alerts
U1	68	13	76	11
U2	68	12	130	9
U3	62	13	75	11
U4	86	10	57	11
U5	64	11	84	11
U6	56	11	76	12
U7	85	11	89	10
U8	112	10	75	12
U9	79	10	59	12
U10	65	11	102	12
U11	67	11	67	12
U12	103	9	75	11
U13	73	10	84	10
U14	108	10	61	11
U15	98	8	85	9
U16	59	12	57	11
U17	123	10	88	11
U18	82	11	94	13
Average	80.94	10.72	79.67	11.06

As can be seen in Table 6.2, the advantage of our online algorithm to network load detection is that it decreases the Operational Expenditure (OPEX) by reducing the human supervision. We remark that our algorithm produces an alert only in case a stationary change in the load happens. The rest of the time the link is considered normal and no intervention from the network manager is required. Regarding the time span of the measurements, our algorithm placed less than 13 potential network load changes requiring human supervision in a period of more than 750 days (including holidays). That means a potential load change nearly every two months.

To illustrate these results, we present in the following figures the obtained clusters using different color-markers to differentiate them. Figure 6.9 shows the obtained clusters for the incoming direction of university link U1 for the time interval 12:00-13:30 (variable 9) and Figure 6.10 shows

the clusters obtained for the outgoing direction of the same university link and the same time interval.

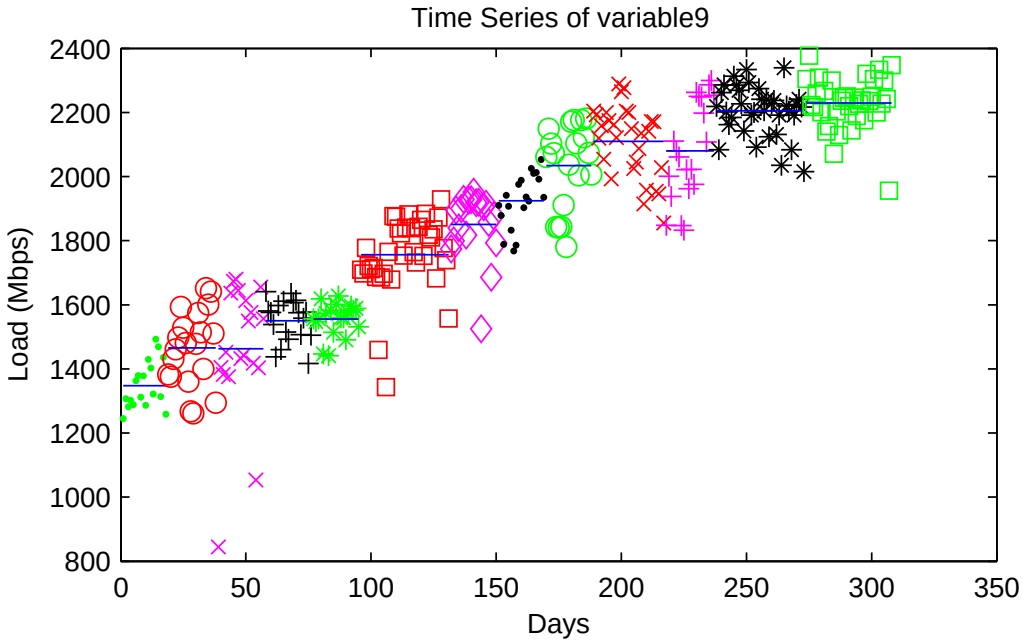

Figure 6.9: Change points found by the online algorithm in the incoming direction of university link U1 on the time interval 12:00-13:30.

Almost all the clusters obtained by the algorithm are reasonable. However, there are some reported clusters that do not seem to have been properly detected. For instance, the change point reported between samples 100 and 150 seems to be a FP. Here is worth remembering the reasoning followed in the validation of the algorithm in Section 6.4. There, it was pointed out that a reported change point can be due to differences in other variables different than the one shown. This is evidenced when comparing Figure 6.10 with Figure 6.11, where we have plotted the same clusters but for the time interval 21:00-22:30 (variable 15). In variable 15 there is actually a noticeably change point between those clusters, which motivates the reported change point by the algorithm although in variable 9 there was not a change point.

Finally, we have used the reported clusters to cross-validate the network model presented in Chapter 5. Thus, we have repeated the univariate normality tests performed in Section 5.4.1 but for the clusters reported by the algorithm. The results are shown in Table for the incoming direction and in Table for the outgoing one.

As can be seen in the tables, the results of the univariate normality tests are slightly worse than when they were applied to fixed length subgroups of the datasets. The reasons to obtain worse results are twofold. First, we are applying now the univariate normality tests to larger populations. Therefore, the normality tests are more powerful, and some cases that were before dubious but finally accepted are now rejected. On the other hand, our algorithm tests whether the means have change or not, but without making any assumption about the variances. However, if the variances are also changing within the reported clusters, the normality tests reject the null hypothesis although the mean remains constant. We plan to enhance our online algorithm by also

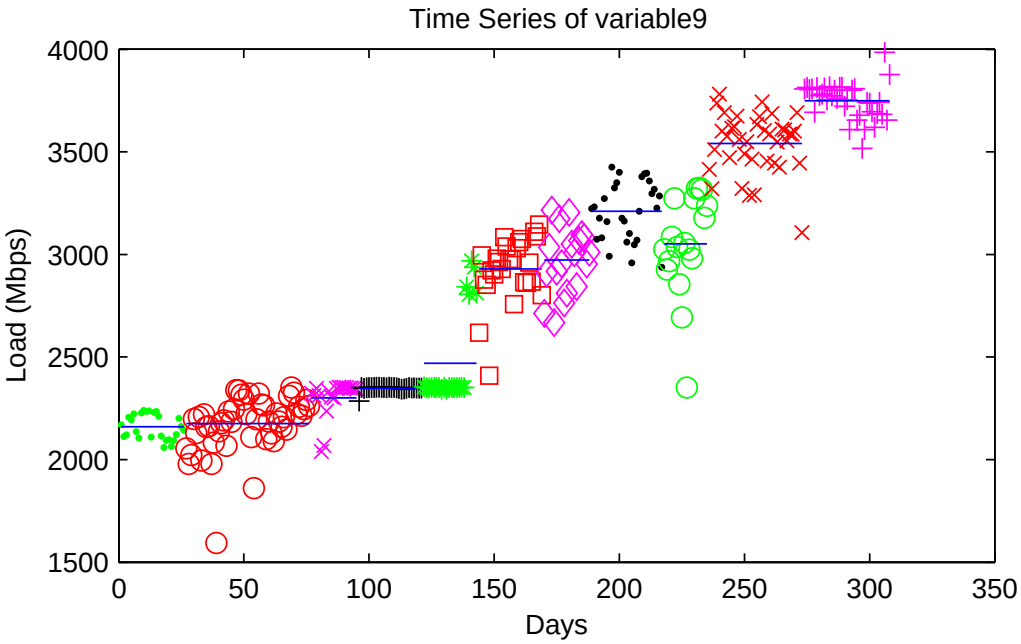

Figure 6.10: Change points found by the online algorithm in the outgoing direction of university link U1 on the time interval 12:00-13:30.

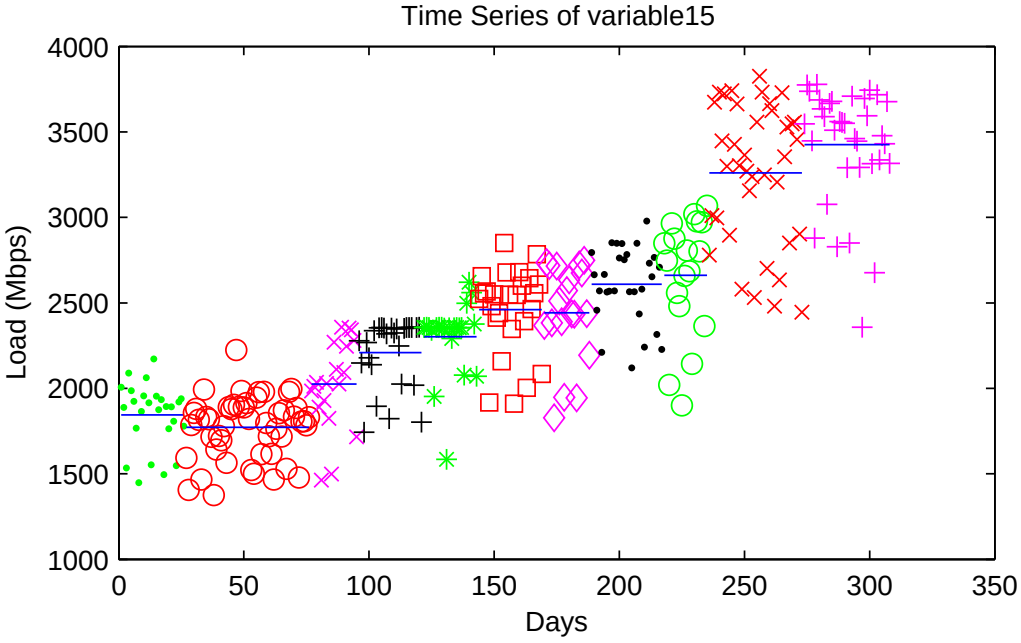

Figure 6.11: Change points found by the online algorithm in the outgoing direction of university link U1 on the time interval 12:00-13:30.

Table 6.3: Percentage of rejections of the normality assumption per variable in the incoming direction for the clusters reported by the online algorithm.

Number of the variable	KS Test	Lilliefors Test	JB Test
1	1.47	25.49	23.53
2	3.92	34.80	26.96
3	6.86	34.80	28.92
4	8.33	34.31	26.96
5	7.35	37.25	29.41
6	5.39	31.37	31.37
7	1.96	25.98	30.88
8	0.98	22.06	24.02
9	0.98	25.49	25.98
10	2.45	21.08	23.53
11	2.45	21.08	19.61
12	1.96	21.57	19.61
13	0.49	22.55	24.51
14	2.45	21.57	22.06
15	1.47	23.04	22.55
16	2.45	19.61	23.04

Table 6.4: Percentage of rejections of the normality assumption per variable in the outgoing direction for the clusters reported by the online algorithm.

Number of the variable	KS Test	Lilliefors Test	JB Test
1	0.48	23.44	26.32
2	3.35	27.27	25.36
3	4.31	27.75	26.32
4	4.31	31.10	30.14
5	5.74	32.54	31.58
6	2.39	16.75	24.88
7	1.91	17.70	24.40
8	2.39	21.05	26.32
9	2.39	22.49	28.71
10	2.39	22.49	29.19
11	0.96	22.49	18.66
12	1.44	20.10	18.18
13	1.91	21.05	19.14
14	1.91	17.22	17.22
15	5.26	26.79	28.23
16	3.83	24.88	27.75

detecting change points taking into account variations in the variance.

6.6 Summary and Conclusions

In this chapter we have presented an online change load detection algorithm aimed to automatically detect change points in the load of the Internet links. This algorithm was first introduced in [MAGD09] and validated in [MA10]. Our algorithm makes use of powerful and well-known statistical tests procedures in order to assess the validity of the clusters obtained by k-means. The results of the algorithm when applied to real network measurements are very promising, as shown in the figures in Section 6.5, and allow a network operator to reduce the OPEX by preventing the network manager to visually inspect continually the time series of the links. The future work will cover the analysis of the obtained change points, in order to model them to allow prediction of future change points.

Chapter 7

Conclusions and Future Work

This master thesis covers the early stages of research work for almost two years. The described contributions follow the investigation of network invariants that could be applied for inference. This research began with the analysis of packet and connection level statistics that were applied to classification of flows by the generating application. This unfruitful classification method moved us to analyze the throughput in a per flow basis. We started analyzing the distribution of the mean values of the throughput, making a distinction whether the protocol under use was TCP or UDP. This distribution did not resemble any well-known distribution, so we decided to suppose normality (that was assumable thanks to the CLT) and compute confidence intervals for the throughput mean value. The underlying reasoning was the assumption that, under adequate link conditions, these confidence intervals must overlap because the mean throughput value is more or less the same. Unfortunately, the results showed that the mean value of the throughput follows a day-night pattern opposite to traffic pattern. When the traffic of the link has a peak, the mean value of the throughput has an off-peak, and vice versa.

This finding motivated us to analyze the day-night traffic pattern, because it is easier to measure and their values can be directly applied to capacity planning tasks. We started by analyzing the day-night traffic pattern of the RedIRIS' network. The conclusions were that this day-night pattern was an invariant during working days. However, weekends do not exhibit the same day-night patter. Instead, the day-night pattern of the weekends was nearly flat. As the amount of traffic during working days is considerably larger than during weekends, we developed a model taking into account only the traffic during working days, ignoring the weekends' traffic. The presented model keeps track of the day-night pattern, averaging measurements in disjoint intervals, thus obtaining a multivariate model for the daily traffic. Based on previous research [vdMMP06, KN02] we fit the distribution of the model to a normal distribution. However, we did not only rely on those works, and tested for multivariate normality real network measurements preprocessed according to our multivariate model. The results of the multivariate normality test show promising, moreover taking into account that they are applied to real world values. Nevertheless, the normality assumption held only when grouping the samples in populations of small size, being always rejected when applied to the whole dataset. We envisaged that this phenomenon was motivated by a change of the normal parameters with time. Thus, we developed an online algorithm for tracking these changes in the parameters, focusing on the mean value of the traffic load.

We presented our algorithm in [MAGD09]. This algorithm uses the previously defined model, and applies powerful machine learning and statistical testing procedures to, respectively, find the changes and determine its significance. The validation of the algorithm [MA10] showed that, under input following the assumptions of the algorithm, our algorithm manages to properly detect the significant change points in a dataset. The application of our algorithm to real network measurements from the Spanish NREN RedIRIS evidences that the network operators can reduce considerably their OPEX, by preventing the network managers to continuously inspecting visually the network time series measurements.

As future work, we plan to further analyze the change points reported by our online algorithm. We find useful to characterize the distribution of the change points, and the correlation of the change points between the incoming and outgoing directions of the same university. We also plan to develop a software application for traffic prediction in hourly intervals.

Bibliography

[And58] T. W. Anderson, *An introduction to multivariate statistical analysis*, Wiley New York, 1958.

[BC02] N. Brownlee and K. C. Claffy, *Understanding Internet traffic streams: dragonflies and tortoises*, IEEE Communications Magazine **40** (2002), no. 10, 110–117.

[BD91] P. J. Brockwell and R. A. Davis, *Time series: theory and methods*, Springer Series in Statistics, Springer, 1991.

[BM01] N. Brownlee and M. Murray, *Streams, Flows and Torrents*, PAM Workshop, 2001.

[BS06] S. A. Baset and H. Schulzrinne, *An Analysis of the Skype Peer-to-Peer Internel Telephony Protocol*, IEEE Infocom, 2006.

[Cla04] B. Claise, *RFC 3954: Cisco Systems NetFlow Services Export Version 9*, 2004.

[Cla06] K. C. Claffy, *"A Day in the Life of the Internet": Proposed community-wide experiment*, ACM Computer Communication Review **36** (2006), no. 5, 39–40.

[Coc97] W. G. Cochran, *Sampling techniques*, Wiley and Sons, NY, 1997.

[CPB93] K. C. Claffy, G. C. Polyzos, and H. W. Braun, *Application of sampling methodologies to network traffic characterization*, ACM SIGCOMM, vol. 23, ACM Press New York, NY, USA, 1993, pp. 194–203.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, Wiley New York, 2001.

[DPV06] A. Dainotti, A. Pescape, and G. Ventre, *A Packet-level Characterization of Network Traffic*, 11th Intenational Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks, 2006, pp. 38–45.

[EP06] S. Ehlert and S. Petgang, *Analysis and Signature of Skype VoIP Session Traffic*, Proceedings of the Fourth IASTED International Conference on Communications, Internet, and Information Technology, 2006, pp. 83–89.

[EV02] C. Estan and G. Varghese, *New directions in traffic measurement and accounting*, ACM SIGCOMM , 2002, pp. 323–336.

[EV03] ———, *New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice*, ACM Transactions on Computer Systems **21** (2003), no. 3, 270–313.

[FXAM04] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon, *Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme*, Computer Networks **46** (2004), no. 2, 253–272.

[GS99] V. Guralnik and J. Srivastava, *Event detection from time series data*, Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, pp. 33–42.

[JB80] C. M. Jarque and A. K. Bera, *Efficient tests for normality, homoscedasticity and serial independence of regression residuals*, Economics Letters **6** (1980), no. 3, 255–259.

[JLM93] V. Jacobson, C. Leres, and S. McCanne, *pcap-Packet Capture library*, 1993.

[JW92] R. A. Johnson and D. W. Wichern, *Applied multivariate statistical analysis*, Prentice-Hall International Editions, 1992.

[KCHP01] E. Keogh, S. Chu, D. Hart, and M. Pazzani, *An online algorithm for segmenting time series*, Proceedings of IEEE International Conference on Data Mining, 2001, pp. 289–296.

[KN02] J. Kilpi and I. Norros, *Testing the Gaussian approximation of aggregate traffic*, Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment (2002), 49–61.

[KPF05] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, *BLINC: multilevel traffic classification in the dark*, Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (New York, NY, USA), vol. 35, ACM, 2005, pp. 229–240.

[LG08] W. J. Liu and J. Gong, *Double sampling for flow measurement on high speed links*, Computer Networks **52** (2008), no. 11, 2221–2226.

[Lil67] H. W. Lilliefors, *On the Kolmogorov-Smirnov test for normality with mean and variance unknown*, Journal of the American Statistical Association **62** (1967), no. 318, 399–402.

[Lim] S. Limited, *Skype*, `http://www.skype.com`.

[MA08] F. Mata and J. Aracil, *Traffic Classification*, Tech. report, Universidad Autónoma de Madrid, 2008, `http://arantxa.ii.uam.es/~fmata/Publications/TechReports/TrafficClassification.pdf`.

[MA10] ———, *Performance Evaluation of an Online Load Change Detection Algorithm*, Accepted for its publication in the Second International Conference on Computer and Automation Engineering, vol. 1, 2010.

[MAGD09] F. Mata, J. Aracil, and J. L. García-Dorado, *Automated Detection of Load Changes in Large-Scale Networks*, Proceedings of First International TMA Workshop (Aachen, Germany), May 2009, pp. 34–41.

[Mah36] P. C. Mahalanobis, *On the generalized distance in statistics*, Proceedings of the National Institute of Science, vol. 12, 1936, p. 49.

[MC00] S. McCreary and K. C. Claffy, *Trends in Wide Area IP Traffic Patterns*, Tech. report, The Cooperative Association for Internet Data Analysis (CAIDA), 2000.

[MD90] J. C. Mogul and S. E. Deering, *RFC 1191: Path MTU Discovery*, 1990.

[MPM05] G. M. Muntean, P. Perry, and L. Murphy, *Objective and Subjective Evaluation of QOAS Video Streaming over Broadband Networks*, IEEE eTransactions on Network and Service Management **2** (2005), no. 1, 19–28.

[MZ05] A. W. Moore and D. Zuev, *Internet traffic classification using bayesian analysis techniques*, Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (New York, NY, USA), 2005, pp. 50–60.

[NAR+04] H. T. M. Neto, J. M. Almeida, L. C. D. Rocha, W. Meira, P. H. C. Guerra, and V. A. F. Almeida, *A characterization of broadband user behavior and their e-business activities*, ACM SIGMETRICS Performance Evaluation Review **32** (2004), no. 3, 3–13.

[Odl03] A. M. Odlyzko, *Internet traffic growth: sources and implications*, Proceedings of SPIE **5247** (2003), 1–15.

[OR98] T. Oetiker and D. Rand, *MRTG: The Multi Router Traffic Grapher*, Proceedings of the 12th USENIX conference on System administration (1998), 141–148.

[OSST04] A. Oveissian, K. Salamatian, A. Soule, and N. Taft, *Fast flow classification over Internet*, Second Annual Conference on Communication Networks and Services Research, May 2004, pp. 235–242.

[PGDM07] M. Perenyi, A. Gefferth, T. D. Dang, and S. Molnar, *Skype Traffic Identification*, IEEE Global Telecommunications Conference, 2007, pp. 399–404.

[PK02] V. Puttagunta and K. Kalpakis, *Adaptive methods for activity monitoring of streaming data*, Proceedings of the International Conferences on Machine Learning and Applications, 2002, pp. 197–203.

[PM07] M. Perenyi and S. Molnar, *Enhanced Skype traffic identification*, Proceedings of the 2nd international conference on Performance evaluation methodologies and tools, 2007.

[PTB+02] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot, *A pragmatic definition of elephants in internet backbone traffic*, Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment (New York, NY, USA), 2002, pp. 175–176.

[PTZD05] K. Papagiannaki, N. Taft, Z. Zhang, and C. Diot, *Long-term forecasting of Internet backbone traffic*, IEEE Transactions on Neural Networks **16** (2005), no. 5, 1110–1124.

[RK96] A. Rueda and W. Kinsner, *A survey of traffic characterization techniques in telecommunication networks*, Canadian Conference on Electrical and Computer Engineering (Calgary, Alta., Canada), vol. 2, May 1996, pp. 830–833.

[RMM08] D. Rossi, M. Mellia, and M. Meo, *Following skype signaling footsteps*, Proceedings of the 4th International Telecommunication Networking WorkShop on QoS in Multiservice IP Networks, 2008, pp. 248–253.

[Rob00] L. G. Roberts, *Beyond Moore's Law: Internet Growth Trends*, Computer **33** (2000), no. 1, 117–119.

[RS94] F. J. Rohlf and R. R. Sokal, *Statistical tables*, WH Freeman, 1994.

[RSSD04] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, *Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification*, Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (New York, NY, USA), 2004, pp. 135–148.

[SAS05] M. Sharifzadeh, F. Azmoodeh, and C. Shahabi, *Change detection in time series data using wavelet footprints*, Lecture Notes in Computer Science **3633** (2005), 127.

[SFKT06] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley, *Characterizing and Detecting Skype-Relayed Traffic*, Proceedings of the 25th IEEE International Conference on Computer Communications, April 2006, pp. 1–12.

[She04] D. Sheskin, *Handbook of parametric and nonparametric statistical procedures*, CRC Press, 2004.

[Ste74] M. A. Stephens, *EDF statistics for goodness of fit and some comparisons*, Journal of the American Statistical Association **69** (1974), no. 347, 730–737.

[Ste97] W. Stevens, *RFC 2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, 1997.

[TMW97] K. Thompson, G. J. Miller, and R. Wilder, *Wide-area Internet traffic patterns and characteristics*, IEEE Network **11** (1997), no. 6, 10–23.

[vdBMvdM+06] H. van den Berg, M. Mandjes, R. van de Meent, A. Pras, F. Roijers, and P. Venemans, *QoS-aware bandwidth provisioning for IP network links*, Computer Networks **50** (2006), no. 5, 631–647.

[vdMMP06] R. van de Meent, M. Mandjes, and A. Pras, *Gaussian traffic everywhere?*, IEEE International Conference on Communications, vol. 2, 2006.

[ZSGK08] M. Zink, K. Suh, Y. Gu, and J. Kurose, *Watch global, cache local: YouTube network traffic at a campus network: measurements and implications*, Proceedings of SPIE **6818** (2008), 681805.

Appendix A

k-means

Throughout the master thesis work we have applied several time clustering techniques in our algorithms. The clustering algorithm selected for this task has been k-means [DHS01]. We present in this appendix a brief description of the k-means algorithm for the sake of completeness.

k-means is a two-step iterative algorithm that finds the clusters that minimize the sum of the squared distances from each instance belonging to the cluster to a representative of it, namely centroid. The algorithm needs as input the number k of clusters expected or actually existing in the dataset. It starts by selecting k random instances from the data as the original centroids. Then, it computes the distances from each instance in the data to every centroid, and assigns the instance to the cluster represented by the centroid to which the distance is the smallest. After that, the *means* step consists of computing the new representatives for each cluster, which are the means of the instances belonging to each cluster. The algorithm then iterates repeating these steps until the new computed centroids are the same that were computed in the previous one.

Appendix B

Sample & Hold

S&H was first introduced in [EV02]. Later, a more comprehensive version was published [EV03]. The objective of the technique is to monitor the traffic of a link, measuring accurately the amount of traffic sent by the "heavy hitters", i.e. the most contributing flows. The results of S&H can then be applied to usage-based pricing, where the most active users are charged proportionally to the amount of the resources consumed, whereas the remaining users just pay a fixed fee. However, as link speeds and the number of flows increase, keeping a counter for each flow is too expensive (using Static Random Access Memory (SRAM)) or slow (using Dynamic Random Access Memory (DRAM)). Therefore, sampling methods should be applied.

S&H use ordinary random sampling, same as NetFlow (see Section 2.2 of Chapter 2), sampling each packet with a probability that depends on its length, but once a packet from a flow is sampled, the following packets belonging to the same flow are also sampled, as follows. If a packet is sampled but the flow it belongs to has not been sampled yet, a new memory entry is created for that flow containing the statistics of the sampled packet. Afterwards, every time that a new packet of the same flow is sent, the memory entry is updated with the corresponding statistics. This procedure is exemplified in Figure B.1.

Searching the memory each time a packet is seen and updating the corresponding counters may lead to overflow resulting in packets not inspected thus reducing the accuracy of the methodology. However, the authors show that S&H has reduced memory requirements that allows the flow memory to be in SRAM instead of in a slow DRAM, allowing scaling the methodology with line speeds.

If we want to detect the flows that send more than $t\%$ of the link capacity in a measurement interval C, there can be at most $100/t$ of such flows. Therefore, the flow memory should be dimensioned to allow the allocation of $100/t * o + S$ flows, where o is an oversampling factor to prevent false positives which may occupy all the positions intended for the most contributing flows and S is some extra memory entries to ensure the flow memory is not filled completely. In order to sample $100/t * o$ flows in average, we should set the bit sampling probability to $p = \frac{100}{t} \cdot \frac{o}{C}$. With these parameters, the authors showed that the false negative probability (i.e. the probability of not detecting one of the large flows of interest) is very close to e^{-o}, that for an oversampling factor $o = 100$ is in the order of 10^{-44}.

S&H has the benefits of being easy to implement and that it generates small reports compared

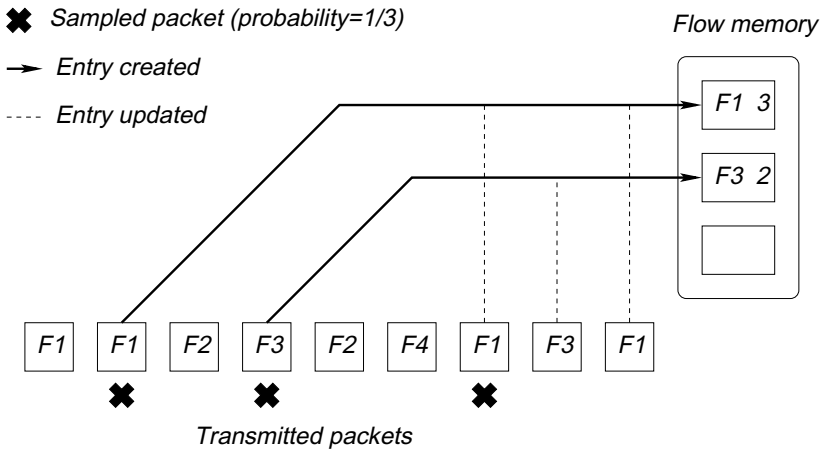

Figure B.1: Example of SH algorithm. The first time a flow is sampled a new entry in the Flow memory is created (solid lines). Then, the counter is updated for the remaining packets belonging to that flow (dashed lines). This figure was taken from [EV02].

to NetFlow. The main differences between both techniques are illustrated in Figure B.2.

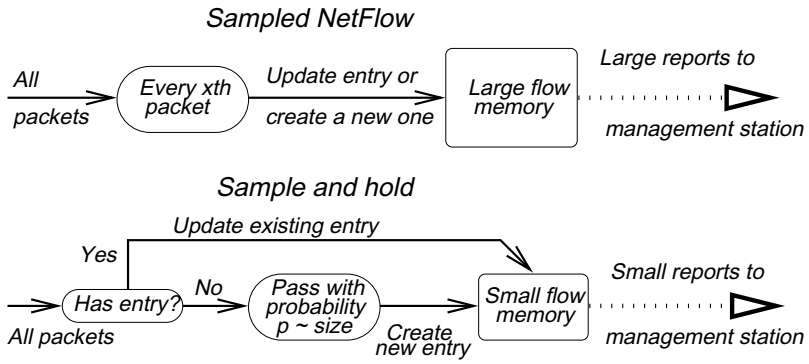

Figure B.2: Differences between NetFlow (top) and SH (bottom). This figure was taken from [EV02].

Appendix C

Univariate Normality Tests

C.1 Kolmogorov-Smirnov Test and Lilliefors' Correction

The KS test is a quite general test to determine the equality of one-dimensional probability distributions. It can be used to compare two samples (two-sample KS test) or to compare a sample with a reference continuous probability distribution (one-sample KS test). In our study we have used the one-sample KS test variant to compare our sample with the normal distribution. The hypothesis of the test is that the sample $X = x_1, x_2, \ldots, x_n$ comes from a continuous probability distribution given by $F(x)$. To proceed with the test the following three steps are needed.

1. Order sample values $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$.

2. Compute the ECDF $F_n(x)$ as follows

$$F_n(x) = \begin{cases} 0 & \text{if } x < x_{(1)} \\ \frac{r}{n} & \text{if } x_{(r)} \leq x < x_{(r+1)} \\ 1 & \text{if } x \geq x(n) \end{cases}$$

3. Compute the maximum discrepancy between the ECDF $F_n(x)$ and the theoretical one $F(x)$ with the statistic

$$D_n = max|F_n(x) - F(x)| \tag{C.1.1}$$

   which distribution, under the null hypothesis, has been tabulated [RS94]. If once fixed α, the computed D_n is greater than the tabulated value, the null hypothesis is rejected.

However, if the theoretical distribution function $F(x)$ is computed by estimating the parameters from the sample, the distribution of D_n is only an approximation, thus the power of the test is reduced [Ste74], and the results of the test are very conservative. The Lilliefors test arises when correcting for this bias. So, Lilliefors [Lil67] computed the distribution of D_n when the parameters of the normal distribution (μ, σ^2) are estimated through the sample parameters $(\bar{x}, \hat{s}^2)$ and tabulated them [She04].

C.2 Jarque-Bera Test

The JB test [JB80] tests the deviation from normality using the skewness and kurtosis of the sample. The JB statistic is as follows

$$JBS = \frac{n}{6}\Big(S^2 + \frac{(K-3)^2}{4}\Big) \tag{C.2.1}$$

where n is the sample size. S is the sample skewness and K is the sample kurtosis given by

$$S = \frac{\hat{\mu}_3}{(\hat{\sigma}^2)^{3/2}} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2)^{3/2}} \tag{C.2.2}$$

and

$$K = \frac{\hat{\mu}_4}{\hat{\sigma}^4} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2)^4}. \tag{C.2.3}$$

This statistic has asymptotically a χ^2 distribution with 2 degrees of freedom under the null hypothesis of normality. This null hypothesis is a joint hypothesis of skewness and excess kurtosis being both zero.

Appendix D

Multivariate Behrens-Fisher Problem

The Behrens-Fisher problem is a statistical problem of testing whether the means of two normally distributed populations $(X^{(1)}, X^{(2)})$ are the same (null hypothesis H_0) or not (alternative hypothesis H_1) when the variances of the populations are unknown. The generalization of this problem to multivariate data is known as the MBFP [And58] that is the one we use here. The assumptions are that $X^{(i)} \sim \mathcal{N}_p(\mu^{(i)}, \mathbf{\Sigma}^{(i)}), i = 1, 2$; i.e. the samples of population i come from a p-variate normal distribution with mean $\mu^{(i)}$ and covariance matrix $\mathbf{\Sigma}^{(i)}$, where in our case $p = 16$. To solve this problem the Hotelling's T^2 statistic is used, and two different cases arise depending on the sizes of the populations. If both populations have the same number of samples N, and the numbering of the samples does not depend on the samples themselves, the procedure is to form a new random variable Y that is the difference of the initial populations, i.e. $y_j = x_j^{(1)} - x_j^{(2)}, j = 1, 2, \ldots, N$. For this new random variable (that under the null hypothesis has zero mean) the sample mean vector $\bar{Y}$ and the sample covariance matrix $\mathbf{S}_y$ are computed. The T^2-statistic in this case is as follows:

$$T^2 = N \frac{\bar{Y} \mathbf{S}_y^{-1} \bar{Y}^t}{N-1} \frac{N-p}{p}, \tag{D.0.1}$$

where $\bar{Y}^t$ denotes the transpose vector of $\bar{Y}$. It can be shown (see chapter 5 of [And58]) that under the null hypothesis T^2 follows a noncentral F distribution with p and $N-p$ degrees of freedom and noncentrality parameter

$$\nu = (\mu^{(1)} - \mu^{(2)}) \mathbf{\Sigma}_y^{-1} (\mu^{(1)} - \mu^{(2)})^t. \tag{D.0.2}$$

Under the null hypothesis, $\mu_1 = \mu_2$, so $\nu = 0$. As the noncentrality parameter is zero, the distribution of equation (D.0.1) turns out to be a Snedecor's F distribution.

On the other hand, when the sizes of the populations are not equal, a transformation is needed before computing the T^2-statistic. If the sizes of $X^{(1)}$ and $X^{(2)}$ are respectively N_1 and N_2, assuming that $N_1 < N_2$ without loss of generality, we obtain a new random variable Q through

the following transformation, as follows:

$$q_j = x_j^{(1)} - \sqrt{\frac{N_1}{N_2}} x_j^{(2)} + \frac{1}{\sqrt{N_1 N_2}} \sum_{k=1}^{N_1} x_k^{(2)} - \frac{1}{N_2} \sum_{l=1}^{N_2} x_l^{(2)}, \quad j = 1, \cdots, N_1, \tag{D.0.3}$$

where $x_n^{(1)}$, $n = 1, 2, \cdots, N_1$, are the samples of $X^{(1)}$ and $x_m^{(2)}$, $m = 1, 2, \cdots, N_2$, are the samples of $X^{(2)}$. As shown by [And58] this new random variable has a mean vector equal to the difference of the mean vectors of the two populations, and the covariance matrix is given by the following equation:

$$Cov(q_n, q_m) = \mathbb{E}[q_n - \mathbb{E}[q_n]] \cdot \mathbb{E}[q_m - \mathbb{E}[q_m]] = \delta_{n,m}(\mathbf{\Sigma}_1 + \frac{N_1}{N_2}\mathbf{\Sigma}_2), \tag{D.0.4}$$

where $\delta_{n,m}$ is the Dirac delta function evaluated in $n-m$ and $\mathbb{E}$ is the Expectation Operator. The T^2-statistic in this case is as follows:

$$T^2 = N_1 \frac{\bar{Q}\mathbf{S}_q^{-1}\bar{Q}^t}{N_1 - 1} \frac{N_1 - p}{p}. \tag{D.0.5}$$

As in the previous case, equation (D.0.5) is distributed under the null hypothesis as a Snedecor's F distribution with p and $N_1 - p$ degrees of freedom.

Once the T^2 statistic is computed taking into account the case that applies of the above described, the statistical test at level α proceeds by comparing the obtained T^2 value with the $1 - \alpha$ percentile of the Snedecor's F distribution with the appropriate degrees of freedom. If the degrees of freedom are p and m, we denote this percentile by $F_{p,m}^{1-\alpha}$. Then, the null hypothesis is rejected if $T^2 > F_{p,m}^{1-\alpha}$.

Appendix E

Analysis of the Hotelling's T Square Statistic

Let us further analyze the T^2 statistic presented in equation (D.0.1) of Appendix D (we suppose that the two populations have the same size N). The term $\bar{Y}S_y^{-1}\bar{Y}^t$ is a quadratic form of the p vector components of the random vector $\bar{Y}$. As we are using synthetic data, we can approximate with the true covariance matrix (the one used to generate the samples in Section 6.3.1) in what follows. This matrix has been chosen to be diagonal, in order to have all the vector components being independent. This implies that the quadratic form defined by the covariance matrix is the weighted sum of the square of all the vector components (being the weights given by the elements of the diagonal covariance matrix).

We now have a look at the simplest case. In this case, all the vector components have the same variance, so the covariance matrix is a multiple of the identity matrix. Then, as the vector components of $\bar{Y}$ are the differences in the means of the two populations, the quadratic form defined by (D.0.1) is the square of the L_2 norm with the usual metric on $\mathbb{R}^{16}$ of the vector $\bar{Y}$, scaled by the variance and other factors that take into account the dimensions of the data and the sample. If we fix the significance value α (for instance, $\alpha = 0.05$) we are comparing the value obtained from (D.0.1) with a value that is a function of N, given that the dimension of the random vector p is fixed. This function is the 1-α percentile of the F distribution with p and $N-p$ degrees of freedom ($F_{p,N-p}^{1-\alpha}$). We reject H_0 if the T^2 statistic value is greater than the value of the function evaluated in that N. This function decreases with N, which means that when we increase the number of samples N, we are stricter with the value of the T^2 statistic (we have more samples so we have better estimations and then we have the same confidence rejecting at lower values of the statistic). So, the maximum value of the function is obtained with the minimum N that it can take. This N is equal to $p+1$, because if $N \leq p$ the estimate of the covariance matrix can be a noninvertible matrix. This means that the maximum value is the 95^{th} percentile of the Snedecor's F distribution with p and 1 degrees of freedom. With the value of p used in our synthetic data ($p = 16$), that gives us a maximum value for the function of 231.9660. Assuming all the vector components of $\bar{Y}$ equal, this gives us

$$T^2 = N\frac{\bar{Y}S_y^{-1}\bar{Y}^t}{N-1}\frac{N-p}{p} \approx N\frac{\bar{Y}\frac{1}{\sigma^2}\mathbb{I}_p\bar{Y}^t}{N-1}\frac{N-p}{p} =$$
$$= \frac{N}{N-1}\frac{N-p}{p}\sum_{i=1}^{p}\frac{\bar{y}_i^2}{\sigma^2} \approx \frac{N}{N-1}\frac{N-p}{p}\frac{p\bar{y}^2}{\sigma^2} =$$
$$= N\frac{N-p}{N-1}\frac{\bar{y}^2}{\sigma^2}. \tag{E.0.1}$$

As we reject the equality of means if the T^2 statistic is greater than the 95^{th} percentile of the F distribution with the corresponding degrees of freedom, this is equivalent to

$$\frac{\bar{y}^2}{\sigma^2} > \frac{F_{p,N-p}^{1-\alpha}}{N}\frac{N-1}{N-p} \tag{E.0.2}$$

For the case stated above ($N-p=1$), we will reject the equality of means if $\frac{\bar{y}^2}{\sigma^2} > 231.9660$. In Table E.1 we summarize the rejecting values under the hypothesis of equality of variances and equality of change in all the vector components for the first ten suitable values of N.

Table E.1: Rejecting values for the quotient between the square of the change in one vector component and its variance.

N	$N-p$	Critical value
17	1	231.9660
18	2	9.1768
19	3	2.7449
20	4	1.3880
21	5	0.8769
22	6	0.6240
23	7	0.4775
24	8	0.3835
25	9	0.3188
26	10	0.2719

It can be observed that these values decrease with N. If we study the critical value as N tends to infinity, this critical value will tend to 0. This means that in the limit we will reject the equality of means unless their difference equals zero. However, with typical values of N this would not happen. These results will be used in the following section when inspecting at fixed significance level the changes reported by the algorithm under the datasets fulfilling the simplifications made in this analysis.

Index