UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ph.D. Thesis

Proactive Measurement Techniques For Network Monitoring In Heterogeneous Environments

Author:
Javier Ramos de Santiago

Supervisor:
Prof. Javier Aracil Rico

Madrid, 2013

DOCTORAL THESIS:	Proactive Measurement Techniques For Network Monitoring In Heterogeneous Environments
AUTHOR:	Javier Ramos de Santiago
SUPERVISOR:	Prof. Javier Aracil Rico

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

PRESIDENT:	Dr. Eduardo Magaña Lizarrondo
MEMBERS:	Dr. Mikel Izal Azcárate Dr. Edmundo Monteiro Dr. József Stéger
SECRETARY:	Dr. Jorge E. López de Vergara Méndez

To my family.
To Victoria.

Summary

Nowadays the increasing bandwidth demand by telecommunication operators' customers as well as the wide spread of new services and applications running over the Internet have converted the design and operation of communication networks into a complex task. To ease the operation and maintenance of such networks, operators have traditionally resorted to the monitoring of their networks to analyze a set of Quality of Service (QoS) parameters and determine the state and quality of the services provided to their customers reducing both Operational Expenditures (OPEX) and Capital Expenditures (CAPEX). Moreover, in the last years, the emergence of other actors such as important banks or Internet companies which provide services making use of their own private communication networks has raised the need for reliable monitoring systems adaptable to different network environments. Traditionally, network monitoring has been classified into active and passive methodologies. The former is based on the injection of traffic on the monitored network to obtain a set of parameters or results. The latter is based on the capture and analysis of traversing network traffic to estimate the value of a set of QoS parameters. Each monitoring flavor presents its own benefits and drawbacks.

In recent years, hybrid systems that make use of both passive and active techniques have been proposed to overcome the limitations of each methodology and get the best of both worlds. Such hybrid technique is known as Reactive Measurement (REM) and uses the value of a set of parameters to trigger response processes to certain values or situations. In this work, Proactive Monitoring (PM) which goes far beyond the REM is presented and proposed. While in REM an event triggers a reaction, in PM, control actions are carried out in order to avoid the occurrence of situations to which react. Additionally, in case such situations are produced, PM allows reacting to them as in a standard reactive system. This new type of monitoring eases not only the maintenance and operation but the dimensioning and deployment of new networks.

The main goal of this thesis is analyzing and proposing proactive monitoring techniques applicable in heterogeneous environments in a simple way. Due to the evolution of actual networks and the diversity of devices and technologies, it is necessary developing monitoring techniques that may be easily integrated in both high-performance systems and low computing-power devices and that may operate at rates ranging from a few Mb/s to multi-Gb/s.

To achieve the proposed objectives, firstly communication networks and QoS fundamentals have been reviewed. Such review has been presented in an orderly way composing a state of the art on the most important methods and concepts related to network monitoring. Next, the literature concerning to active monitoring methods has been analyzed taking into account both file-transfer and packet-pair methods. Such analysis has revealed the influence of external parameters such as Central Processing Unit (CPU) load and self-induced traffic over the active measurement techniques. In heterogeneous environments, such external parameters may not be easily controlled. To overcome this limitation a method for detecting and subsequently removing polluted measurements (due to the influence of external parameters) has been developed. Additionally, it has been proved the influence of traffic control mechanisms such as shapers or policers over the active measurement methods. To solve these problems an algorithm that detects the presence of such mechanisms and estimates, if possible, the parameters of the shaping or policing system is proposed.

Later on, an analysis of the state of the art of passive measurement methods has been performed with particular emphasis on those based on flows and trajectory sampling. Derivative of such analysis and based on the need of obtaining simple methods which are able to operate in heterogeneous environments at different speeds, a fast packet correlation algorithm has been proposed. Such algorithm may be used in selective sampling or trajectory sampling and is able to process about 6 Gb/s of real network traffic. Moreover, an efficient architecture for network flow construction and simple statistics gathering has been presented.

Additionally, in this work, a detection algorithm for time instants in which there is presence of anomalous or polluting traffic based on the analysis of time series is presented. Such algorithm has been validated using real traffic from an operator and a bank network, obtaining promising results.

Finally, an architecture for a proactive monitoring system that integrates the previously presented ideas has been proposed. Particular applicability examples of PM in heterogeneous environments have been also provided.

Keywords: proactive monitoring; packet-pair; shaper detection; flow creation; passive measurements; active measurements; heterogeneous networks;

Resumen

Hoy en día la creciente demanda de ancho de banda por parte de los clientes de los operadores de telecomunicaciones así como la proliferación de nuevos servicios y aplicaciones que funcionan a través de Internet ha convertido el diseño y la operación de redes de comunicaciones en una tarea de gran complejidad. Para facilitar la operación y el mantenimiento, los operadores han recurrido tradicionalmente a la monitorización de sus redes de comunicaciones para analizar un conjunto de parámetros de calidad de servicio (QoS) con los que poder determinar el estado y la calidad del servicio que ofertan a sus clientes reduciendo sus costes capitales y de operación. Además, en los últimos años la aparición de otros actores como grandes bancos o empresas de Internet que disponen de redes privadas de comunicaciones a través de las cuales ofrecen servicios ha despertado la necesidad de sistemas de monitorización fiables que puedan adaptarse a diferentes entornos de red. Típicamente la monitorización de red se ha dividido en monitorización activa y pasiva. La primera se basa en la inyección de tráfico en la red a analizar para obtener un conjunto de parámetros o resultados. La segunda se basa en la captura y análisis del tráfico que circula por la red para la obtención de los parámetros de calidad de servicio. Cada uno de los tipos de monitorización presenta ventajas e inconvenientes.

En los últimos años se ha propuesto el uso de sistemas híbridos que utilizan medidas pasivas y activas para obtener lo mejor de ambas técnicas. Este tipo de medida híbrida conocida como medida reactiva (REM) usa el valor de una serie de parámetros para desencadenar procesos de respuesta ante ciertos valores o situaciones. En este trabajo se presenta y propone el concepto de monitorización proactiva (PM) que va más allá de la monitorización reactiva. Mientras que en la monitorización reactiva se espera la ocurrencia de un evento para desencadenar una reacción, en la monitorización proactiva se realizan acciones de control para evitar que se produzcan dichas situaciones a las que reaccionar. Además, en caso de que estas situaciones se produzcan se puede responder igualmente de manera reactiva. Este nuevo tipo de monitorización facilita no solo el mantenimiento sino también el despliegue y dimensionado de nuevas redes.

El objetivo principal de este trabajo es analizar y proponer técnicas de monitorización proactiva que puedan ser aplicadas en ambientes heterogéneos de manera sencilla. Debido a que las redes actuales han evolucionado y se pueden encontrar diferentes tipos de dispositivos y tecnologías, es necesario crear métodos de monitorización que puedan ser integrados tanto en sistemas de alto rendimiento como en dispositivos con una potencia de cálculo reducida y que puedan actuar sobre un rango de velocidades que van desde los pocos Mb/s hasta tasas multi-Gb/s.

Para conseguir este objetivo, primero se han revisado los conceptos fundamentales acerca de redes de comunicaciones así como de los parámetros de calidad de servicio. Esta revisión se ha presentado de manera ordenada constituyendo un estado del arte de los métodos y conceptos más importantes relacionados con la monitorización. Seguidamente se ha analizado el estado del arte sobre métodos de medida activos analizando los métodos de descarga de ficheros y de pares de paquetes. El análisis de estos métodos ha demostrado la sensibilidad de los mismos a ciertos parámetros como la carga de trabajo del procesador así como a la cantidad de tráfico interferente autoinducido. En los entornos heterogéneos este tipo de parámetros no puede ser fácilmente controlado así que se ha propuesto un método para detectar la contaminación de las medidas debidas a la influencia de estos parámetros para su posterior eliminación en el análisis de los resultados de las medidas. Adicionalmente se ha comprobado que los métodos de medida activos son sensibles a la presencia de mecanismos de control de tráfico como *shapers* y *policers*. Para solucionar este tipo de problemas se ha propuesto un algoritmo que detecta la presencia de este tipo de mecanismos y estima, si es posible, los parámetros que los conforman.

Más adelante se ha realizado un análisis del estado del arte de los métodos de medida pasivos haciendo especial hincapié en aquellos basados en flujos y en *trajectory sampling*. Derivado de este análisis y de la necesidad de obtener métodos sencillos y que sean capaces de funcionar en entornos heterogéneos y a diferentes velocidades, se ha propuesto un sistema de correlación rápida de paquetes para realizar muestreo selectivo y *trajectory sampling* que es capaz de procesar cerca de 6 Gb/s. Además se ha presentado una arquitectura para la formación eficiente de flujos y la obtención de estadísticas sencillas.

Adicionalmente en este trabajo se ha propuesto un algoritmo basado en el análisis de las series temporales de parámetros simples para la detección de instantes temporales en los que existe contaminación por tráfico anómalo. Este algoritmo ha sido validado con tráfico real de una red bancaria obteniendo unos resultados de detección muy favorables.

Por último se ha propuesto y mostrado la arquitectura de un sistema proactivo de monitorización que integra las ideas anteriormente presentadas indicando ejemplos concretos de la aplicabilidad de este tipo de monitorización a las redes heterogéneas.

Palabras clave: monitorización proactiva; medidas activas; medidas pasivas; pares de paquetes; detección de shapers; formación de flujos; redes heterogéneas;

Acknowledgments

Now that this work is coming to an end, it is time to thank all those who have participated in one way or another in the completion of this work. Thank you very much to those who have contributed with their expertise and those who have given me their support on this long road.

First of all I would like to thank my supervisor, Javier Aracil, for his time and dedication on the construction of this work. Thank you for introducing me to the world of research and allowing me to grow as a researcher and professional.

In the same way, I would like to thank my colleagues from the High-Performance Computing and Networking group: Juan Antonio Andrés, José Luis Añamuro, Marco Forconesi, Rubén García, Paco Gómez, Iván González, Diego Guerra, Diego Hernando, Rafael Leira, Jorge López de Vergara, Sergio López, Daniel Michaud, Luis de Pedro, Enrique Prieto, Germán Retamosa, Paula Roquero, Javier Santos, Alfredo Sosa, Gustavo Sutter, Carlos Vega, José Fernando Zazo and those that are no longer here: Miguel Cubillo, Walter Fuertes, Jaime Fullaondo, Alvaro García, Jaime Garnica, Pedro Gómez, Ismael Gómez, José Alberto Hernández, Bas Huiszoon, Víctor Lucas, David Madrigal, Santiago Pina, Mario Poyato, Alfredo Salvador and Diego Sánchez. Thank you for all this time at the laboratory and all the good moments we have shared.

Especially I would like to thank José Luis García, Víctor López, Felipe Mata, Víctor Moreno, David Muelas and Pedro Santiago for their ideas and support throughout these years and all the good moments we have shared. I have learned a lot from you guys.

All my work would not have been possible without the support of the

Universidad Autónoma de Madrid and the Departamento de Tecnología Electrónica y de las Comunicaciones (former Departamento de Informática) of the Escuela Politécnica Superior. I would like to express my gratitude to all the people involved in the projects that have partially funded this work: PASITO and OneLab2.

I would also like to express my gratitude to Carsten Schmoll for hosting me in Berlin. This internship helped me to grow up as a researcher and learn new ways of working. Thank you very much to all the people at Fraunhofer FOKUS and especially to the NGNI research group for making me feel at home. This includes Thomas Magedanz, Christian Henke, Julian Vetter, Ramón Masek, Jakub Kocur, Florian Ermisch and Jacob Schramm.

I would like to acknowledge my family for their support and understanding throughout these years. Thanks to my parents for making me the person I am today. Thank you very much for instilling the value of hard work and curiosity and for providing me the opportunity to grow as a person and as a professional. Thanks to my brother and sister for being an example and leading the way.

I am grateful to my friends for making this long journey a little more enjoyable. We have to celebrate this. I would also like to express my gratitude to all the people with whom I shared my college time. Thanks for all the good times and the support during these years.

Finally, last but not least, I would like to thank Victoria. Without you, this work would not have been possible. Thanks for supporting and cheering me up when everything was a mess. You have been my cornerstone, a part of this work is yours. I definitely owe you a celebration.

Contents

List of Figures

List of Tables

Acronyms

ANOVA ANalysis Of VAriance. 50, 58

API Application Programming Interface. 65

BE Best Effort. 21

BMC Best Master Clock. 31

BPF Berkeley Packet Filter. 135

BTC Bulk Transfer Capacity. 11, 88

CAIDA The Cooperative Association for Internet Data Analysis. 94, 95, 100, 101, 108, 109

CAPEX Capital Expenditures. vii, 2

CBQ Class-Based Queuing. 24

CBR Constant Bit Rate. xxii, 65, 68, 69, 83, 140, 148

CBS Committed Burst Size. 17, 19, 20

CIR Committed Information Rate. 17, 19, 20, 78

CPU Central Processing Unit. viii, 5, 6, 37, 42, 46, 48–50, 55–58, 62, 63, 70, 78, 83, 122, 123, 133, 140, 142, 151

CUSUM Cumulative Sum. 118

CV Coefficient of Variation. 13

DDoS Distributed Denial of Service. 93

DNS Domain Name System. 88

DPI Deep Packet Inspection. 86

DSL Digital Subscriber Line. 12

EBS Excess Burst Size. 17, 19, 20

ECDF Empirical Cumulative Distribution Function. 115

ETSI European Telecommunications Standards Institute. 39, 40

FCFS First Come First Served. 21

FIFO First In-First Out. 21, 22

FN False Negative. 120

FP False Positive. 120

FPC Fast Packet Correlation. xxiii, 85, 99–101, 106, 109, 134, 136, 141, 142, 150

FPR False Positives Rate. 120, 122, 123, 142

FQ Fair Queuing. 22–24

FTP File Transfer Protocol. 114

GPS Global Positioning System. 29, 33, 34, 43

HTML HyperText Markup Language. 28

HTTP HyperText Transfer Protocol. 12, 39, 40, 86, 89, 114, 132

HTTPS HyperText Transfer Protocol Secure. 114

IC Interrupt Coalescence. 38, 44, 46, 58, 70

IEEE Institute of Electrical and Electronics Engineers. 33

IETF Internet Engineering Task Force. 27

IFG Inter-Frame Gap. 95

IP Internet Protocol. 10, 14, 25–28, 33, 86, 90, 91, 95, 97, 103, 104, 106, 135

IPFIX Internet Protocol Flow Information eXport. 26–28, 86–88, 102, 105, 109, 131, 134, 142, 150

IPTV Internet Protocol Television. 111

IPv6 Internet Protocol version 6. 33

JNI Java Native Interface. 65

JSD Jensen-Shannon Divergence. 127, 128, 138, 143

KLD Kullback-Leibler Divergence. 127

LDAP Lightweight Directory Access Protocol. 114

MPLS Multi-Protocol Label Switching. 27

Mpps Million Packets per Second. 97, 101

MRTG Multi Router Traffic Grapher. 28, 86, 87, 102, 105, 106, 109, 117, 131, 134–136, 142, 150

MSS Maximum Segment Size. 89

MTU Maximum Transmission Unit. 10, 58

NAT Network Address Translation. 91

NEBA Nuevo servicio Ethernet de Banda Ancha. 144, 152

NGN Next Generation Networks. 3

NIC Network Interface Cards. 56, 63, 70, 97, 144

NIDS Network Intrusion Detection Systems. 86

NMLib Network Measurement Library. 105, 106, 109, 142, 150

NTP Network Time Protocol. 29–31, 33, 34, 43

OID Object IDentifier. 28

OPEX Operational Expenditures. vii, 2

OTS Off-The-Shelf. 3

OWD One-Way Delay. 3, 11–13, 29, 38, 40, 42, 43, 70, 88, 93, 99, 100, 105, 109, 112, 126, 129, 131, 132, 134–136, 141

PAT Port Address Translation. 91

PBS Peak Burst Size. 17, 20

PCA Principal Component Analysis. 112

PCAP Packet Capture. 86, 89, 102, 114, 120

PIR Peak Information Rate. 17, 19, 20, 78, 80

PLR Packet Loss Rate. 3, 42, 43, 100, 112, 125, 126, 129, 132, 134

PM Proactive Monitoring. vii–ix, 2–5, 7, 13, 25, 37, 85, 109, 125–127, 129, 138, 139, 142

PNMF Proactive Network Monitoring Framework. 125, 130, 136, 138, 143, 151

PQ Priority Queuing. 21, 22

PTP Precision Time Protocol. 29, 31, 33, 34, 43

QoE Quality of Experience. 14

QoS Quality of Service. vii, viii, 1, 5, 6, 9, 11–15, 37–39, 42, 46, 57, 70, 78, 83, 86–89, 105, 108, 109, 125, 126, 129, 136, 138–140, 142, 143, 145, 150

RAM Random Access Memory. 94

RCPQ Rate-Controlled Priority Queuing. 22

REM Reactive Measurement. vii, 2, 126

ROC Receiver Operating Characteristic. 120

RR Round Robin. 22–24

RSS Receive-Side Scaling. 95

RT Real Time. 22

RTP Real-time Transport Protocol. 27, 114

RTT Round-Trip Time. 12, 29, 30, 40, 43, 78, 88–90, 105, 125, 126, 131

SCCP Skinny Client Control Protocol. 114

SDN Software-Defined Networking. 3

SLA Service-Level Agreement. 3, 4, 15, 17, 38, 144

SNMP Simple Network Management Protocol. 28, 88

SPAN Switched Port Analyzer. 86

SRR Shaped Round Robin. 24

TAI International Atomic Time. 32

TCP Transmission Control Protocol. xxi, xxii, 12, 14–16, 21, 22, 25, 26, 40, 41, 43, 48, 51, 52, 55–57, 65–69, 80, 88–91, 97, 103–106, 113, 114, 132

TN True Negative. 120

ToS Type of Service. 26

TP True Positive. 120

TPR True Positives Rate. 120, 123, 142

TTL Time To Live. 100

UDP User Datagram Protocol. xxii, xxv, 21, 26, 31, 33, 41, 43, 55–61, 65, 67–69, 76, 83, 90, 91, 97, 106, 113, 132, 140, 148

UTC Coordinated Universal Time. 29

VLAN Virtual Local Area Network. 112

VoD Video on Demand. 114, 115, 120

VoIP Voice over Internet Protocol. 14, 15, 69, 111, 112, 114, 115, 120

WFQ Weight Fair Queuing. 23

WRR Weighted Round Robin. 24

Chapter 1

Introduction

This chapter provides an overview of this Ph.D. thesis and introduces its motivation, presents its objectives and hypothesis, and finally describes its main contributions outlining its organization.

1.1 Overview and Motivation

Nowadays, the number of Internet users is growing continuously and quickly. Such increase has entailed an increment on the complexity and size of both commercial and domestic networks. Moreover, new services and protocols have arisen changing several of the previously well-established ideas about traffic characterization. In this light, monitoring and characterizing networks have become paramount importance tasks for both operators and regulatory bodies. In this scenario, several data must be gathered to analyze the behavior and proper working of networks. This information can be condensed into a set of measurable parameters that can be used to assure Quality of Service (QoS) and derive operational related facts. To accomplish such task, several network measurement methodologies can be applied. These methodologies can be grouped into active and passive categories. Each methodology has its advantages and disadvantages and must be applied depending on the monitoring scenario. Typically, passive monitoring has minimal impact on the monitored network at the expense of gathering less precise metrics. On the contrary, active monitoring provides precise metrics but introduces traffic on the measured network modifying its behavior and affecting production traffic which in some cases is unacceptable.

In recent years, hybrid systems that make use of both passive and active techniques have been proposed to overcome the limitations of each methodology and get the best of both worlds. Such hybrid technique is known as Reactive Measurement (REM) [AP08, BKM+09] and uses the value of a set of parameters to trigger response processes to certain values or situations. In this work Proactive Monitoring (PM) which goes far beyond the REM is presented and proposed. While in REM an event is expected for triggering a reaction, in PM control actions are carried out in order to avoid the occurrence of situations to which react. Additionally, in case such situations are produced, PM allows reacting to them as in a standard reactive system. This new type of monitoring eases not only the maintenance and operation but the dimensioning and deployment of new networks. Moreover, PM reduces the amount of information transferred over the network which results in a better utilization of the network resources especially when the same network is used for both data transmission and monitoring. Additionally, the use of PM techniques reduces the operational and capital expenditure investments Operational Expenditures (OPEX) and Capital Expenditures (CAPEX) providing centralized network management and early problem diagnosis. In contrast with traditional statistical monitoring, PM provides more information about the network state in a real-time fashion allowing also the construction of statistical models on top of the gathered information.

The combination of measurement methodologies implies that several decisions must be taken on whether active or passive monitoring must be used. PM provides both time scheduled and event based measurements but the decision of using polling/time scheduled or event-based methodologies depends on the monitored network and the real-time data gathered. The selection of the applicable measurement methodology on large-scale networks entails the correlation of network traffic across multiple hops. Such correlation is not trivial and the process complexity increases as the monitored network grows in number of customers and the capacity of the links increases. Moreover traditional networks have evolved in the last years into Next Generation Networks (NGN) composed of a heterogeneous combination of devices and technologies that has raised the complexity of the monitoring tasks. Due to such evolution, it is necessary to develop monitoring techniques that may be easily integrated in both high-performance systems and low computing-power devices and that may operate at rates ranging from a few Mb/s to multi-Gb/s. Such diverse scenario presents several challenges as monitoring systems must be as simple and powerful as possible while maintaining a high degree of flexibility and reliability with minimal cost. To address such tasks, the community has recently focused in the use of Off-The-Shelf (OTS) systems [GDMR+13, GHH+09, DA03] which combine commodity hardware and open-source software in a simple and low-cost solution. Depending on the characteristics of the measured network, such systems may be tuned to reduce the costs or increase the performance.

Additionally, operators are being increasingly aware every day of the importance of realistic network measurements not only in their core networks by also on their access networks to have a clearer idea of the quality perceived by their users. Moreover both corporate and domestic customers have become more demanding in terms of bandwidth and quality. For example corporate users, sign exclusive contracts with the operators which contain hard restrictions respecting to availability, One-Way Delay (OWD), Packet Loss Rate (PLR) or offered bandwidth. Such contracts are ironclad by means of a Service-Level Agreement (SLA) and breaches in the contracts result in strong monetary losses for the operator. In such scenarios, PM helps operators not only providing real-time realistic measurements that may prevent the violation of the agreed SLA values but also detecting significant changes in the characteristics and behavior of the customer network allowing the operator to offer new products and services tailored to the customer's need.

Recently, the development of Software-Defined Networking (SDN) techniques has opened new avenues for an efficient network deployment and use. The integration of PM inside the SDN design flow fits perfectly, as the incorporation of a centralized measurement system along with the control plane allows a more efficient use of the network resources and eases the problem detection and automatic testing.

In accordance with all the above mentioned, PM has turned out to be an important methodology to provide quality to the operators' customers and to ease the use and deployment of the networks of the future with applicability in various fields such as traffic engineering, attack and threats detection, SLA validation or network design and operational verification.

1.2 Objectives

This thesis presents a methodology for applying PM based on novel active and passive techniques in an efficient way. The main objectives of the thesis are:

1. Analyzing and developing new active and passive monitoring techniques.
2. Creating new algorithms for active and passive network measurement integration.
3. Developing a PM framework based on simple decision algorithms.
4. Applying all the developed techniques in heterogeneous environments with minimal cost.

Little work has been carried out in the field of proactive monitoring. The main problem is the lack of concreteness on the metrics used and the effective performance obtained by the proposed techniques. This thesis aims to fill the existing gap proposing a set of metrics that can be used to determine whether active or passive monitoring should be used, providing as well a framework to execute proactive monitoring in large-scale heterogeneous networks. Moreover, the performance of the proposed solution is analyzed in terms of achieved throughput.

To tackle the monitoring tasks, new algorithms and methods must be designed in order to work in heterogeneous environments. In this thesis, one of the main goals is bringing into focus the achieved performance of the algorithms due to the wide range of link speeds present on heterogeneous networks. Additionally, this work has focused on the simplicity and efficiency of the algorithms in terms of computational power owing to the restrictions of the heterogeneous systems.

Regarding active monitoring methods, the main objective is the characterization of the pollution of the measurements by the influence of external parameters such as Central Processing Unit (CPU) load or interfering traffic. Such characterization is important as measurement systems on heterogeneous networks are deployed over non-dedicated systems that execute multiple concurrent processes and may degrade the quality of the measurements providing, thus, a wrong estimation of QoS parameters. Besides designing pollution-aware algorithms, active algorithms should take into account the influence of shaping and policing methods which are very common in corporate and domestic environments. The detection and characterization of the parameters of such control methods is a paramount importance goal of this thesis.

Considering passive monitoring techniques, the main goal of this work is the characterization and proposal of new packet and flow correlation methods focusing on the performance analysis and the limitation on the use of monitoring systems' resources.

Finally, with respect to the integration of active and passive methodologies into PM systems, the main goal is the development of simple triggering algorithms with low computational cost based on minimal information. An additional objective is the development and evaluation of a PM framework that puts together all the previous ideas in a simple and cost-effective manner.

1.3 Thesis Structure

The rest of the present document is structured as follows. First, in Chapter 2, communication networks and QoS fundamentals have been reviewed. Such review is presented in an orderly way and provides a state of the art on the most important methods and concepts related to network monitoring focusing on QoS metrics, shaping and policing methodologies, router queuing disciplines and flow-level and packet-level monitoring systems. Due to the importance of time synchronization on network monitoring a brief review of the most important methodologies is also presented.

Next, in Chapter 3 the literature concerning to active monitoring methods has been analyzed taking into account both file-transfer and packet-pair methods. In this section an exhaustive analysis of the influence of external parameters over the accuracy of the measurements has been carried out.

Such analysis has revealed the influence of external parameters such as CPU load and self-induced traffic over the active measurement techniques. In heterogeneous environments, such external parameters may not be easily controlled. To overcome this limitation a method for detecting and subsequently removing polluted measures (due to the influence of external parameters) has been developed. Additionally, it has been shown that traffic control mechanisms such as shapers or policers, have an influence over the active measurement methods. To solve these problems an algorithm that detects the presence of such mechanisms and estimates, if possible, the parameters of the shaping or policing system is proposed.

In Chapter 4, an analysis of the state of the art of passive measurement methods has been performed with particular emphasis on those based on flows and trajectory sampling. Derivative of such analysis and based on the need of obtaining simple methods which are able to operate in heterogeneous environments at different speeds, a fast packet correlation algorithm has been proposed. Such algorithm may be used in selective sampling or trajectory sampling and is able to process about 6 Gb/s of real network traffic. Moreover, an efficient architecture for network flow construction and simple statistics gathering has been presented.

Additionally, in this work, a detection algorithm for time instants in which there is presence of anomalous or polluting traffic based on the analysis of time series is presented in Chapter 5. Such algorithm has been validated using real traffic from an operator and a bank network, obtaining promising results.

Finally, an architecture for a proactive monitoring system that integrates the previously presented ideas has been proposed in Chapter 6. Particular applicability examples of PM in heterogeneous environments have been also provided.

Chapter 2

Background

This chapter provides the background and revises the most relevant concepts related to Quality of Service (QoS) and monitoring metrics. The structure of the chapter is as follows. First, traffic QoS metrics and assurance mechanisms such as traffic shaping and policing are described in Section 2.1 in order to determine the base measurement scenarios. Section 2.2 describes some important ideas and technologies related to traffic monitoring including the definition of key aspects such as flows and sessions as well as time synchronization mechanisms.

2.1 QoS Mechanisms

In order to provide QoS on both domestic and commercial networks, operators apply different techniques to control traffic at intermediate routers. Such techniques can produce undesirable effects such as packet drop or delay over the traversing traffic. In this scenario, understanding the QoS mechanisms and effects turns out to be of paramount importance on the design and implementation of measurement techniques. Additionally, a set of parameters must be monitored in order to assure QoS. This section provides an overview of the most relevant QoS parameters and techniques that must be taken into account when designing and implementing measurement methodologies.

2.1.1 QoS metrics

Capacity

The capacity of a level 2 link is defined as the constant transmission rate. Such constant transmission rate is limited by the physical characteristics of the transmission medium and by the electrical/optical characteristics of the transmitter and receiver hardware. At Internet Protocol (IP) level, capacity is defined as the transmission rate taking into account the overhead produced by the link layer headers. The capacity in this case can be defined as:

$$C_{L3} = C_{L2} \frac{1}{1 + \frac{H_{L2}}{L_{L3}}} \tag{2.1}$$

where C_{L2} is the level 2 capacity, H_{L2} is the link level preamble and header length and L_{L3} is the IP packet size represented in bytes. The capacity of the hop i, C_i, is defined as the maximum transmission rate at IP level. Therefore, the capacity of one hop is always calculated using level 2 Maximum Transmission Unit (MTU) IP sized packets. The capacity of an end-to-end path, C, is defined as the minimum capacity of all the path hops. The hop with the smaller capacity is known as *narrow link*.

Available Bandwidth

The available bandwidth of an end-to-end path is defined as the non-used capacity in a given time period. This metric depends on as much from the physical characteristics as from the link traffic load along the time. To calculate the available bandwidth it is necessary to know the loads of the links in advance. As such instantaneous utilization is not very practical, the average utilization over a time period is used. Such metric is defined as:

$$\overline{u}(t - \tau, t) = \frac{1}{\tau} \int_{t-\tau}^{t} u(x)\, dx. \tag{2.2}$$

where u(x) is the instantaneous utilization of the link that either takes the value 1 or the value 0 and τ is the time interval also known as *averaging timescale*. Taking into account the previous definition, it can be stated that the average available bandwidth of a hop i is:

$$\overline{A_i} = (1 - \overline{u_i})C_i \tag{2.3}$$

and the average available bandwidth of an end-to-end path is the minimum of the per-hop average available bandwidth. The hop with the minimum available bandwidth is known as the *tight link*. As the average available bandwidth varies along the time, it is necessary measuring it quickly especially when the measured value is used to adapt the content distribution of upper-layer applications.

Throughput

Another important QoS metric related to bandwidth is the TCP throughput of a connection. The main disadvantage of this metric is that depends on different factors such as the data transfer size, the number of concurrent TCP connections, the congestion in the traversing links or the amount of TCP or UDP cross traffic. Bulk Transfer Capacity (BTC) [MA01] represents the throughput of a TCP connection. Note that this metric is not applicable in all scenarios and due to the dependency on other parameters it has narrow scope of application.

One-Way Delay (OWD)

OWD is defined as the elapsed time between the first bit of a packet in the source observation point and the last bit of a packet in the destination observation point [AKZ99a]. This metric can be measured in a certain link or along an end-to-end path. End-to-end OWD is composed of: transmission delay, propagation delay, processing delay and queuing delay [HMn07]. Transmission delay is the time needed to transmit all bits in a given packet. Such delay depends on the packet length, transmission rate and physical medium. Propagation delay is the time elapsed between the last bit of a packet is emitted and the same bit is received on the other end. Processing delay is the time needed by each router or network equipment to process a packet. Queuing delay is the time spent by a packet waiting in a router queue until is processed.

Round-Trip Time (RTT)

RTT is defined as the time interval between the first bit of a Transmission Control Protocol (TCP) sent segment and the last bit of the corresponding received TCP ACK [CFGS11]. Although RTT is defined over TCP, the concept could be extended to any bidirectional protocol [AKZ99c] even to services on top of TCP. Unlike OWD, RTT provides information about the two directions of a communication which is useful when measuring asymmetric links such as Digital Subscriber Line (DSL). RTT also takes into account processing times at each end of the connection. For example, measuring RTT of a HyperText Transfer Protocol (HTTP) connection implies waiting for the server to either generate an HTTP response or a TCP RST in case connection cannot be handled. Measuring RTT in this scenario implies including the HTTP server processing delay as part of communications delay which may be unacceptable in some cases.

Jitter

The term jitter is normally misused depending on the context. In the QoS scenario, jitter refers to the delay variation of a given stream of packets. From now on, the term packet delay variation will be used instead of jitter. In this case, delay variation over a stream of packets can be defined as the difference between the OWD of a selected group of packets [DC02]. Such selection can be done by means of either a deterministic or random selection function applied to the set of received packets. Only ordered packet pairs are used to calculate delay variation. Packet delay variation of packet i is analytically defined as:

$$PDV(i) = D(i) - D(min) \tag{2.4}$$

where D(i) represents the delay of packet i and D(min) represents the minimum packet delay on the observed time interval. Additionally, other approaches define the packet delay variation as the standard deviation of the OWD of the observed packets in given a time period [IDVFE10]. Using the previous definition, the packet delay variation can be calculated as:

$$PDV = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}(OWD_i - \overline{OWD})^2} \tag{2.5}$$

where OWD_i represents the OWD of $i-th$ packet and $\overline{OWD}$ represents the average value of OWD on the measurement interval. Note that packet OWD must be calculated disregarding lost packets as stated in [AKZ99a]. Other approach to measure delay variation consist on calculating the Coefficient of Variation (CV) of the measured packets OWD. This metric provides a normalized measure of OWD dispersion. Unlike the previous approaches, the CV method provides a unitless magnitude which gives an idea about whether the delay variation is large or not. The larger the coefficient of variation is, the greater the variability of OWD is. This approach is useful when relative information is used to determine the quality of a monitored end-to-end path or link. Delay variation as a coefficient of variation is analytically defined as:

$$PDV = \frac{\sigma_{OWD}}{\mu_{OWD}} \tag{2.6}$$

where σ_{OWD} represents the standard deviation of the measured OWDs and μ_{OWD} represents the average value of the measured OWDs. The coefficient of variation can be expressed also as a percentage which fits better into threshold-based decision systems and eases the integration into Proactive Monitoring (PM) decision systems.

Packet Loss

One important parameter when analyzing QoS is packet loss. Packet loss is defined as the amount of lost packets to total sent packets in a given time period [AKZ99b]. One packet is considered lost if it does not reach its destination, arrives with errors or it is received with excessive delay. Note that packets may not reach their destination due to several causes such as: packet queue drops along an end-to-end path or physical link problems. In case of protocols that allow fragmentation such as IP, one packet is considered as lost if one of its fragments is lost. Additionally, as stated above, some protocols such as Voice over Internet Protocol (VoIP) related, mark a packet as lost when it has arrived with a large delay and it is no longer necessary. Packet loss is an important parameter as a high packet loss can imply throughput degradation due to error correction mechanisms implemented in transport protocols such as TCP. Moreover, this parameter has a big impact over real-time protocols as degrades the Quality of Experience (QoE) perceived by a user. Packet loss can be measured either one-way or round-trip. Measuring packet loss in a one-way fashion is beneficial as packet loss rate does not necessarily have to be symmetric although links are symmetric. This is particularly relevant when QoS policies are applied over the measured links since each direction can have its own set of QoS rules. To actively measure the packet loss rate between a source and a destination point, a packet train is sent containing incremental sequence numbers. On arrival, packets are received and analyzed to detect gaps in the sequence of identifiers. With all this information, packet loss rate is calculated using the following equation:

$$PLR = 1 - \frac{packets_{RCV}}{packets_{SND}} \tag{2.7}$$

where $packets_{RCV}$ denotes the amount of correctly received packets and $packets_{SND}$ the total sent packets. Packets can be sent using constant inter-departure times or generated randomly following a specific pattern such as a Poisson process [AKZ99b]. The sending rate of the packets must be correctly adjusted as selecting a high rate implies a lot of interfering packets while selecting a low rate can mask interesting networks effects

To monitor packet loss passively several approximations have been presented [PKP+06, FUK+09]. Such approximations are based on the retrieval and correlation of traffic between two or more points on the network. Analyzing the traffic that flows between two points allows the proposed methodologies to estimate the traffic lost in one specific link or end-to-end path subtracting sent traffic to received traffic. Other approximations such as [BV02], track TCP connections and calculate packet losses using TCP sequence number and retransmission information. These methodologies, present some errors on their estimation as only a subset of traffic or network paths are monitored. Although, the precision is fair enough for general purposes, some other applications such as VoIP need a more accurate estimation. Additionally, capturing and correlating traffic in high speed environments [FUK^{+}09] is not a trivial task which may require special hardware or software.

2.1.2 Traffic Shaping and Traffic Policing

One important aspect when measuring QoS parameters is the influence of QoS assurance mechanisms such as traffic shaping and traffic policing. Such mechanisms modify several traffic characteristics resulting in different QoS parameters values than expected. Specifically, shaping and policing techniques have impact on the capacity, available bandwidth, packet loss and delays. To get the big picture of the measurement scenario is important to know how such mechanisms work and quantify their impact on the QoS metrics. Traffic shaping and traffic policing are logical methods to limit and control bandwidth in communication networks. Such limitation usually is used to control and ensure that a packet or data source adheres to a specific contract or Service-Level Agreement (SLA). Both shaping and policing, attempt to maintain the traffic rate below certain predetermined level. The main difference between them is how they deal with traffic and the areas in which these methods are applicable. Traffic policing uses a drop or marking policy. Packets belonging to bursts that exceed the previously marked limit rate are dropped or marked as droppable for further processing. This method does not modify the traffic in any way so that the original delay is maintained and also the original traffic characteristics. It therefore maintains traffic bursts contained in the original traffic but not exceeding the limit. Whenever limits are exceed, packets are dropped and consequently packet loss may occur while measuring. Traffic policing can be applied on both inbound and outbound traffic. The main disadvantage of policing is the exceeding rate traffic loss which may degrade the performance of active

TCP connections due to the multiple retransmissions sent. On Figure 2.1 the effect of policing over traffic can be observed.

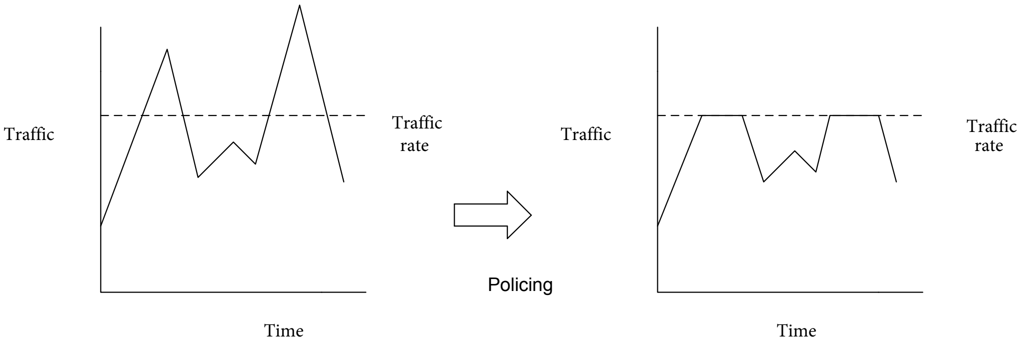


Figure 2.1: Traffic Policing

Traffic shaping uses a queuing-based policy to maintain a constant output rate. Every time a packet arrives to the shaper is queued. At constant time intervals packets are dequeued producing a constant output rate. Unlike traffic policing, original traffic characteristics and delays disappear due to queuing process. The main advantage of this method, besides maintaining a constant output rate, is that allows controlling incoming traffic bursts to obtain a smoother and continuous output. Additionally, this method reduces traffic losses whenever incoming traffic exceeds the predetermined traffic rate reducing thus, the average number of retransmissions. The main disadvantage of the method is the delay increase due to queuing which can adversely affect real-time systems or multimedia. The Figure 2.2 shows the effect of shaping over the traffic.

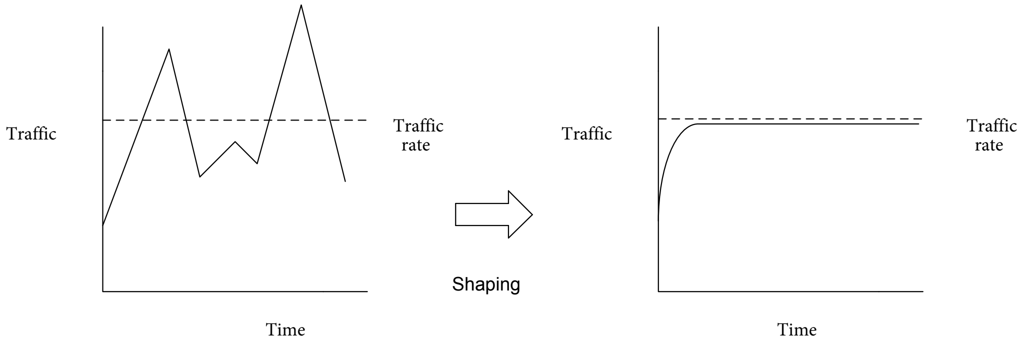


Figure 2.2: Traffic Shaping

Both in traffic shaping and policing several parameters can be adjusted to calibrate the tolerance and operation mode. Such parameters are:

- Peak Information Rate (PIR): maximum customer transmission rate expressed in bits/s and previously agreed between customer and operator by means of a SLA. PIR value can never be greater than the capacity of the link provided by the operator.

- Committed Information Rate (CIR): long-range average traffic rate that the operator agrees to provide to a customer by means of a SLA. This parameter is expressed in bits/s and is generally lower than the PIR. In any case, the CIR can never be greater than the PIR.

- Committed Burst Size (CBS): maximum allowed burst size. This parameter specifies the maximum number of bytes that can be transmitted at PIR without violating the CIR agreement.

- Excess Burst Size (EBS): Burst size threshold above CBS. Whenever traffic bursts exceed the EBS, incoming packets are marked as droppable.

- Peak Burst Size (PBS): CBS similar parameter defined with respect to PIR instead of CIR.

Token Bucket Algorithm

For both shaping and policing, Token Bucket algorithm is used to control the transmission rate taking into account traffic bursts. The algorithm is based on the existence of a bucket full of tokens. A token represents an amount of bytes or a packet of defined size. Token Bucket algorithm is defined by the next variables:

- r: Token generation rate
- d: Maximum bucket size
- C: Maximum transmission rate
- B: Buffer size

Figure 2.3 shows the diagram of an standard Token Bucket.

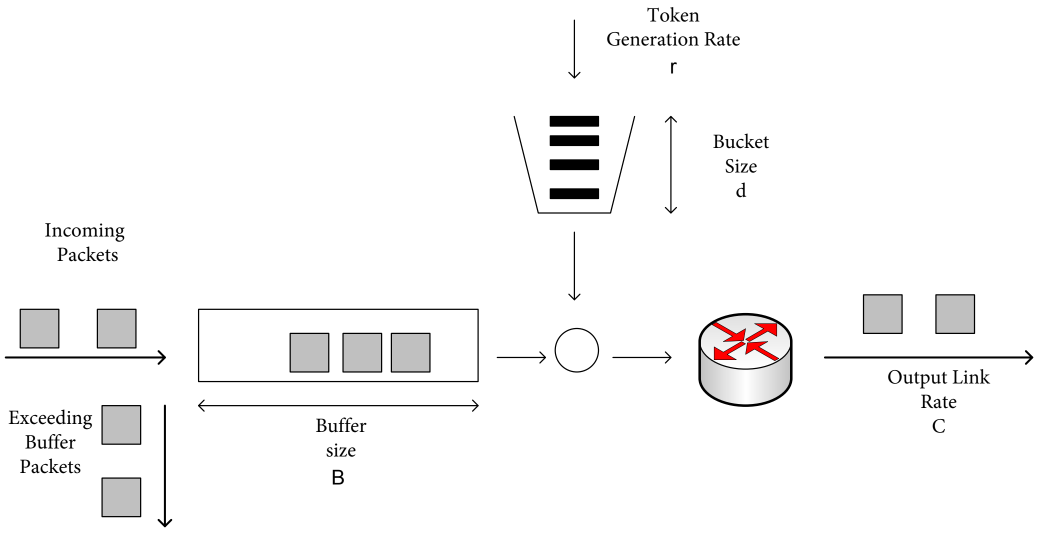


Figure 2.3: Token Bucket Diagram

Tokens are generated and stored in the bucket at rate r. In case that the bucket exceeds its maximum capacity d, new generated tokens are dropped. Whenever traffic is to be sent, the bucket is checked for the existence of as many tokens as bytes or packets are needed to be sent. In case that the bucket contains enough tokens, the packet or packets are sent and tokens are removed from the bucket. If the bucket is empty, depending on whether a queue is used or not, packets are queued or dropped. Queued packets must wait until enough tokens are available. Using this token consumer/producer mechanism, the output traffic rate is controlled. The algorithm allows the existence of traffic bursts. If the bucket is full of tokens, bursts of length d at rate C are allowed. On the contrary, if the bucket is empty, packets are sent at rate r. Applying the algorithm the previously commented shaping and policing parameters may be set. The CIR parameter corresponds to the r parameter of the model while the PIR corresponds to the C parameter and the CBS to the d parameter.

Understanding this admission control mechanism is important, as stated before, since both commercial and domestic networks are nowadays logically limited by this mechanism.

Traffic Policing using Token Bucket

Single-Rate Three Color Marker (srTCM) Single Rate Three Color Marker [HG99a] is a Token Bucket based mechanism to apply policing at a single rate, mainly CIR. The method marks packets with three different colors (green, yellow and red) according to the incoming traffic. Whenever packets must be discarded, red and yellow ones are discarded in first place. The configurable parameters for this method are: CIR, CBS and EBS. The main objective of this mechanism is to ensure that the average traffic rate is roughly the CIR. To this end, two buckets named C and E are used. In the case of C bucket, the maximum size is CBS while in case of E bucket is EBS. The token generation rate for both of them is CIR—every 1/CIR seconds a token is generated. Each time a token is generated is automatically added to the buckets unless buckets are full.

When a B bytes packet arrives, the bucket C is checked for token existence. In case C has enough tokens, packet is marked as green and the corresponding tokens to B bytes are removed from C bucket. On the contrary, if the E bucket has enough tokens, packet is marked as yellow and the corresponding tokens to B bytes are removed from the E bucket. If both E and C buckets happen to be empty, packet is marked as red. Using this method, output rate is enforced to be in the bounds of CIR allowing certain bursts depending on the values of CBS and EBS

Two-Rate Three Color Marker (trTCM) Two Rate Three Color Marker [HG99b] is a Token Bucket based mechanism to apply policing independently at two rates—CIR and PIR. As the previous algorithm, incoming packets are marked as green, yellow or red. The configurable parameters for this method are: CIR, PIR, CBS and PBS. This mechanism uses two buckets named C and P. The bucket C has a maximum capacity of CBS tokens and a token generation rate of CIR. The bucket P has a maximum capacity of PBS tokens and a token generation rate of PIR. Initially, both buckets are full and each time a token is generated is automatically added to its corresponding bucket unless is full.

When a B bytes packet arrives, the bucket P is checked for token existence. In case P does not have enough tokens, packet is marked as red. If the bucket P has enough tokens, the C bucket is checked for token existence. If the bucket C does not have enough tokens, packet is marked as yellow and tokens corresponding to B bytes are removed from the P bucket. If the C bucket does have enough tokens, packet is marked as green and tokens corresponding to B bytes are removed from both C and P buckets.

Traffic Shaping using Token Bucket To apply shaping a Token Bucket method can be used. Such method is similar to the Single Rate Three Color Marker and Two Rate Three Color Marker policing methods. Tokens are generated at a constant CIR rate if the bucket is empty. To apply shaping it is necessary the existence of an admission queue where incoming traffic is stored. When a packet of B bytes arrives, the bucket is checked for token existence. While the bucket has enough tokens packets are sent at PIR rate. In this way, adjusting PIR and CIR, the bursty traffic is regulated and a constant CIR output rate is achieved.

2.1.3 Router Queuing Disciplines

In addition to the physical or logical traffic limitations, there exists another source of uncertainties when measuring large-scale networks. Such source of uncertainties is associated to the behavior of the routers that compose the measured network, specifically, to the queuing disciplines used to manage the traffic. Router queuing mechanisms can modify traffic characteristics such as delay, delay variation or effective bandwidth. Thus, it is important to analyze and understand the most popular queuing disciplines in order to design effective and realistic measurement methods. In a queue system, several queuing disciplines exist. Such disciplines allow organizing the traversing traffic and deal with it one way or another. Some queuing disciplines can act as bandwidth limiters which clearly impacts on the measurements. In this part of the thesis, some queuing disciplines will be described discussing their advantages and disadvantages as well as the traffic characteristics that may modify.

First In-First Out (FIFO)

FIFO is the basic and standard discipline used by default in most of the commercial routers. In this discipline, the packets are queued in the order they arrive and depart from the queue in the same order. This discipline is also known as First Come First Served (FCFS). The main benefit of this method is its simplicity which eases the implementation of both software and hardware routers. FIFO process all the packets in the same manner which makes it a good method for processing traffic marked as Best Effort (BE). This fact does not make it particularly good for other type of traffic since all traffic is penalized in the same way when congestion exists. For example, FIFO favors User Datagram Protocol (UDP) traffic over TCP insomuch as TCP uses retransmissions and reduces its rate to minimize packet losses.

Priority Queuing (PQ)

PQ somewhat solves FIFO discipline problems defining N queues with priority ranging from one to N in such way that traffic can be classified into different classes. The method is based on the idea that traffic from queue j can only be dequeued if queues from one to j-1 are empty. This mechanism retains the simplicity of FIFO queuing while traffic classification is possible. The major drawback of the method is that starvation can occur. For example, if traffic is always present in high priority queues, low priority queues are never processed. This discipline is highly recommended in scenarios where high-priority traffic represents a small fraction of total traffic. For instance, this discipline can be used to create Real Time (RT) traffic queues that can be used for video or audio transmission. This queue discipline must be used with extremely caution when managing TCP traffic as TCP congestion control mechanisms may cause more serious starvation for other traffic in different queues.

To address the starvation problem one queuing discipline is used, Rate-Controlled Priority Queuing (RCPQ). In this discipline several usage thresholds are applied to each queue. Thus while higher priority queues do use less capacity than the predefined usage threshold, the method works as PQ. When the threshold is exceeded traffic is restricted in order to favor lower priority queues.

Fair Queuing (FQ)

FQ is a discipline that allows the creation of classful queues based on packet contents. Each packet that arrives to the system is classified in one queue ranging from one to N. Each queue has assigned one fraction of 1/N of the total output bandwidth. The system scheduler loops over the N queues in Round Robin (RR) order dequeuing one packet per queue in case the queue is not empty.

This discipline is easily implementable and scalable in such a way that adding a new queue does not imply significant changes on the algorithm. One of the disadvantages of this method is the negative impact over traffic that demands a minimum rate. In this case, as the total bandwidth is divided by 1/N, if the traffic demand is higher than 1/N packet losses can occur. For instance, bandwidth measurement algorithms need to have available all the output bandwidth to perform correctly and provide reliable results and only 1/N will be available.

Other FQ problem is that queues containing large packets obtain more output bandwidth than queues with smaller ones as one packet per queue is dequeued each iteration. Thus, small packets obtain less output bandwidth which can affect negatively especially when such packets are used for bandwidth measurements or multimedia content transmission purposes.

Weight Fair Queuing (WFQ)

This queuing discipline is an approximation to the Generalized Processor Sharing (GPS) [PG93]. The algorithm divides the incoming traffic into N queues and the output bandwidth between such N queues in a weighted fashion. The global weight sum is equal to the output capacity. In WFQ unlike in FQ, the scheduler dequeues packets from the N queues in finalization time order. Such finalization time is estimated by an approximation to the theoretical model Weighted bit-by-bit round robin. This model, loops over the queues using RR algorithm outputting one bit at a time. Whenever a packet is complete, is dequeued and sent. Note that larger packets will have to wait in queue more than smaller ones. This theoretical approximation is not directly applicable as reassembling all the previously extracted bits of a packet would be prohibitive in terms of processing time and power.

For an output link of capacity C, the output rate of class i with the weight w_i is denoted by:

$$C_i = \frac{Cw_i}{w_1 + w_2 + ... + w_N} \tag{2.8}$$

This discipline solves the problems presented by FQ and allows a differentiated class distribution of the output bandwidth. In a situation in which all queues have traffic, the bandwidth of each queue is limited by the Equation 2.8 which imposes a limitation when measuring bandwidth in systems with this type of queuing.

Weighted Round Robin (WRR)

To solve the FQ problem of the traffic that demands a certain amount of bandwidth, WRR discipline also known as Class-Based Queuing (CBQ) is used. This method divides the input traffic into m different classes according to its bandwidth requirements. Each of these classes has an associated weight whose sum is 100%. Inside each i class, N_i FQ queues exist.

Every time a packet is to be dequeued, the scheduler visits the m classes in RR order. Inside each class the FQ queues are also visited in RR order. The individual weight of every FQ queue j inside the class i is determined by the next equation:

$$W_{ij} = W_i \times \frac{1}{N_i}; j = 0, 1, ...N_i \tag{2.9}$$

As in the previous case when all queues are full, the bandwidth of specific traffic may be limited due to the weight of the queue containing it and adversely affect the link bandwidth measurements. Although the traffic bandwidth requirements distribution problem is solved, this method does not solve the imbalance produced by the difference in the sizes of the packets.

Shaped Round Robin (SRR)

SRR is a discipline adopted by some router vendors as Cisco which is based on the specification of the packet extraction rate from a specific queue. Using this technique, the bandwidth assigned to each queue is strictly controlled. To control the bandwidth, the scheduler visits the queues in RR order and establishes time intervals in which packets can be extracted, adjusting thus, the maximum per-queue bandwidth. SRR has also a more relaxed working mode also known as shared mode. Such mode allows the use of the bandwidth assigned to other queues in case these are empty. On the contrary, the standard mode known as shaped mode does not allow this behavior. Shared mode is used when the maximum efficiency is to be obtained from a queue system as the idle queues may be used to store the traffic excess from other queues. Shaped mode is used when a strict maximum bandwidth limit is needed to be established on certain queues. This method can be used together with other previously applied bandwidth control methods such as traffic shaping or traffic policing.

2.2 Network Measurements

One key aspect of PM, is the management and handling of data derived and used in network measurement process. This section provides a brief overview of concepts and technologies related to network measurements both active and passive.

2.2.1 Flows and Sessions

A flow is defined as set of packets that share the same source and destination IP addresses, source and destination ports and transport protocol identification. A session or bidirectional flow is defined as a set of packets that share same permutable source and destination IP addresses, source and destination ports and transport protocol identification. Note that usually, a bidirectional flow or session is composed by two flows, one per direction. For example, one TCP connection can be grouped into a unique session or a flow pair. From now on the term session will be used to describe a bidirectional flow while the term flow will be used to describe a unidirectional flow.

Both sessions and flows are interesting from measurement and monitoring point of view. For instance, a monitoring system may collect flow records and generate statistics to describe and characterize the monitored network. In this light several standards and de-facto standards are being developed and used nowadays. Following, some of these standards are described for the sake of completeness.

NetFlow

NetFlow is a Cisco proprietary protocol designed to collect statistics and monitor IP traffic traversing routers and switches. Other vendors had acquired this protocol as a de-facto standard in their equipments. NetFlow v5, is the most extended and simpler version. In NetFlow v5 a flow is defined as a set of packets sharing a 7-tuple, namely: source and destination IP addresses, IP protocol, source and destination ports (supposing either TCP or UDP packets), ingress interface and IP Type of Service (ToS). Equipments implementing NetFlow maintain an in-memory table containing all active flows. Each flow entry contains the 7-tuple information as well as number of bytes and packets corresponding to such flow and first and last packet timestamps.

Each time a packet arrives to a router or switch that implements NetFlow, the packet is processed and its 7-tuple is compared with the in-memory flow table. If the flow entry does not exist and the equipment has memory available, a new flow entry is created. In case that the flow does not exist but there is no space available, one used entry is exported and its memory space is reused. One flow may be exported if:

- Flow has not been updated in a determined period of time. By default this time is set to 15 seconds on Cisco routers and switches.
- Flow has been active for a long time period. By default this time is set to 30 minutes on Cisco routers and switches.
- In a TCP flow, a FIN or RST packet has been received.

Each time a flow is exported, a NetFlow record is generated and sent to a collector which may process it. Each flow record contains the 7-tuple information as well as start and end flow timestamp and number of packets and bytes in the flow. Newer versions of NetFlow such as v9 have been used as basis for standardization of other protocols as in the case of Internet Protocol Flow Information eXport (IPFIX). More detailed description of NetFlow records can be found on [Cla04]. Note that, when monitoring high-speed links, all packets cannot be processed for NetFlow table update due to the high memory and processing power needed. In such case, packet sampling may be applied. In the case of routers and switches, sampling is applied in a deterministic way selecting one packet per each received N packets. Sampling traffic may lead to wrong flow statistics and problems that must be taken into account when using record information for monitoring and analysis purposes [Duf04, PRTV10, dRCGDA13]. NetFlow records are commonly used on passive network monitoring and analysis due to the simplicity and widespread deployment of Netflow-capable routers.

IPFIX

IPFIX [QZCZ04, Cla08, QBC$^+$08, TB08] is an Internet Engineering Task Force (IETF) protocol created to establish a common IP flow information export and data definition mechanism. This standard defines not only the type and format of transferred information and data from an exporter to a collector but also the protocol and communication mechanisms. IPFIX is based on NetFlow v9. IPFIX redefines the concept of IP flow extending the classical NetFlow definition to include the idea of a function that is applied to each packet to determine whether the packet belongs to a flow or not. The function may be applied to:

1. One or more packet header fields, transport header fields or application header fields. For example, a function can be applied over the 5-tuple classical fields (source and destination IP addresses, source and destination ports and transport protocol) and also over a Real-time Transport Protocol (RTP) header field defining thus, a RTP flow.

2. One or more intrinsic packet characteristics. For example, a function can be applied over Multi-Protocol Label Switching (MPLS) labels to group all the packets with the same labels in a unique flow.

3. One or more fields derived from packet processing. For instance packets may be grouped in a unique flow based on the next hop IP address.

The IPFIX flows contain usually: a flow key based on the previously described selection fields and a flow record containing measured properties of the flow such as number of bytes and packets. Flow records are generated by a metering process that receives packet headers, characteristics and treatment results and applies a set of functions. Such functions include packet header capturing, timestamping, sampling, classifying, and flow update and creation. As in NetFlow case, when flows expire must be exported. This exportation is done using IPFIX protocol messages which make use of templates to define and packetize the flow information and statistics. Custom templates can be added to extend the protocol in a simple and standard way positioning IPFIX as one of the most extended mechanisms to monitor IP traffic.

Multi Router Traffic Grapher (MRTG)

MRTG [OR98] is a link monitoring and traffic load measurement tool. This application allows the graphical representation of several network parameters such as link utilization. Additionally, MRTG has evolved allowing the measurement and representation of almost any kind of parameter. The tool periodically requests the value of the monitored parameters and graphs them. By default, parameter value requests are done in five minutes intervals but allows data aggregation with different granularities such as weeks, months or years. Typically, MRTG has two working modes. The first one is based on Simple Network Management Protocol (SNMP). In this working mode, the MRTG process, constructs and sends a SNMP request containing the Object IDentifier (OID) of the resource that is to be monitored. The response generated by the SNMP agent is received and stored in a measurement database and using all the measurement data a HyperText Markup Language (HTML) document is generated. Such document contains both graphs and information about the monitored resources. The second working mode is based on customs scripts. As in SNMP case, every so often, the customs scripts are executed and the results are stored in the measurement database. Such measurements can also be graphed as the SNMP response values.

MRTG tool[1] is written in Perl, and is available for the most popular operating systems —Windows, Linux and Mac. MRTG measurements are very useful when passive monitoring large-scale networks as gives an idea of network general load and utilization. Combining this methodology with NetFlow or IPFIX almost any traffic characteristic can be monitored and

[1] `http://oss.oetiker.ch/mrtg/`

represented.

2.2.2 Time synchronization

Network measurement process, independently whether is active or passive, needs time synchronization between all the elements to obtain reliable and valid results. This is especially important when measuring parameters such as OWD or RTT. Moreover, in high-speed environments, such need becomes critical as operation time intervals are very small and high accuracy is also needed. In this light, this section describes some time synchronization mechanisms commenting their major benefits and drawbacks. In this section only Network Time Protocol (NTP), Precision Time Protocol (PTP) and Global Positioning System (GPS) techniques are analyzed due to their importance and wide dissemination. From now own the terms accuracy and precision will be used. Thus, is important to correctly define what is understood by accuracy and precision. To this end, the definitions in [IS08] are used. Accuracy is defined as the mean time or frequency error between one target clock and a perfect reference clock over a time period. Precision is defined as the deviation of the error from the mean.

NTP

NTP [Mil85, MMBK10] is a client-server time synchronization protocol widely used on the Internet. Basically, NTP was designed to distribute and synchronize Coordinated Universal Time (UTC) time between different network points. NTP is based on a hierarchical organization defining three main clock strata levels namely:

- Stratum 0: This stratum is composed by high-precision clocks such as atomic clocks, GPS or radio clocks. Normally, such clocks are not directly connected to the network but connected to computers that act as NTP stratum 1 server.

- Stratum 1: This stratum is composed by computers attached to Stratum 0 devices. Such computers are known as time servers and provide time synchronization to Stratum 2 equipment via NTP.

- Stratum 2: This stratum is composed by computers that communicate with Stratum 1 computers and act as time servers for lower strata. Stratum 2 machines also communicate with other Stratum 2 equipments to obtain more precision and stability.

NTP uses a time correction algorithm to distribute the time across the network. Such algorithm is based on the RTT calculation. To calculate the RTT, the NTP client sends a packet timestamped with value t1. On the server side, client packet is timestamped upon receiving with value t2. The server sends a packet timestamped with value t3 that is received and timestamped on the client with value t4. The RTT is calculated as:

$$RTT = (t4 - t1) \tag{2.10}$$

And the clock time offset c as:

$$c = \frac{t2 - t1 + t3 - t4}{2} \tag{2.11}$$

Such formulas, implicitly assume that the delay in each direction is symmetric and that the drift rates of client and server clocks are small. Such assumptions may not be realistic in all scenarios and NTP mat have a systematic bias of half the RTT.

All modern operating systems implement NTP clients to keep local time synchronized. Usually, the local clocks are corrected several times a day and periodic NTP request are done. The achievable accuracy depends on both network delay variations and source clock precision. Typically, on the Internet the NTP accuracy ranges from 5 ms to 100 ms. NTP performs bad in presence of high network delays.

In [Min99], a wide NTP analysis is done and the results suggest that only the 10% of the analyzed NTP servers have network access delays higher than 100 ms with a mean of 33 ms, a median of 32 ms and a standard deviation 115 ms. Also the author of [Min99] points to a reduction of such times in comparison with previous studies due to the improvement of the networks quality. Additionally, the clock offset is analyzed on several hosts obtaining values of mean offset of 8.2 ms, median of 1.8 ms and standard deviation of 18 ms.

NTP is a good choice for time synchronization when ms accuracy is enough and a widely compatible and scalable system is needed. The main advantage is that NTP works over almost any kind of network as far as UDP is available.

PTP

PTP [IS08] is a network time synchronization protocol. Unlike NTP, PTP uses the master/slave architecture to maintain time synchronized between multiple equipments. Each machine in a network can act a either a master (time source) or a slave —time destination. The selection between master and slave is done dynamically along the synchronization process. Thus, a clock may act as master for a time period but if a more accurate clock is added to the network, it will change its role to slave transferring the master role to the new clock. Additionally another role called boundary clock is used. Boundary clocks are connected to two or more network segments and are used to bridge synchronization from one network segment to another. Note that each network segment can have its own master clock. A grandmaster clock is selected among all the master clocks to synchronize clocks located on different network segments. The protocol is conceived to be distributed and is based on the exchange of PTP timing messages between the grandmaster clock and the slaves using such timing information to adjust and compensate their clocks using the time of their master in the hierarchy.

The synchronization process is based on two stages comprising the establishment of master-slave hierarchy and the time synchronization itself. Each PTP node has its own state machine to determine whether it works as a master, as a slave or as a passive node. Passive nodes do not synchronize to any clock and do not provide time synchronization to other nodes. The selection of master and slave clocks is made using the Best Master Clock (BMC) algorithm. Such algorithm analyzes clocks data to select the best clock. The analyzed data is sent periodically in Announce messages by the grandmaster clock. The clock selection algorithm uses the next properties in the indicated order to make a decision of the best clock.

1. Priority 1: This parameter indicates the belonging of a clock to an ordered set of clocks from which the master should be selected. It can be set by the user.

2. Clock class: This parameter defines the clock International Atomic Time (TAI) traceability, that is, define whether or not the clock has a set of properties related to a TAI international standard by means of an unbroken chain of comparisons including stated uncertainties.

3. Clock accuracy: This parameter defines the clock accuracy.

4. Offset-scaled Log Variance: This parameter defines the stability of a clock along time.

5. Priority 2: This parameter establishes a fine-grained ordering among equivalent clocks.

6. Clock identity: A unique clock identifier.

Once master and slave clocks are selected the synchronization process starts. For every slave, the time offset with respect to the master clock is calculated following the next process:

1. The master sends a Sync message to the slave and stores the transmission time t1.

2. The slave receives the Sync message and stores the reception time t2.

3. The master sends t1 timestamp embedding it on both Sync and Follow_Up messages to the slaves.

4. The slave sends a Delay_Req message to the master and stores the sending time t3.

5. The master receives the Delay_Req message and stores the reception time t4.

6. The master sends to the slave the timestamp t4 embedding it in a Delay_Resp message.

When this message exchange is over, the slave has received t1, t2, t3 and t4. Using such timestamps, the offset may be calculated as well as the mean propagation time between two clocks. As in NTP case, the offset computation assumes that propagation times in each direction are equal. For asymmetric links there exists an error in the calculated clock offset. Using the previously stored times, the offset from master is calculated as:

$$o(t) = \frac{t2 - t1 - t4 - t3}{2} \tag{2.12}$$

Once the slave clock knows the offset can correct itself. The offset calculation is carried out in a relatively small time period and, thus, the calculated offset can be considered constant. Such calculation must be done repeatedly to correct the time drifts. Regarding accuracy, PTP can achieve clock synchronization with errors lower than 1 μs on Ethernet networks [IS08].

PTP emerged as an alternative for local systems where high-accuracy synchronization is needed and NTP does not provide enough accuracy. Also PTP is presented as an alternative to GPS synchronization since the latter adds per-node monetary costs increasing the deployment investments. Additionally, in some scenarios, the GPS signal may be inaccessible. This is the case of big data-centers which usually are located in basements or closed environments where the cost of transporting the signal is very high.

Although PTP originally used multicast UDP over IP messages, the protocol has evolved to support messages over Internet Protocol version 6 (IPv6) and also over Institute of Electrical and Electronics Engineers (IEEE) 802.3 Ethernet directly.

GPS

GPS is a satellite navigation system that can provide location and time information in an accurate way. In recent years this technology has gained tremendous popularity and has been widely disseminated in all kind equipments. Some examples are the GPS receivers installed on car navigation systems or the receiver included in modern commercial smartphones. GPS technology is based on the calculation of message propagation time and distance from satellites to receivers. To this end, both receivers and satellites are equipped with special hardware. In the case of satellites, atomic clocks are installed to perform the timing of the messages. On the receiver side accurate hardware clocks are installed.

Using this precise time characteristic provided by the GPS technology, several nodes can be synchronized without any communication installing GPS receivers on each node. Timing with GPS is a very good approach when synchronization systems based on packet transmission such as NTP or PTP are not allowed or the cost in terms of extra traffic is not assumable. The main disadvantage of such approach is the increment of monetary cost of receivers' deployment and the impossibility, in some cases, to carry the signal to the receivers.

Focusing on traffic monitoring, several projects and works have been carried out using GPS techniques [MMI$^+$05, SMRD06, LAM$^+$11, FDL$^+$01]. Such approaches make use of GPS receivers to build a network measurement and control infrastructure based on accurate timestamping. Depending on the application the GPS receiver may be connected to specific network hardware such as Endace DAG[2] or NetFPGA[3] cards. Other applications make use of commodity hardware together with specific software to synchronize the clock using a GPS [SBDR10].

The average accuracy achieved by a commercial GPS receiver is in the bounds of 200 ns [EGE02]. Additionally, extra processing or signal quality may degrade such figure to 500 ns. In any case, the accuracy obtained by this

[2] `http://www.endace.com/`

[3] `http://netfpga.org/`

technology outperforms all the previously presented methodologies at the expense of increasing the deployment cost. However, hybrid solutions had been developed to reduce such cost keeping a reasonable clock accuracy [HG03].

Chapter 3

Active Measurements

This chapter provides the background and revises the most relevant works related to active network measurements. In addition, new active measurement techniques are proposed and analyzed in order to provide a better integration into Proactive Monitoring (PM) systems. The structure of the chapter is as follows. First, a brief description about active measurement methodology is done in Section 3.1. Then, in Section 3.2 an overview of active measurement techniques is done dividing the existing methods into File-Transfer based and Packet Pair based. These methods lack of precision estimating Quality of Service (QoS) parameters when monitoring heterogeneous networks on general purpose machines. In Section 3.3 such problem is addressed taking into account the influence of external parameters such as Central Processing Unit (CPU) and memory load, concurrent traffic or the application of QoS mechanisms among others. In the survey of Section 3.3, several algorithms are proposed to address the previously presented problems. To finalize, some important ideas are presented in Section 3.4 as conclusion.

3.1 Introduction

Active measurement techniques are based on the idea of injecting traffic into a network to measure its characteristics. Such approach can be applied to large scale networks either end-to-end or per link to obtain reliable statistics and determine the quality of the observed network segment. On the one hand, active measurements are network-invasive due to their very nature. Such characteristic is not desirable in some scenarios as the measured link behavior is being influenced by the measurement traffic. On the other hand, in some cases QoS parameters must be estimated accurately in a given time period. For example, Service-Level Agreement (SLA) validation process over a specific link or path requires active measurement to obtain parameters such as path or link capacity, One-Way Delay (OWD) or packet loss. Some of these parameters such as link capacity cannot be estimated in a passive way as links are not usually fully-loaded. In other cases the estimation of the parameters could be done in a less accurate way. For example, to passively estimate OWD packet correlation is needed besides end-to-end time synchronization. Additionally, some network behaviors can only be correctly observed when adding load to the links. This is the case of traffic shaping and policing parameters estimation for SLA verification. Due to the aforementioned advantages, active measurements must be taken into account when monitoring networks in a proactive way to improve passive estimations and detect specific problems. Nevertheless, the implementation of active methodologies on real systems brings up several problems such as the self-induced traffic, the CPU load, the Interrupt Coalescence (IC) or the scheduling policies applied by intermediate routers. Such problems must be taken into account in order to obtain reliable results and a full understanding of measured network. Depending on the type of generated traffic and the methodology used, three main categories can be distinguished: Bulk-Data transfer, Packet-Pair and Packet-Train. Every, methodology has its own advantages and disadvantages. In Section 3.2 these methods are described in depth.

3.2 Active Techniques

Active measurement techniques aim to measure several network characteristics by generating traffic and observing its behavior as the traffic traverses the network. The major drawback of this method is its intrusiveness as extra traffic must be generated. In some situations such intrusiveness is desirable, for instance when trying to stress a network to characterize existence of misused or available resources. Active techniques may be used periodically to test and analyze the networks. Such process leads to a stratification of the active measurement to characterize the network behavior along the time depending on the state of the network. Active techniques can be divided into two big groups mainly: File-Transfer/Bulk Data Transfer techniques and Packet-Pair techniques. Such division is based on the technique used to generate traffic and subsequently analyze it to estimate the QoS parameters. In this section, several techniques and tools are commented composing a brief state of the art of the active measurement methodologies including some comments about their strengths and weaknesses as well as how some QoS parameters are estimated.

3.2.1 File-Transfer

This measurement method formally defined by European Telecommunications Standards Institute (ETSI) EG 202 057-4 [Ins08] aims to estimate QoS parameters using a HyperText Transfer Protocol (HTTP) file transfer. The transfer must be done querying a dedicated test server and downloading a file which must be, in size, eight times the nominal bandwidth of the measured link. For example, on a 1 Mbps link the test file size must be 10^6 Bytes (8 Mbits) in size. Such file must be randomly generated to avoid any web server optimization. Furthermore, an integrity check must be performed by means of a SHA digest associated with the file. To calculate the QoS both server response and file download times are calculated. To estimate the bandwidth of the measured link or end-to-end path, the next formula is used:

$$BW = \frac{8N}{\tau_f - \tau_i}(Mbps) \tag{3.1}$$

where N is the size of the downloaded file expressed in bytes and τ_f,τ_i represent the file download finalization and start time, expressed in μs, respectively. The download time can be measured at different protocol levels (Physical Layer, IP, TCP) and application level is considered in these measurements [AAMD06].

According to the ETSI standard, the file, in size, eight times the nominal speed of the link must be downloaded in about eight seconds to verify that the measured bandwidth is about 100% the nominal bandwidth. Several commercial solutions such as Speedtest[1] use this method to provide an estimation of the bandwidth. Such platforms perform several download trials in order to provide a better estimation.

Other parameter that can be estimated using this method is Round-Trip Time (RTT). In this case, the RTT is estimated as the average HTTP server response time or the Transmission Control Protocol (TCP) RTT. Additionally, packet loss rate can also be estimated using this method [BMSDM12]. The OWD can be estimated as the half of RTT as stated in the ETSI guide [Ins08]. Such estimation is only applicable when the measured link or end-to-end path is symmetric. Even more, if the measured link or end-to-end path is symmetric it cannot be assumed that the delays are similar in both directions.

The main advantage of this method is that the measurement is performed at the user level and it provides a closest idea of the user experience. File-download techniques are very simple to implement but they have twofold drawbacks. On one hand, the download times are large, in the order of 8 seconds because the downloaded file is 8 times the nominal bandwidth of the measured link, according to the ETSI guide [Ins08]. On the other hand, the influence of cross-traffic is high. This is expected because TCP performs a rate adjustment that depends on the number of concurrent connections in the bottleneck link. Whenever a TCP connection is established, for instance a HTTP connection, a congestion-window mechanism is used to limit the amount of data that can be sent without receiving an acknowledgment. Due to the slow start method, such congestion window is increased until losses are detected or ACK reception timeouts expire. At this point, the mechanism reduces the congestion windows until either the minimum value of the window is reached or no losses are detected. Once this state is reached, the congestion window is increased again and the previous processes are repeated. This

[1] `www.speedtest.net`

mechanism produces a saw-tooth behavior as Figure 3.1 for TCP Reno — W/2.

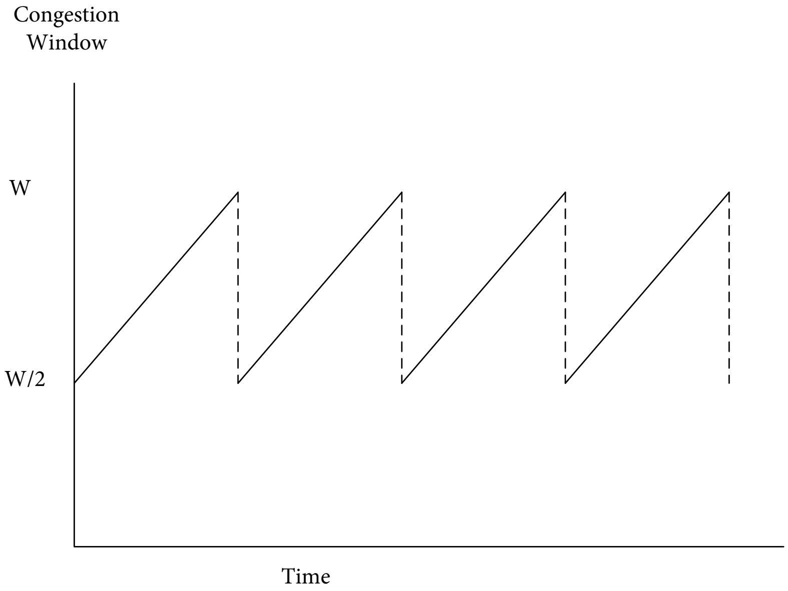


Figure 3.1: Saw-Tooth behavior produced by TCP congestion mechanism

Given that a file download measurement can be executed in general purpose machines, other concurrent TCP and User Datagram Protocol (UDP) connections may exist. Equations 3.2 and 3.3 [HAN02] show that both the bandwidth of a single TCP connection and the aggregate bandwidth of a set of TCP connections depend of the loss probability p_i of each connection. If a measurement is carried out making use of TCP and either UDP or TCP concurrent intensive traffic exists, losses can be detected and the effective bandwidth of the measurement connection decreases.

$$BW \leq \frac{MSS}{RTT}\frac{C}{\sqrt{p}} \tag{3.2}$$

$$BW_{agg} \leq \frac{MSS}{RTT}\left[\frac{1}{\sqrt{p_1}} + \frac{1}{\sqrt{p_2}} + ... + \frac{1}{\sqrt{p_n}}\right] \tag{3.3}$$

Additionally, if two TCP connections are sequentially created the first one will be favored due to the slow start mechanism. TCP connections must spin out some time until an equitable bandwidth distribution is obtained. In some cases the time needed to reach such equilibrium exceeds the time required to perform a file download test.

Other disadvantage of this measurement methodology, is that is affected by CPU and memory load, providing distorted results as shown in Section 3.3

3.2.2 Packet-Pair

Packet-pair [Jac88, Kes91, Pax96, DRM04, DRM01] is an active measurement method based on sending multiple packet pairs from a source to a destination point to calculate QoS parameters. Each pair is composed by equally sized packets sent back-to-back, this is, at maximum allowed speed in a link or end-to-end path. The pair dispersion in a specific link is defined as the time between the last bit of the first packet and the last bit of the second packet. The pair dispersion in the destination if no interfering traffic is present is defined by:

$$\Delta_r = \frac{L}{\min_{i=0,..,H} C_i} = \frac{L}{C} \tag{3.4}$$
$$H = \#\ \text{hops}$$

where L represents the size of the packet expressed in bits and C_i is the capacity of link i in an end-to-end path of H hops expressed in Mbps. Δ_r is the dispersion or inter-arrival expressed in μs. To calculate the capacity of the end-to-end path Equation 3.5 is used.

$$C = \frac{L}{\Delta_r} \tag{3.5}$$

Figure 3.2 shows the behavior of this measurement method.

These types of methods allow the estimation of other QoS metrics such as OWD, delay variation or Packet Loss Rate (PLR). To calculate the OWD packets are timestamped both on departure and arrival time. Calculating the difference between the two timestamps the OWD may be estimated. To calculate the delay variations, the algorithms described in Section 2.1.1 can

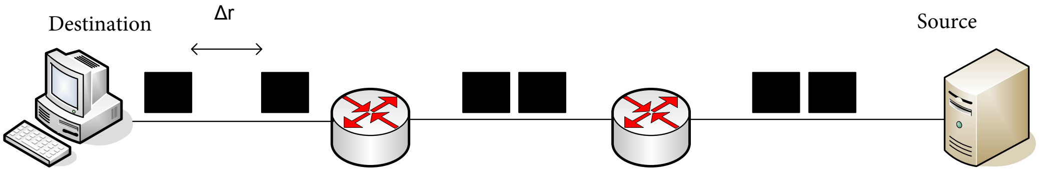


Figure 3.2: Packet Pair method

be used. As in the case of OWD, to calculate PLR, packets are sequentially numbered on the source measurement point and checked for gaps at destination measurement point. Calculating the ratio between the received and sent packets the PLR can be estimated. To calculate the RTT the measurement must be carried out in both directions estimating the two OWD. Usually, packet-pair methods are implemented using UDP as transport protocol, unlike the file-download technique, which uses TCP.

One advantage of these types of methods is that measurements take little time. For instance, a bandwidth measurement in a 10 Mbps link can take less than 1 ms —i.e. the transmission of two packets with size 600 bytes in a 10 Mbps link takes $\frac{2\times600\times8}{10} = 910$ μs<1 ms. However, the packet-pair method accuracy is highly dependent on the clock resolution for sending and receiving packets. Usually, measurements are performed at the application layer and the clock resolution is the one provided by the operating system. To mitigate these uncertainties either Libpcap [2] or WinPcap [3] drivers can be used. Such drivers provide a packet timestamp in the μs timescale. Additionally, if more precision is required, some kernel-space solutions such as Pktgen [Ols05] may be used. In any case, to obtain reliable results source and destination measurement clocks must be synchronized using techniques such as Network Time Protocol (NTP), Precision Time Protocol (PTP) or Global Positioning System (GPS) depending on the required precision.

The major drawback of packet pair techniques is the impact of interfering traffic during the measurement period. In a realistic scenario, the measured network transports traffic that can concurrently traverse the measured link along with the measurement traffic. Such interfering traffic may influence

[2] www.tcpdump.org

[3] www.winpcap.org

the inter-arrival time measured by the packet-pair methods in two ways:

- Inter-arrival Expansion: when one or more interfering packets slip in between two measurement packets, the measured inter-arrival Δ_r increases and the estimated bandwidth decreases. This effect is shown on Figure 3.3.

- Inter-arrival Compression: when two measurement packets with inter-arrival Δ_{rt} are queued together and interfering traffic fills the router queue, the two measurement packets are dequeued as fast as the outer link allows. Thus the measurement packets would have an inter-arrival Δ_r that could be smaller than the one measured at previous hop increasing the estimated bandwidth. This effect is shown on Figure 3.4

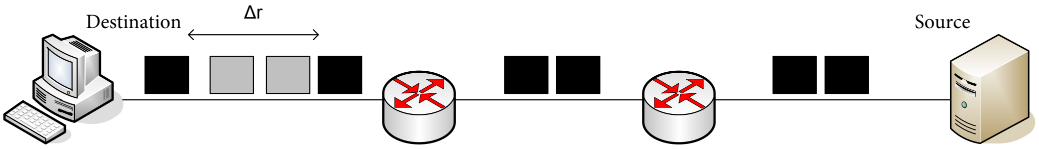


Figure 3.3: Inter-arrival expansion due to interfering traffic

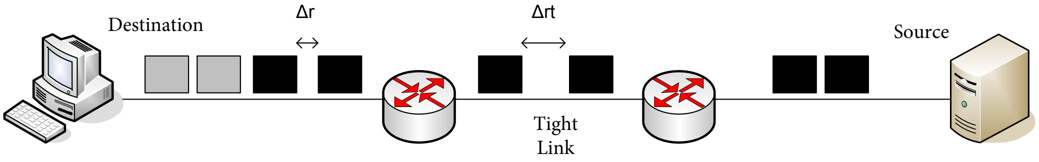


Figure 3.4: Inter-arrival compression due to interfering traffic

Other adverse effect that affects packet pair methods is the existence of IC. This mechanism produces an overestimation in the measured bandwidth. Such effect is covered in Section 3.3.

3.2.3 Packet Train

The packet-pair method is very sensitive to cross-traffic, which motivates the use of a packet train [DRM01, Joh03, MBG00, MBG02]. Using packet pair method, there is a single inter-packet gap, which may be easily filled by interfering traffic. In order to decrease the chances of cross-traffic slip in the inter-packet gap, a packet-train of N packets is sent instead. The dispersion Δ_R in this case is calculated as the addition of individual pair dispersions:

$$\Delta_R(N) = \sum_{k=1}^{N} \Delta_r(i) \tag{3.6}$$

The dispersion factor D is defined as:

$$D = \frac{(N-1)L}{\Delta_R(N)} \tag{3.7}$$

where L is the packet size expressed in bits.

In case that there is no interfering traffic, the dispersion factor is equal to the link or end-to-end path capacity. Otherwise, the dispersion factor only presents an approximation to the average link or end-to-end path capacity. As N grows, the dispersion rate variance is reduced and the estimation approximates to real capacity. Figure 3.5 shows the behavior of packet train method.

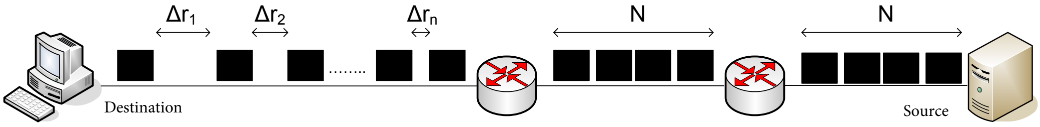


Figure 3.5: Packet Train method

Packet-trains constitute a robust technique against cross-traffic albeit not totally immune. As the number of packets in the train grows, the probability of every gap being occupied with cross-traffic decreases. However, large packet-trains have negative effects, as shown in [DRM04]. As it turns out, large packet-trains are intrusive. For instance, a packet-train of $N = 1000$ Ethernet-MTU-sized packets is 1.5 MB worth of traffic. Since the packet-train technique is based on link flooding, 1000 packets of 1500 Bytes saturate a 10 Mbps link for 1.2 s. This can produce packet loss or throughput degradation to other connections which may be active in that moment. Thus, there is a trade-off between cross-traffic immunity and intrusiveness.

3.3 Active Measurement Techniques Problems

Active measurement methodologies provide reliable results but some problems may affect the quality of the obtained results. In this section, such problems are analyzed and its impact over the results is quantified providing a general overview of several aspects that must be taken into account when actively measuring. After an extensive study, five problems that impact on the active measurement results have been identified. Such problems are: CPU and memory load, self-induced traffic, IC and QoS assurance mechanisms.

3.3.1 Testbed Description

Active measurement techniques have been assessed in a local area network. Figure 3.6 shows the testbed topology. Link 3 is a link with variable speed. It represents the bottleneck link. The switch represents the "core" network —assumed to operate at 1 Gb/s. There is a cross-traffic source which sends packets to the measurement PC. Note that this is traffic generated from other concurrent applications because it is a download from the measurement PC which is simultaneous with the measurement.

The equipment used in the testbed is the following: the client processor is a Pentium Dual Core 1.80GHz 32 bits and it has Linux, Windows and Mac OS operating systems installed; the measurement server processor is an Intel Xeon Quad Core 2.33GHz 64 bits with Linux operating system installed; the cross-traffic source is an AMD Geode LX 700 433MHz 32 bits and it has Linux operating system installed. All the experiments were done in Windows (XP and Vista), Linux (Ubuntu) and Mac OS on the client.

The bottleneck link is simulated using Linux `tc`[4] on a PC, which acts as a router. The bottleneck links speeds are 900 Kb/s, 6 Mb/s, 12 Mb/s, which are typical access link speeds for residential users and small companies,

[4] `http://linux.die.net/man/8/tc`

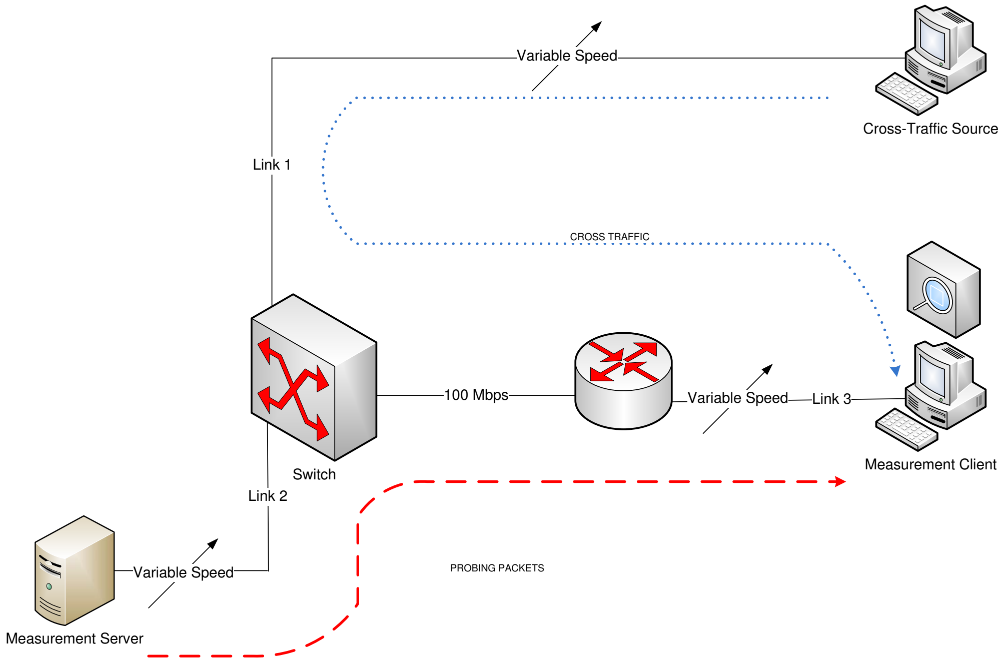


Figure 3.6: Testbed Topology

and 100 Mb/s, which is available in the access link in more of 4 millions of households in Japan by means of FTTH [CFEK06].

On the controlled testbed, the generated cross-traffic was injected with `hping2`[5] for UDP traffic and `wget`[6] for TCP traffic. For the generation of randomly delayed cross-traffic, Linux `tc` tool with `netem` extension was used to generate Pareto-Normal [FAM01, ZH07, ITF04] distributed inter-departure packet times.

To generate CPU load, a custom C language program which makes floating point and file processing operations was used. To generate memory load a custom C program was used to allocate memory and keep it booked during the tests. The pseudocode is shown in the Appendix A. Also a similar Java program was used to generate CPU and memory load. These programs emulate different concurrent applications.

3.3.2 CPU and memory load

Most part of the measurement processes are executed in general purpose machines that run many other processes concurrently. Thus, is important to quantify and analyze the impact of the CPU load over the measurement process. First what is understood by CPU usage (cpu_frac) of a measurement process must be formally defined:

$$cpu\_frac = \frac{t_{proc\_end} - t_{proc\_begin}}{t_{total\_end} - t_{total\_begin}} \tag{3.8}$$

where t_{proc_end} and t_{proc_begin} are the total cumulative processor times which are devoted to the measurement process at the end and at the beginning of the measurement, respectively. Similarly, t_{total_end} and t_{total_begin} are the total cumulative processor times at the end and at the beginning of the measurement, for all the applications including the measurement process. Therefore, cpu_frac represents the fraction of time that the measurement process is being executed in the processor.

[5] `http://linux.die.net/man/8/hping2`

[6] `http://linux.die.net/man/1/wget`

When the measurement system is loaded with many concurrent applications running, the measurement process will be less time in the processor —and cpu_frac will have values near zero. On the other hand, if the measurement system is not loaded, with a few or no concurrent applications running, the measurement process will be more time in the processor —and cpu_frac will have greater values. Furthermore, the scales of cpu_frac are different between different operating systems due to differences between processor schedulers. Moreover, it is not possible to obtain all values of cpu_frac in the range from 0 to 1. For example, the operating systems does not allocate 100% of processing time to a single process, even if the processor is idle.

Note that the total CPU time of a process can be split into the following partial CPU times:

- User Time: specifies the number of processor counter ticks spent in user-level processes.
- Kernel Time: specifies the number of processor counter ticks spent in kernel-level processes and in transitions such as context switches.
- Interrupt Time: specifies the number of processor counter ticks spent in detecting and handling system interrupts —hardware and software.
- Idle Time: specifies the number of processor counter ticks spent in the idle process. The Idle process is a simple thread which is used to calculate the percentage of free CPU time in a system. This process is always running.

The total process CPU time (t_{proc}) used on this analysis is defined as the addition of all the partial CPU times used by the measurement process during the measurement —i.e. user, kernel, interrupt.

In the same way memory load can be formally defined as:

$$mem = \frac{\text{free memory}}{\text{total memory}} \tag{3.9}$$

where mem simply represents the fraction of free memory on the system.

To analyze the impact of CPU and memory load over the quality of an active measurement, several measurements have been done while increasing the CPU and memory load following the process commented in Section 3.3.1. The measurements have been carried out in the 6 Mb/s and 100 Mb/s scenarios as representatives of domestic and high-speed scenarios.

Figure 3.7 shows the scatter plots of the bandwidth measurement and the CPU and memory load parameters on Windows. This figure represents the degree of correlation between each pair of variables. For instance, the upper figure shows the bandwidth measurement as a function of the cpu_frac parameter.

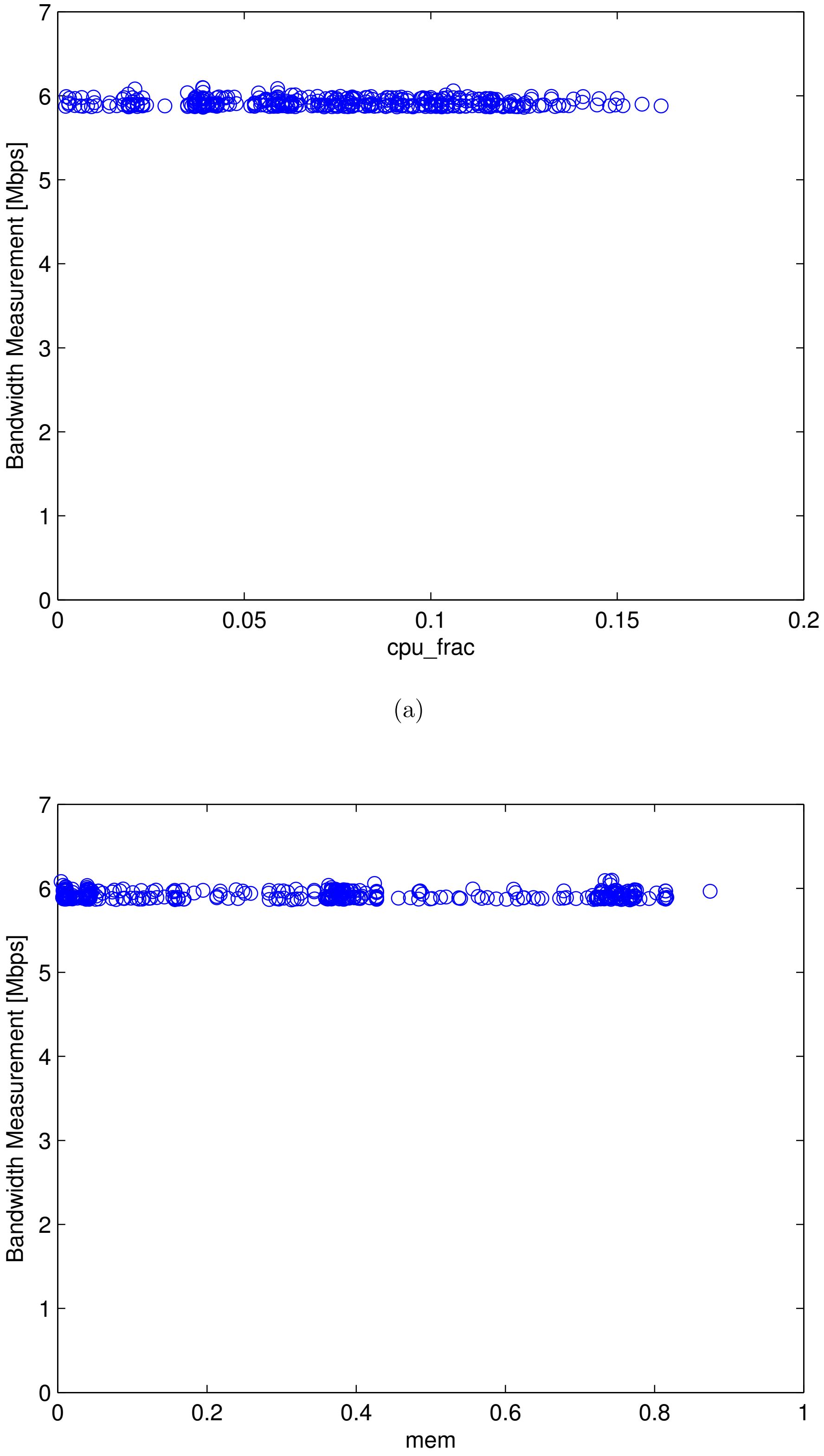
In this case (when the link capacity is 6 Mb/s) it can be observed that there is no correlation between the load (CPU or memory) and the bandwidth measurement. This fact can be statistically proved by means of a ANalysis Of VAriance (ANOVA) test.

The experiment is repeated changing the link capacity to 100 Mb/s. Fig. 3.8 shows the scatter plots of the measured bandwidth and the CPU and memory load parameters. In this case, it can be observed that there is high correlation between the CPU load and the bandwidth measurement. However, it can also be seen that the memory load seems to be uncorrelated with the bandwidth measurement and the CPU load.

In order to model the effects of the CPU and memory load on the measurement, the following linear model has been considered:

$$y = \beta_0 + \beta_1 \cdot cpu\_frac + \beta_2 \cdot mem + \epsilon \tag{3.10}$$

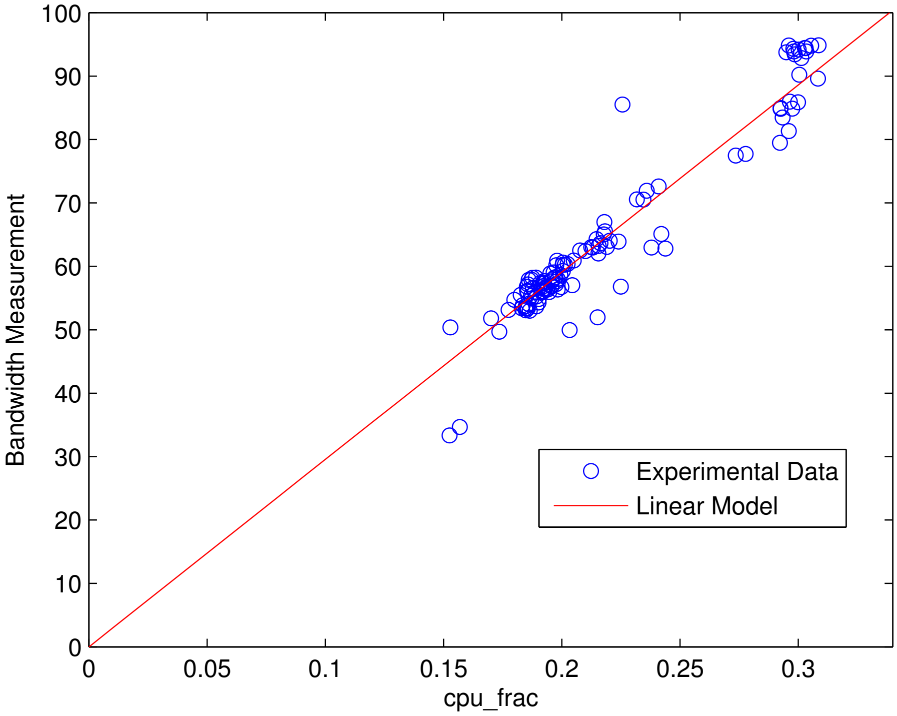
where y is the bandwidth measurement and $\epsilon \sim N(0, \sigma^2)$ represents the measurement error. Note that β_1 and β_2 values represent the influence of CPU and memory load, respectively, on the bandwidth measure. After applying linear regression, Table 3.1 shows the regression coefficients obtained, $\hat{\beta}_i$, for each variable explanatory, the standard deviation estimated and the p-value, for the test with null hypothesis $\beta_i = 0, i = 0, 1, 2$. Furthermore, the last row of the table provides us the R^2 value which represents the amount of variance explained by the linear model. The results show that the memory



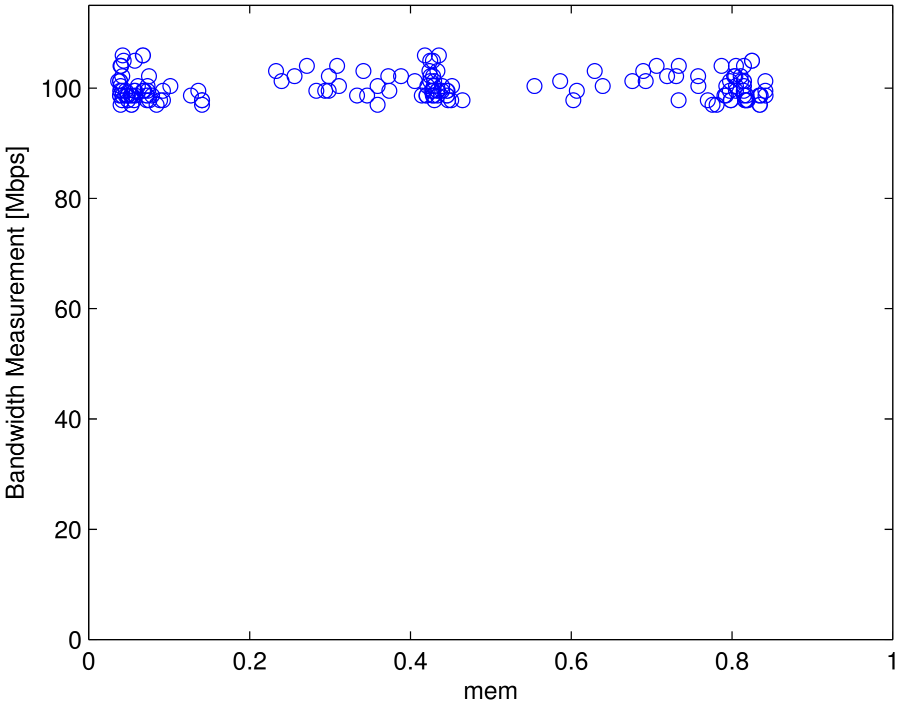
(a)

(b)

Figure 3.7: Bandwidth measurement versus system load parameters with file-download technique on Windows using TCP as transport protocol. Link capacity 6 Mb/s (a) cpu_frac (b) mem



(a)



(b)

Figure 3.8: Bandwidth measurement versus system load parameters with file-download technique on Windows using TCP as transport protocol. Link capacity 100 Mb/s (a) cpu_frac (b) mem

Table 3.1: Regression coefficients for linear model considering CPU and memory load with file-download technique on Windows using TCP as transport protocol. The link capacity is equal to 100 Mb/s

	Coefficient	**Sd. error**	**p-value**
$\hat{\beta}_0$	-3.4789	2.0154	0.087
$\hat{\beta}_1$	311.3522	8.3068	$< 10^{-16}$
$\hat{\beta}_2$	-0.4605	2.1346	0.830
R^2	0.9233		

Table 3.2: Regression coefficients for linear model considering CPU load only, with file-download technique on Windows using TCP as transport protocol. The link capacity is equal to 100 Mb/s

	Coefficient	**Sd. error**	**p-value**
$\hat{\beta}$	295.29	1.61	$< 10^{-16}$
R^2	0.9965		

load does not have influence on the measurement whereas the CPU load is very influential on the bandwidth measurement. It is also worth remarking that the coefficient β_0 is not statistically significant —at common significance level $\alpha = 0.05$. This fact is coherent in the sense of *if the computer does not give any processing time to the measurement process* (*$cpu\_frac = 0$*)*, the measured bandwidth is zero.*

Therefore, a simplified linear model, which only considers the CPU load, is more advisable:

$$y = \beta \cdot cpu\_frac + \epsilon \tag{3.11}$$

where y is the bandwidth measurement and $\epsilon \sim N(0, \sigma^2)$ represents the measurement error. After applying linear regression, Table 3.2 shows the summary of the linear regression model obtained. Note that the value of β represents the slope in the linear regression model.

This simplified model explains the 99.65% of the variance of the bandwidth measurement. It is worth remarking that the simplified model explains a greater amount of the variance of the measurement than the previous model —which explains the 92.33% of the variance.

Figure 3.9 shows the linear regression estimated for the bandwidth measurement as a function of *cpu_frac*. Note that process time ratio below 30% for the measurement application imply severe underestimation error —up to 90%.

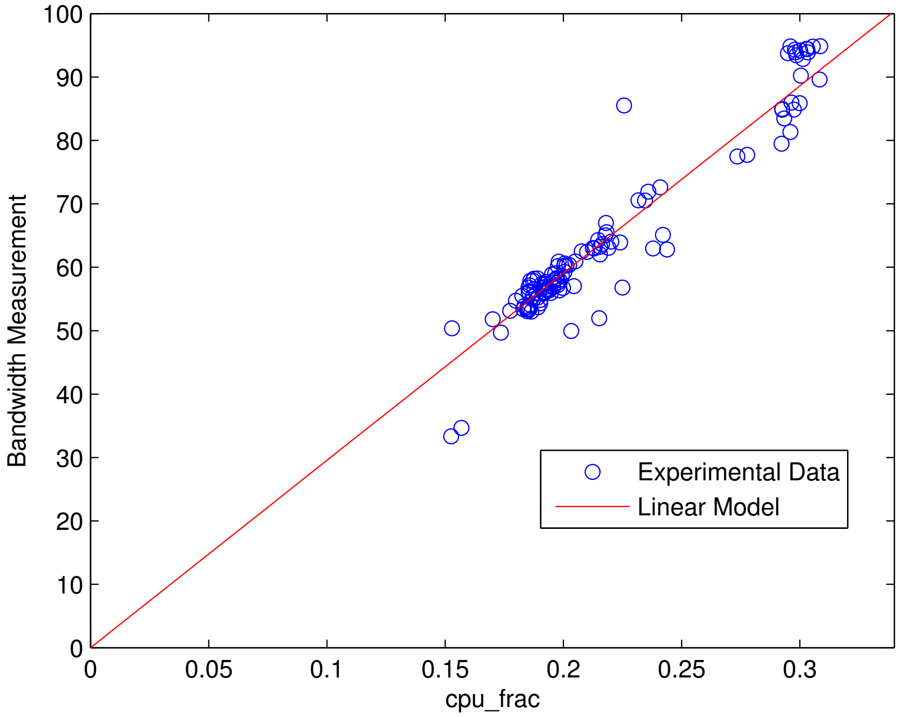


Figure 3.9: Linear regression model adjusted for the bandwidth measurement with file-download technique on Windows using TCP as transport protocol. The link capacity is equal to 100 Mb/s

The previous experiments were performed on Windows. The experiments were repeated on Linux, and the results obtained were very similar: linear dependence between *cpu_frac* and the bandwidth measurement and uncorrelation between the memory load and the bandwidth measurement, although there are slight differences in the slope. Fig. 3.10 shows the scatter plot between *cpu_frac* and the bandwidth measurement in this case. Note that the scales of *cpu_frac* are different between Linux and Windows due to differences between processor schedulers. Experiments were also performed on Mac OS X, obtaining similar results. Fig. 3.11 shows the linear regression estimated for Mac OS X. It is worth remarking that it is more difficult to degrade CPU load smoothly in Mac OS X than in Windows or Linux.

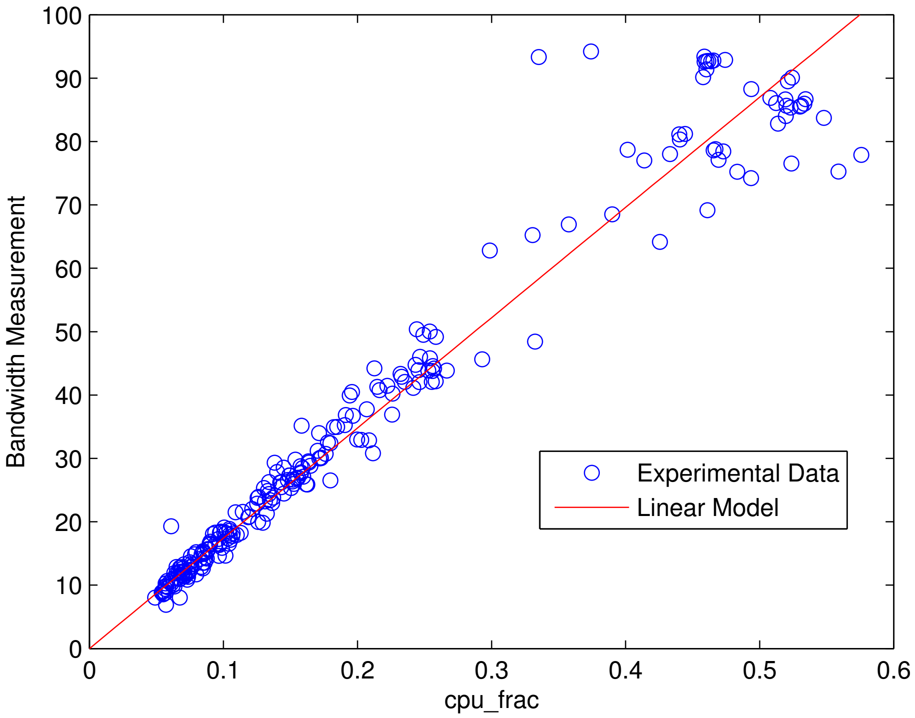


Figure 3.10: Linear regression model adjusted for the bandwidth measurement with file-download technique on Linux using TCP as transport protocol. The link capacity is equal to 100 Mb/s

But why is the TCP-based measurement influenced by the CPU load at 100 Mb/s? As it turns out, TCP needs more processing capacity than UDP due to control flow mechanisms, ACK sending, etc. When the link

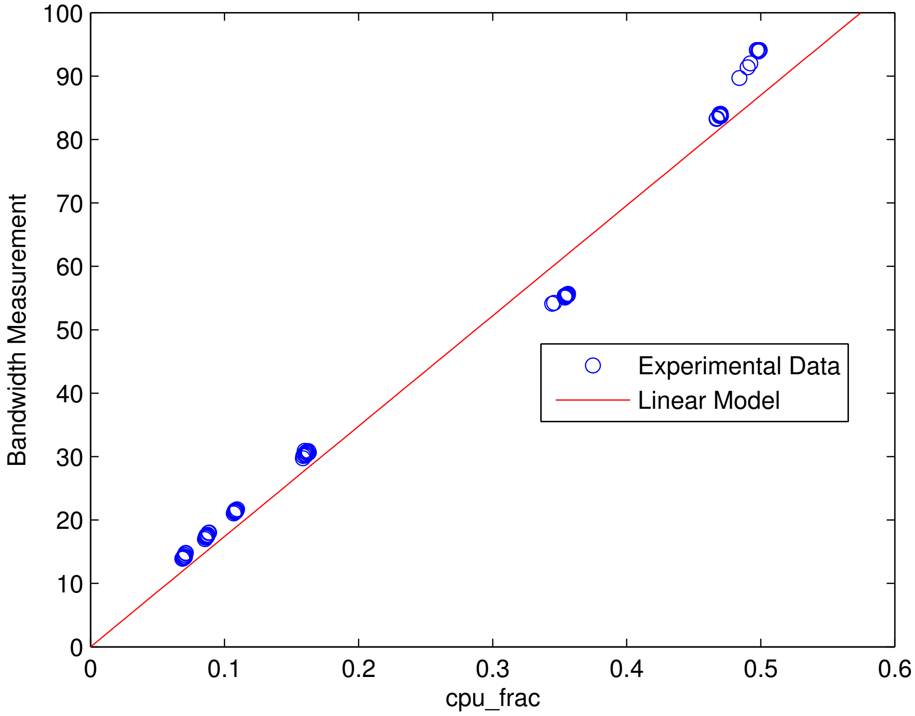


Figure 3.11: Linear regression model adjusted for the bandwidth measurement with file-download technique on Mac OS X using TCP as transport protocol. The link capacity is equal to 100 Mb/s

capacity is high (100 Mb/s) and the measurement system is loaded, there is not enough processing time to perform TCP processing. It is worth noting that the inter-arrival time (for a packet size of 1500 Bytes) at 100 Mb/s is 120 μs. On the other hand, when the speed link is low (6 Mb/s) the inter-packet gap is large enough to perform TCP operations even if the CPU is fully loaded —the inter-arrival time for a packet size of 1500 Bytes at 6 Mb/s is 2 ms. Moreover, CPU load can produce packet loss at 100 Mb/s — overflow of Network Interface Cards (NIC) buffers. Such packet loss produces retransmissions and the download time increases.

To show the differences between TCP and UDP behaviors, it have been monitored, with the help of `strace`[7], the system calls during the execution of a TCP file-download (by means of wget) and a UDP packet-train sending —by means of hping. Figure 3.12, 3.13 show the six heaviest system calls for both cases. It can be observed that the number of system calls (open/close the socket, read/write in the socket, etc.) in the TCP download is much greater than in the UDP packets-train case. Note that the system calls involved in TCP processing take significant times while this is not the case for UDP, whose execution times are so small that cannot be reported by strace.

```
  % time seconds  usecs/call   calls   errors  syscall
-----------------------------------------------------------------
1  83.17  0.143883        5     30852          select
2  13.90  0.024056        1     31384          write
3  1.76   0.003050        0     30879          read
4  1.17   0.002019        0     30851          clock_gettime
5  0.00   0.000000        0        69      30  open
6  0.00   0.000000        0        44          close
```

Figure 3.12: `strace` output of a TCP file-download using wget

```
  % time seconds  usecs/call   calls   errors  syscall
-----------------------------------------------------------------
1   nan   0.000000        0        15          read
2   nan   0.000000        0         4          write
3   nan   0.000000        0        14          open
4   nan   0.000000        0        20          close
5   nan   0.000000        0         1          execve
6   nan   0.000000        0       101          time
```

Figure 3.13: `strace` output of an UDP packet-train reception

In light of these results, it can be concluded that the file-download technique is very sensitive to the CPU load and, then, is not a good choice to estimate QoS parameters in general-purpose PCs running concurrent software.

[7] `http://linux.die.net/man/1/strace`

Once it has been demonstrated the negative impact of CPU load over the file-transfer methods, packet-train methods should be analyzed. In next experiments, the effects of memory and CPU load on the measurement obtained with the packet-train technique are analyzed. For the next experiments packet trains with $N = 100$ Ethernet-Maximum Transmission Unit (MTU)-sized packets are used. This technique uses UDP as transport protocol.

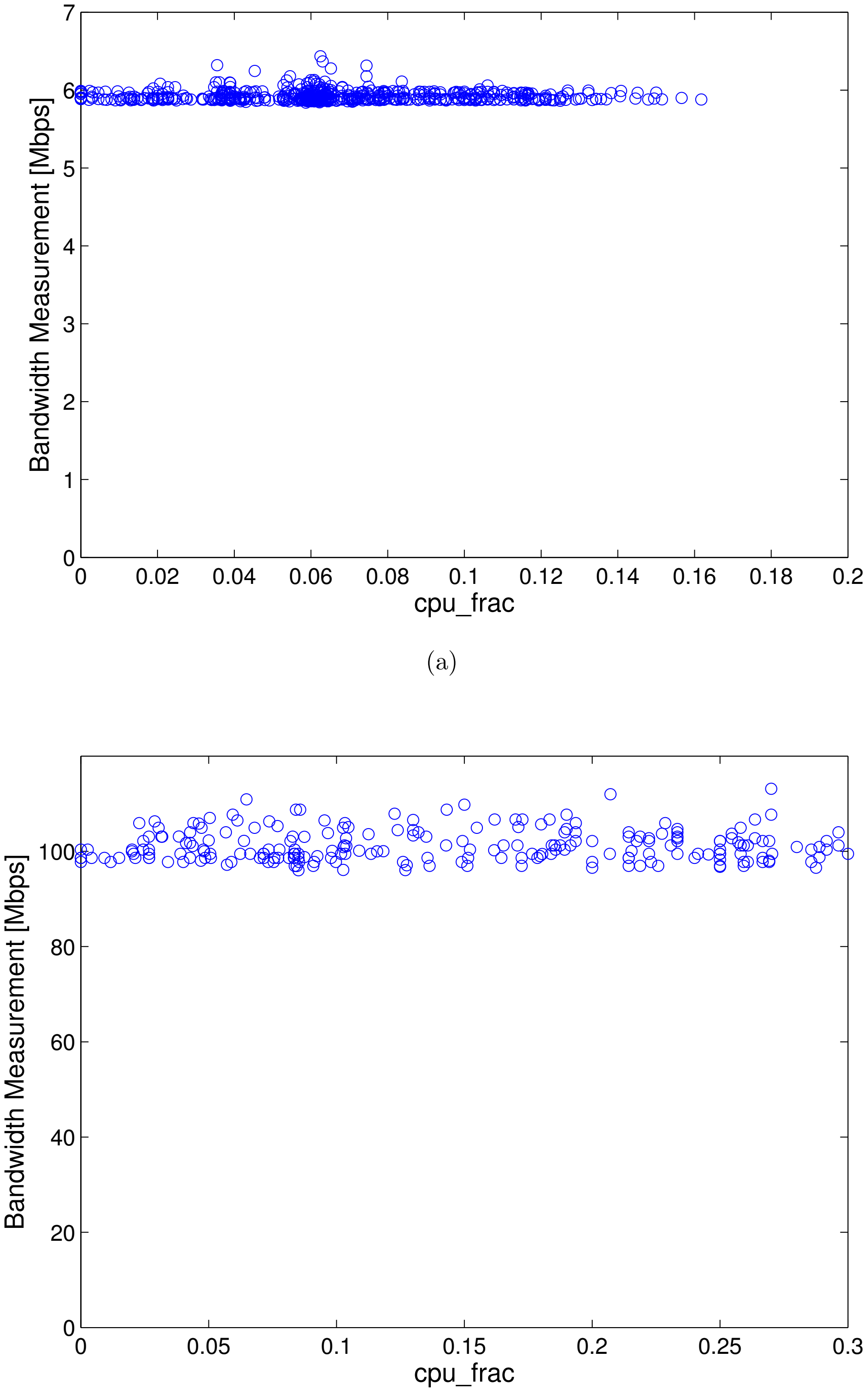
In high speeds links (100 Mb/s or greater), the packet-train technique is affected by IC [PJD04]. This effect is analyzed on Section 3.3.4. For the sake of simplicity and correctness this feature is disabled in the operating system when experiments are being executed.

First of all, the influence of CPU load on the bandwidth measurement is analyzed. Figure 3.14 shows the scatter plot between cpu_frac and the bandwidth measurement for the different test cases. It can be observed that the measurements do not depend on the CPU load, unlike the file-download technique.

Furthermore, Figure 3.15 shows the scatter plot between mem and the bandwidth measurement. It can be observed that the memory load does not have influence on the bandwidth measurement either.

To further analyze the influence of external factors in the packet-train method performance, the effects of other factors such as operating system are studied. To this end, several measurements are performed with the packet-train technique on the most popular operating systems, namely: Windows, Linux and Mac OS.

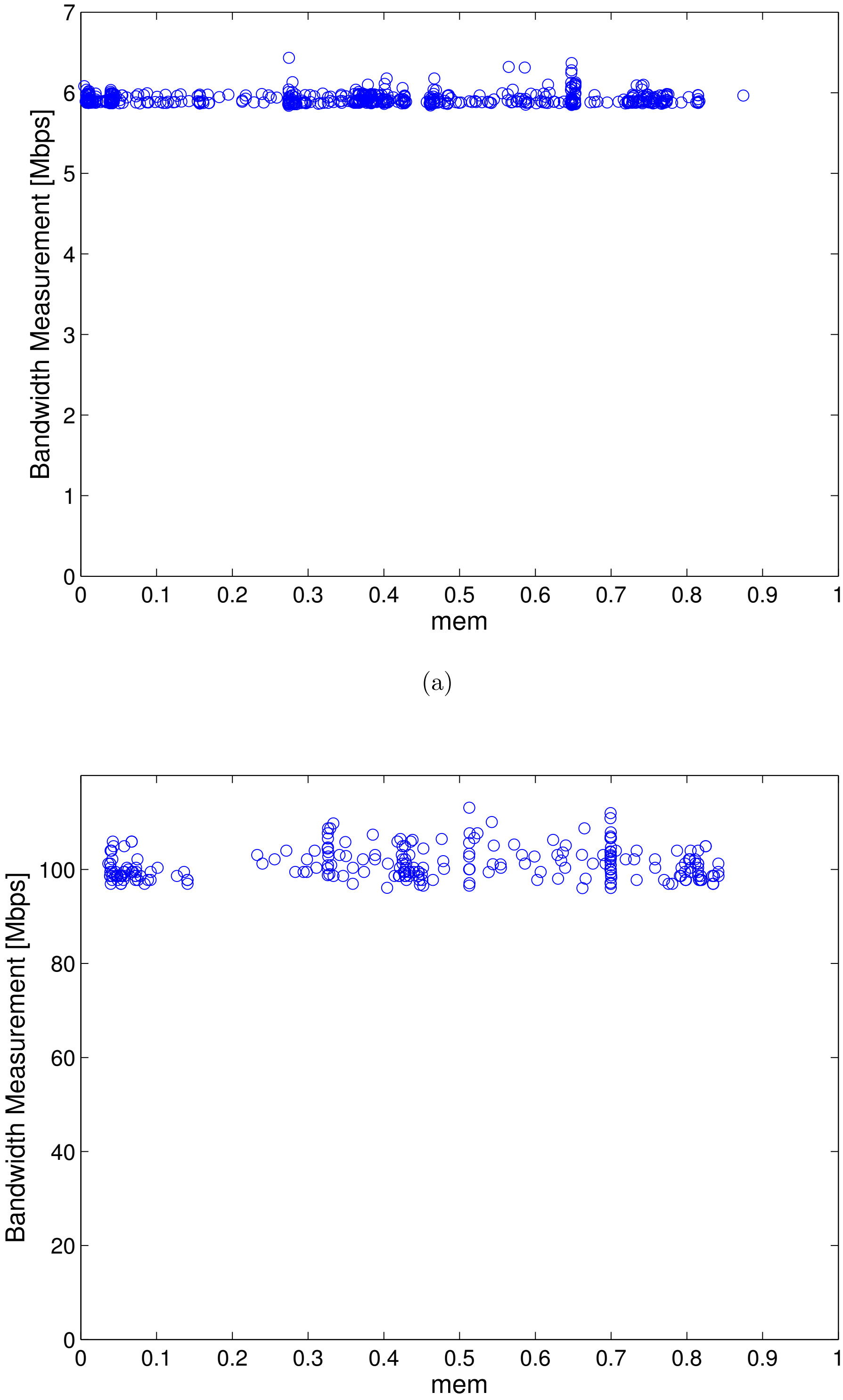
In order to statistically prove the independence between the measurement and the external factors (operating system, memory and CPU load), the ANOVA test is performed. A discretization must be made to the continuous explanatory variables: mem and cpu_frac. To do this, three levels are defined, namely: low (the lowest 33% of the measurements), high (the highest 33%) and medium —the remaining measurements. Tables 3.3 and 3.4 show the results of the ANOVA test, for both cases —6 Mb/s and 100 Mb/s. It can be concluded that none of the factors are statistically significant —at common significance level $\alpha = 0.05$.



(a)

(b)

Figure 3.14: *cpu_frac* vs bandwidth measurement with packet-train technique on Windows, Linux and Mac OS using UDP as transport protocol. (a) Link Capacity 6 Mb/s (b) Link Capacity 100 Mb/s



(a)

(b)

Figure 3.15: *mem* vs bandwidth measurement with packet-train technique on Windows, Linux and Mac OS using UDP as transport protocol. (a) Link Capacity 6 Mb/s (b) Link Capacity 100 Mb/s

Table 3.3: Analysis of variance table for the bandwidth measurement with packet-train technique on Windows, Linux and Mac OS using UDP as transport protocol. Link capacity 100 Mb/s

Factor	**Degrees of Freedom**	**Sum Sq.**	**p-value**
Operating System	2	30.00	0.1221
mem	2	14.52	0.3598
cpu_frac	2	18.7	0.2671
Residuals	245	1732.21	

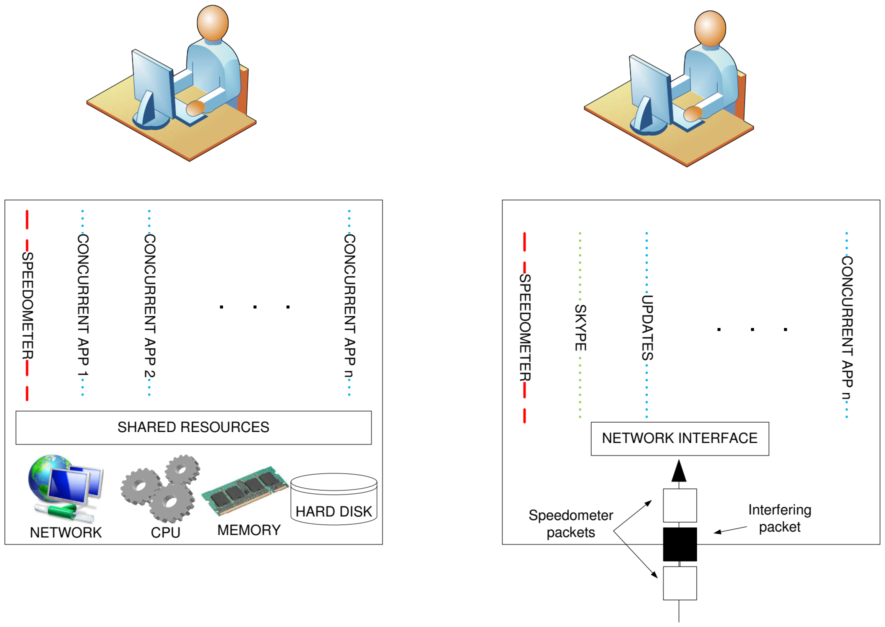
Table 3.4: Analysis of variance table for the bandwidth measurement with packet-train technique on Windows, Linux and Mac OS using UDP as transport protocol. Link capacity 6 Mb/s

Factor	**Degrees of Freedom**	**Sum Sq.**	**p-value**
Operating System	2	0.01189	0.3084
mem	2	0.01451	0.2381
cpu_frac	2	0.01189	0.3084
Residuals	623	3.14262	

Therefore, it can be empirically concluded that the packet-pair technique is highly immune to both CPU and memory load. In addition, it has been proved that the operating system does not have influence on the measurement.

3.3.3 Self-induced Traffic

In this section the impact of cross-traffic from other applications in the measurement accuracy is analyzed. When measurements are being performed in general-purpose systems, several concurrent applications may be executing and generating traffic. Figure 3.16 depicts such situation.



(a) (b)

Figure 3.16: Effects of concurrent applications. (a) The measurement is affected by concurrent applications, which share system resources such as CPU, memory, hard disk and network (b) Cross-traffic generated by concurrent applications (such as Skype, Updates, Spotify, Antivirus ...) interferes in the measurement

Clearly, cross-traffic from other applications has an impact in the measurement accuracy. Figure 3.17 shows a worst case in which the NIC driver interleaves one packet of interfering traffic between every two packets of the probe packet-train. In the depicted example, the client performs a measurement in the direction server to client, while concurrently downloading a video clip. In that case, the bottleneck link bandwidth estimation is exactly one half of the real link bandwidth value.

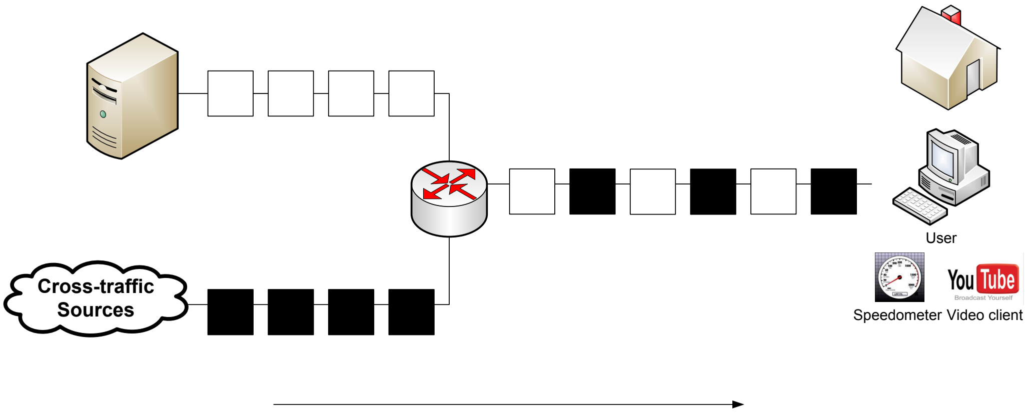


Figure 3.17: Incorrect measurement due to cross-traffic

Note that the larger the packet-train the better against interfering traffic. However, the packet-train is more intrusive in the bottleneck link and the measurement time is larger than the packet-pair technique. In this light, system-level limitations in the estimation of cross-traffic from other applications and CPU and memory load are explained.

An analysis is made on the operating system dynamics in the worst case depicted by Figure 3.18, which shows a packet-train sent from the measurement server to the client. Meanwhile, a video was downloaded to the client.

Figure 3.18 shows packet interleaving, that is, there are some interfering packet from a concurrent application (video download) between two probing packets.

```
time arrival  source>destination  protocol details
20.722397 measurementServer>client: UDP,length 1400
20.731881 videoServer>client: ack 78 win 14600
20.733910 measurementServer>client: UDP,length 1400
20.734258 videoServer>client: 73666:75126(1460) ack 78 win 14600
20.734344 videoServer>client: 75126:76128(1002) ack 78 win 14600
20.745433 measurementServer >client: UDP,length 1400
```

Figure 3.18: `tcpdump` output: Cross-traffic interferes on the measurement

The previous experiment motivates the use of techniques that estimate the interfering traffic generated from other applications, in order to drop measurements with a high interference level. In a standard scenario, the measurement process does not have higher priority in comparison to other concurrent applications. Nowadays, the measurement application may run on top of a virtual machine, which makes things worse compared to native processes [WN10]. Note that measurements of interfering traffic must be performed at the application layer. This is also a distinguishing feature of the standard measurement scenario: there is no access to privileged kernel routines that provide fine-grain traffic measurements at driver level. Again, this is because the measurement process is just another application.

In this case study, a Java program is in charge of measuring CPU, memory and cross-traffic load parameters. To this end, the SIGAR JNI API [8] is used. SIGAR API provides a platform-independent approach for measuring system resource parameters. JNI is a native C interface linkable from Java. Specifically, SIGAR is used to obtain:

- The percentage of CPU time used on the bandwidth measuring process.
- The amount of free memory on the system during the measurement process.

[8] `www.hyperic.com/products/sigar.html`

- The number of bytes and packets received on the NIC during the measurement process.

It is worth noting that the above performance items cannot be obtained with arbitrarily small resolution. This API uses system Java Native Interface (JNI) calls to the underlying operating system to obtain the requested parameters. If multiple queries are executed in a short space of time, the operating systems counters (received bytes, number of interrupts, etc) will not be updated and the Application Programming Interface (API) returns invalid values. In addition, multiple queries to the operating system produce a large processing overhead. Consequently, the measurement intervals must be relatively large, in order to obtain reliable results. As a consequence of this limitation, only the number of interfering packets for the entire duration of the train can be obtained, and not the packet timestamps.

The effect of cross-traffic on both file-download and packet-train techniques (with train length of $N = 100$ Ethernet-MTU-sized packets) must be analyzed and quantified. Besides one measurement process (based on the ETSI recommendation [Ins08]), the commercial measurement system Speedtest[9] is also used for file-download.

Three cross-traffic scenarios are considered: interfering TCP traffic, UDP cross-traffic with random packet inter-arrival times and UDP cross-traffic with Constant Bit Rate (CBR).

Firstly, the impact of concurrent TCP flows in a general-purpose PC is analyzed. In order to generate the interfering TCP flows, a large file is downloaded using `wget`. The file size is 100 MB and the download lasts around 2 minutes when the link capacity is equal to 6 Mb/s. There are five replicas of the experiment for each cross-traffic level —x-axis in Figure 3.19, 3.21, 3.22. Figure 3.19 shows the bandwidth measurements obtained with packet-train (using $N = 100$ Ethernet-MTU-sized packets), file-download and Speedtest for different amount of interfering TCP connections. It can be observed that TCP cross-traffic does not affect the packet-train method (which is based on UDP packets) as expected. However, the custom file-download method

[9] `www.speedtest.net`

and Speedtest are affected by the presence of interfering TCP connections because they also feature congestion control.

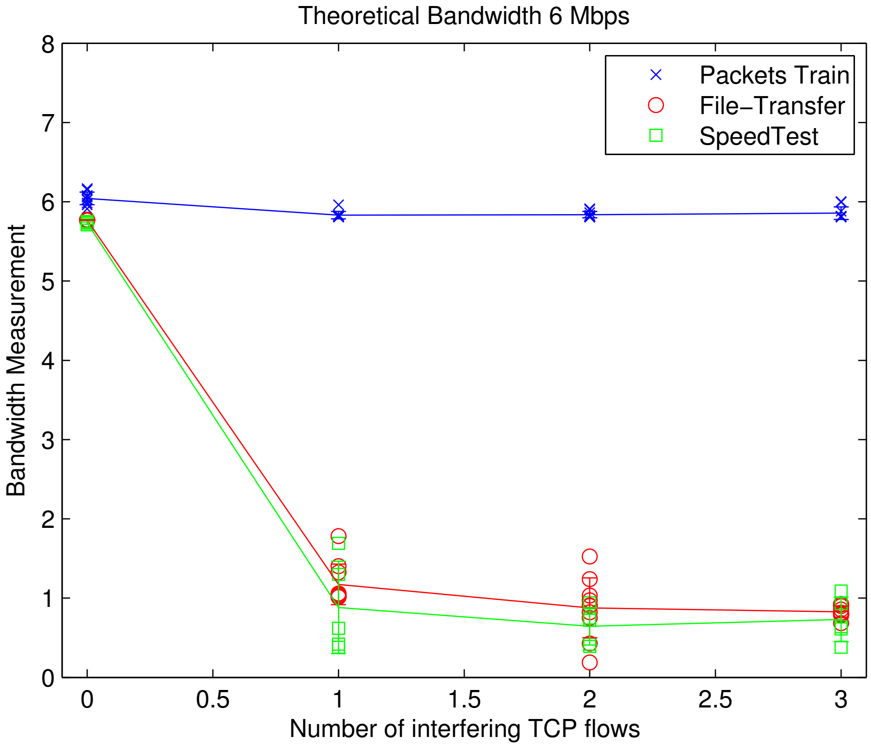


Figure 3.19: Bandwidth measurements with interfering TCP flows using file-download technique, Speedtest and packet-train technique

This result clearly shows that TCP-based speedometers are highly affected by concurrent TCP flows. It could be expected that the total bandwidth was equally shared by all flows, but that does not happen in this case —for instance, when there is one concurrent flow it is expected to obtain around 3 Mb/s but is obtained around 2 Mb/s instead. The main reason is that the adaptation process is not so fast (can take several seconds or even minutes) and the duration of the measurement is not enough to reach the stationary state in which the total capacity is equally shared by all flows.

This behavior can be observed in Fig. 3.20, where the throughput of a concurrent flow and of the file-download measurement are shown —the link capacity is 10 Mb/s. It can be observed that the first flow reaches the maximum throughput quickly. At 17 seconds, a measurement starts consisting of a 10 MB file-download. Such measurement flow does not reach half the total bandwidth until 13 seconds after the start time. Note that the measurement takes 24 seconds to download a 10 MB file. Therefore, the obtained measurement is $\frac{10\times 8}{24} = 3.3$ Mb/s, i.e. a third of the total capacity, as in the experiments in Fig. 3.19.

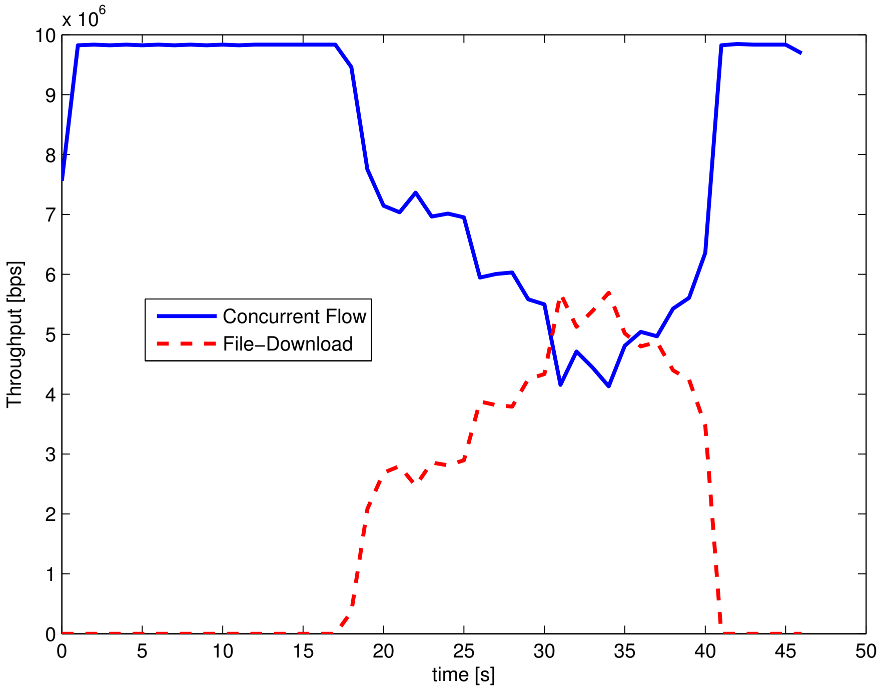


Figure 3.20: TCP behavior with concurrent downloads

Secondly, the impact of UDP cross-traffic is analyzed. For the next experiments it is considered that inter-packet gaps are heavy-tailed distributed according to a Pareto-Normal distribution —with mean value equals 200 ms and standard deviation equals 200 ms. This kind of traffic emulates traffic produced by a concurrent application in a general-purpose PC (such as video or audio streaming) [FAM01, ITF04, ZH07]. There are five replicas of the experiment for each cross-traffic level. Figure 3.21 shows the bandwidth measurements obtained with the three methods varying the UDP cross-traffic rate from 0 to 6 Mb/s (link capacity), in the case of random delayed packets. On the one hand, it can be observed that UDP cross-traffic does not affect the packet-train method for these cross-traffic rates. On the other hand, the file-download method and Speedtest are affected by the UDP cross-traffic since both techniques are TCP-based, which adapts its rate to the available bandwidth. It is worth noting that TCP connections adapt to available bandwidth better in this case (UDP interfering traffic) than in previous case —TCP concurrent connections.

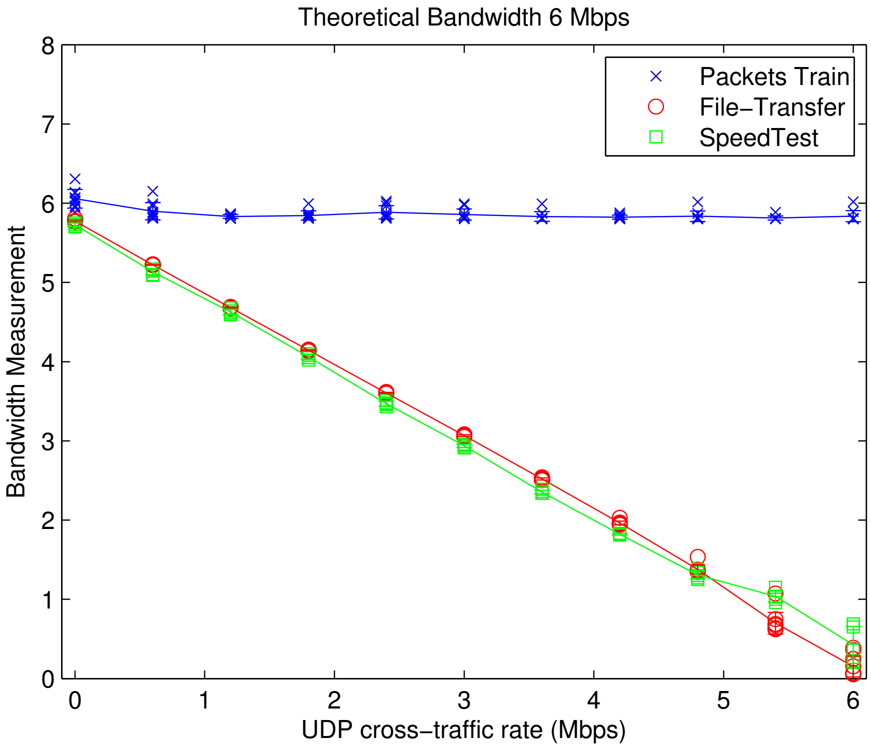


Figure 3.21: Bandwidth measurements with UDP interfering traffic (randomly delayed) using file-download technique, Speedtest and packet-train technique

In the third experiment, CBR UDP cross-traffic is injected. This is the worst-case for the packet-train method as visually observed in Figure 3.17.

Moreover, CBR is a common model for Voice over Internet Protocol (VoIP) applications or periodic heartbeats —but only at low transmission rates. As in the previous experiments, there are five replicas of the experiment for each cross-traffic level. Figure 3.22 shows the bandwidth measurement versus cross-traffic rate. The packet-train method is accurate if the cross-traffic rate is less than the link capacity. However, if the cross-traffic rate equals the link capacity, the bandwidth measurement is reduced to a half because the inter-arrival time is doubled due to the interfering packets. The behavior of the TCP-based methods (File-transfer and Speedtest) is similar to the one observed in the previous experiment.

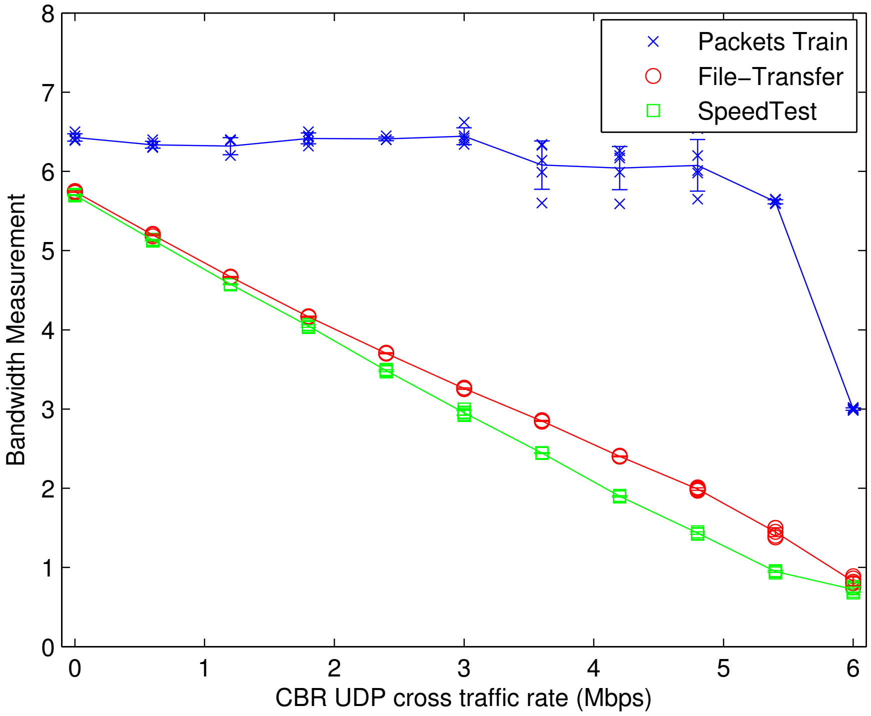


Figure 3.22: Bandwidth measurements with UDPinterfering traffic (CBR) using file-download technique, Speedtest and packet-train technique

3.3.4 Interrupt Coalescence

In high speeds links (100 Mb/s or greater), the packet-train is affected by IC [PJD04]. IC, is a NIC mechanism by which several packets are queued at NIC level before generating a processing interrupt. Such mechanism, moderates the presence of interrupts reducing, thus, the processor time devoted to interrupt handling produced by packets arrival. If this technique is not applied, capturing traffic at high-speed, could lead to a CPU saturation stealing general purpose processing time to the detriment of packet processing time.

As this mechanism queues packets for a time lapse and then transfers them in batch mode, the perceived packet inter-arrival times could be lower than the real inter-arrival times due to the grouping of several packets. These smaller inter-arrivals have a great impact over the QoS parameters estimation. For instance the measured bandwidth by packet-train methods is increased and the OWD is also increased. To avoid this, this feature must be disabled in the operating system when measuring. In the cases when such option is not available, a packet-train technique is proposed to measure in presence of interrupt coalescence. Such technique is proposed in Section 3.3.6.

3.3.5 Threshold-based rejection techniques for biased measurements

File-transfer techniques have shown bad performance in presence of cross-traffic and high CPU load, as it can be observed in the previous experiments and in Section 3.3.3. Therefore, only packet-train techniques are taken into account in what follows. The third experiment in the Section 3.3.3 shows that although packet-train methods are more robust to cross-traffic they may not produce accurate results. This fact motivates the development of an analytic model, (which is explained in this section) that takes the number of interfering packets as an input and marks the measurement as either valid or invalid.

In a packet-train measurement, a train of N packets is sent. The obtained measurement is deemed as correct if at least two probe packets arrive contiguously —i.e. with no interfering packets in between. Such packets serve to accurately estimate the bottleneck link bandwidth. Figure 3.23 shows this behavior.

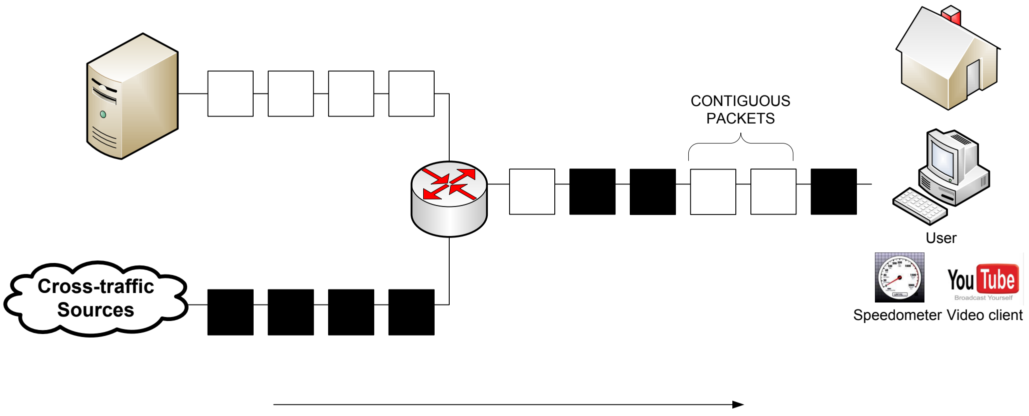


Figure 3.23: Correct measurement in spite of cross-traffic

In this light, an estimation of the number of interfering packets that fall in between probe packets becomes crucial. However, it can only be measured the total number of interfering packets throughout the packet-train. Thus, it is key to estimate what is the probability that a given number of interfering packets (say m) occupies all probe inter-packet gaps. To do so, a simplifying assumption must be made: an interfering packet makes a random (uniform) choice of all the probe inter-packet gaps. Clearly, any given interfering packet may not be transmitted in a probe inter-packet gap that happened before its actual transmission time. However, note that the above assumption represents a worst-case for the packet probe. Indeed, since packets may choose any given probe inter-packet gap the probability of all gaps being occupied increases with respect to the real-life case, as shown in Figure 3.24.

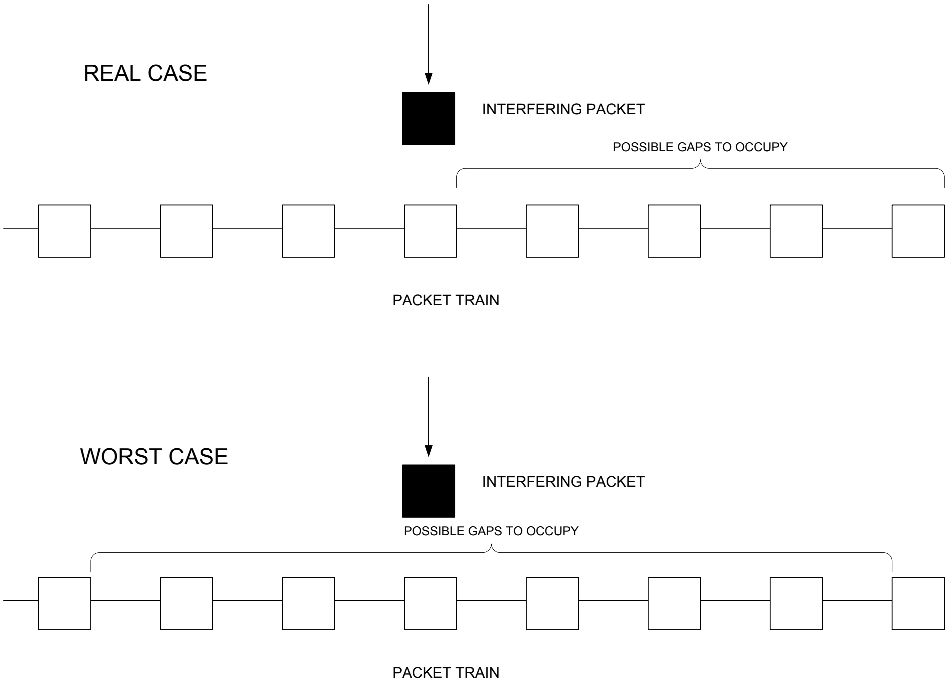


Figure 3.24: Comparison between the real case and the worst case — assumption of the model

Being this model worst-case, the main goal is to tag a given measurement as invalid if the probability of all gaps being occupied is larger than a threshold ϵ. Low values of ϵ provide a large rejection rate and vice versa, much like a significance level. The model formulation is actually more conservative and will reject possibly valid measurements. Namely, the proposed model has a larger rejection rate than strictly necessary, but will filter out invalid measurements at the threshold probability ϵ.

With the previous assumptions, the random scatter of the m packets (from cross-traffic) across $N-1$ gaps follows a multinomial distribution: let

N and m be the length of the train and the amount of interfering packet, respectively, where each packet independently drops into a gap with probability $\frac{1}{N-1}$ —i.e. each interfering packet has the same probability of drop in each gap. Let $n_1, \ldots, n_{N-1}$ be a given distribution of the interfering packets where n_i is the amount of packets which fall into the ith gap and $\sum_i n_i = m$. The probability of a given gap occupancy vector $(n_1, \ldots, n_m)$ is:

$$P(n_1, \ldots, n_m) = \prod_{k=1}^{m-1} \binom{m - s_{k-1}}{s_k - s_{k-1}} \left(\frac{1}{N-1}\right)^{s_k - s_{k-1}} \left(\frac{N-2}{N-1}\right)^{m-s_k} \quad (3.12)$$

$$\text{where } s_k = \sum_{i=1}^{k} n_i$$

The measurement rejection probability (i.e. the probability that no gap contains less than 1 packet) can be formally defined as:

$$P\left(\min_{i=1,\ldots,N-1} n_i \geq 1\right) \quad (3.13)$$

It turns out that the multinomial distributions calculations are involved, especially with large packet-trains. The reader is referred to the Appendix B for further insight.

Figure 3.25 shows the measurement rejection probability (Equation 3.13) for packet-train lengths, $N = \{10, 50, 100, 150\}$ and varying the amount of interfering packets, $m = \{0, \ldots, 1000\}$. Clearly, the rejection probability increases with the number of interfering packets m. Moreover, it can be observed that the rejection probability decreases quickly as the length of train (N) increases. For instance, if $N = 10$ and $m = 3 \cdot N = 30$ the rejection probability is 75%, whereas if $N = 50$ and $m = 3 \cdot N = 150$ the rejection probability is 8.5%. On the other hand, if $N = 100$ and $m = 3 \cdot N = 300$ the probability is 0.005 and, even more, if $N = 150$ and $m = 3 \cdot N = 450$ the probability is 0.0003. Using $\epsilon = 0.05$, the maximum number of interfering packets to accept a measurement is equal to 12, 141, 351, 587 for packet-train lengths (N) of 10, 50, 100 and 150 packets. Accordingly, when a measure is performed, the packet-train length, N, is known beforehand and the number of interfering packets, m, can be obtained through the SIGAR API. Thanks to the model, the probability that the measurement is correct can be calculated. Namely, with a packet-train of length N, what is the maximum number of interfering packets to consider that a measurement is correct, with probability greater than $1-\epsilon$, which is shown in Figure 3.26.

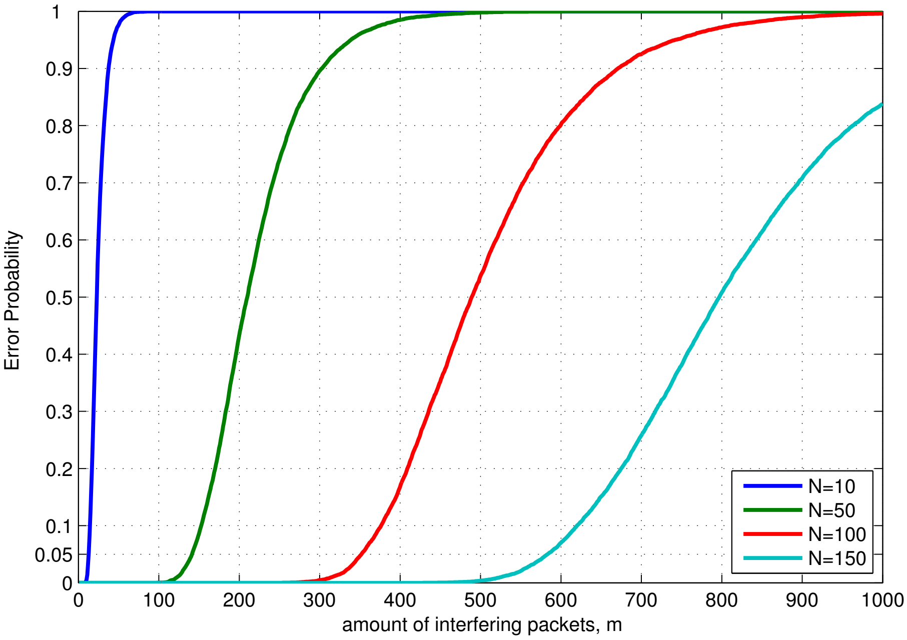


Figure 3.25: Error probability as a function of the number of interfering packets, m, for several packet-train lengths, N

Model Validation

In this section, the proposed model is checked by means of both simulations and experiments. That is, ensure that a measurement labeled as correct by the model is actually correct, with probability greater than $1-\epsilon$.

Firstly, samples of inter-arrival packets times are generated and, according to them, the number of packets that have filled every inter-packet gap of the measurement train are counted. Should all gaps be filled by at least one interfering packet then the measurement is marked as incorrect.

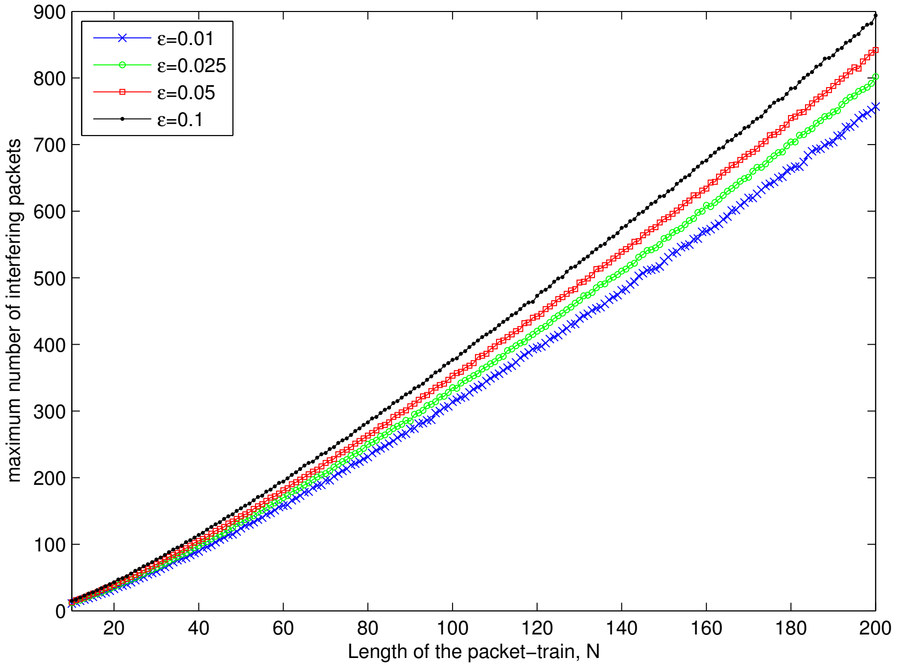


Figure 3.26: Maximum number of interfering packets to accept a measurement as a function of the packet-train length, N, for different values of the error probability threshold, ϵ

Two different inter-arrival time distributions have been simulated: on the one hand, the classic model of exponential inter-arrival times —according to a Poisson arrivals process; on the other hand, a long-term dependency using a heavy-tailed distribution like Pareto-Normal. The latter distribution (defined in `netem` tool) is a weighted sum of a Pareto sample with $\alpha = 3$ (finite variance) with weight 75% and a Normal sample with weight 25%. Figure 3.27 shows the estimated error probability for each distribution (exponential and Pareto) and the theoretical probability provided by the proposed model —$\epsilon = 0.05$. It can be observed that the estimated error probability for each distribution is below the rejection probability provided by the model, as expected.

Finally, the performance of the proposed model is assessed with exper-

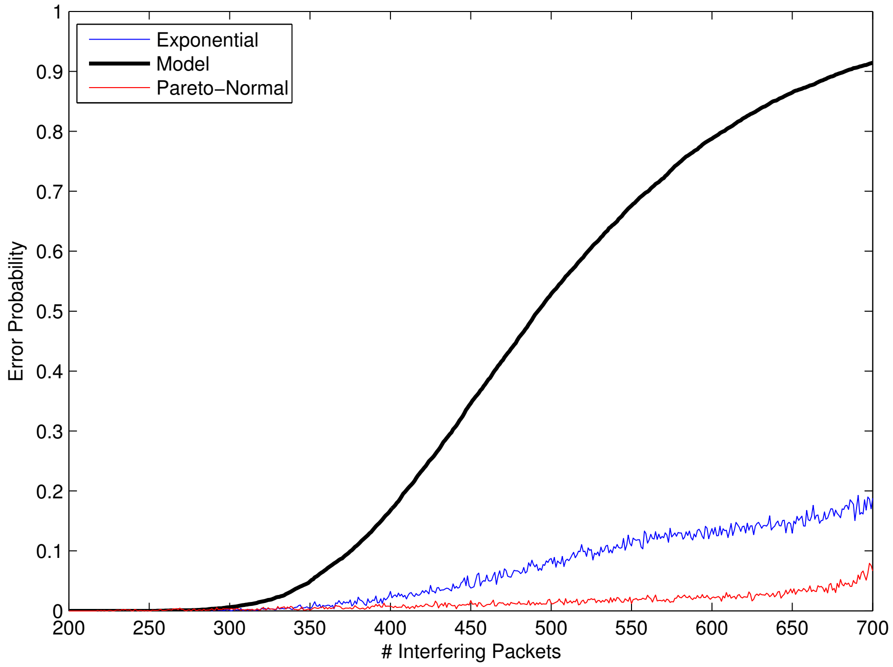


Figure 3.27: Comparison between the theoretical error probabilities provided by the model and the estimated error probabilities by means of simulations for Exponential and Pareto distributions. The packet-train length is equal to $N = 100$ packets

iments. To this end, more than three thousands measurements were performed and the number of incorrect measurements was counted, comparing this account with the theoretical value provided by the model. A measurement is marked as incorrect when the measured bandwidth is reduced by half —or less. Actually, the obtained bandwidth is divided by $2, 3, 4 \ldots$ (when the minimum amount of interfering packets in a inter-packet gap is $1, 2, 3$) due to fact that the size of interfering packets is the same as probing packets. This is possible because the experimental scenario is controlled —i.e. no cross-traffic is present, besides the cross-traffic intentionally injected. In this experiment, 100 Ethernet-MTU-sized packets long trains are used. Moreover, UDP cross-traffic (with random inter-arrival times distributed according to a Pareto-Normal distribution) is injected with a mean rate of three times the measurement rate —obtaining around 300 interfering packets per train. The experiment is performed for three different bandwidth values: 900 Kb/s, 6 Mb/s and 12 Mb/s.

Table 3.5 shows the results obtained. $\bar{m}$ and $\bar{N}$ represent the mean number of interfering packets and probe packets, respectively. It is also worth remarking that the length of the train is not $N = 100$ for every measurement due to the packet loss. The estimated error counts the proportion of wrong measurements i.e. $\frac{\#incorrectmeasurements}{\#totalmeasurements}$. The theoretical error is bounded by the error probability for $N = 100,\ m = 270$ (the maximum length of the train and the minimum number of interfering packets, in the experiments) and by the error probability for $N = 93,\ m = 300$ —the minimum length of the train and the maximum number of interfering packets, in the experiments. Note that the error probability given by the model is greater than the error ratio obtained in the experiments. Therefore, the model gives a valid upper bound for the error probability.

Thus, it can be concluded that the proposed model provides a sound methodology to filter out incorrect measurements polluted by cross-traffic.

Table 3.5: Empirical evaluation of the theoretical model. $270 \leq m \leq 300, 93 \leq N \leq 100$

Bandwidth	$\bar{m}$	$\bar{N}$	**# Measur.**	**Est. Error**	**Theo. Error**
900 Kb/s	282	96	3601	0.0061	$0.0006 \leq p \leq 0.0187$
6 Mb/s	281	96	5021	0.0030	$0.0006 \leq p \leq 0.0187$
12 Mb/s	289	96	4332	0.0115	$0.0006 \leq p \leq 0.0187$

3.3.6 QoS Mechanisms Impact

Estimation technique of Token Bucket Parameters

One key aspect when measuring heterogeneous networks is determining the presence of Token-Bucket like mechanisms to adjust the measurement method and estimate the shaping/policing parameters. Estimating these parameters allows a more precise and accurate QoS parameters measurement. Measuring using the file-transfer technique provides an accurate estimation in terms of measured bandwidth and RTT but as shown in Section 3.3.2 suffers a degradation in presence of high CPU load and cross-traffic. From now on, only packet-train based method will be studied.

In packet-train-based measurements, a train of N packets is sent. Given the existence of a shaping/policing mechanism, the first r packets (while the bucket is not empty) are sent to the maximum rate —Peak Information Rate (PIR). The rest of packets are sent at Committed Information Rate (CIR). If $r < N$, two (sub)packet-trains may be differentiated: the one sent to PIR rate and the one sent to CIR rate. Therefore, CIR and PIR can be estimated as follows:

$$CIR \approx \min_{i=r,\dots,N-1} \left(\frac{B}{t_{i+1} - t_i} \right) \tag{3.14}$$

$$PIR \approx \min_{i=1,\dots,r-1} \left(\frac{B}{t_{i+1} - t_i} \right) \tag{3.15}$$

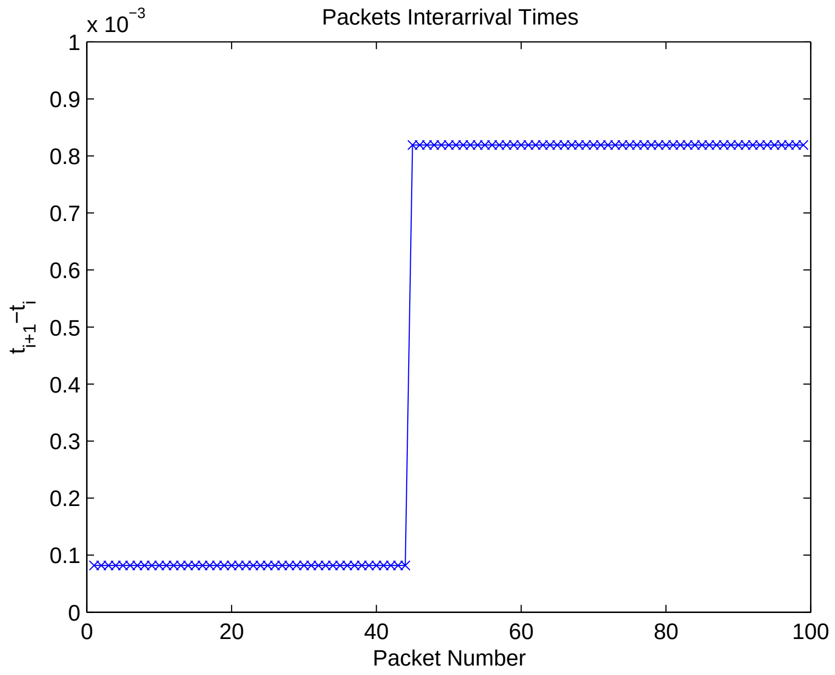
Moreover, burst length can be estimated as follows:

$$B_l \approx r \times B \left(1 - \frac{CIR}{PIR} \right) \tag{3.16}$$

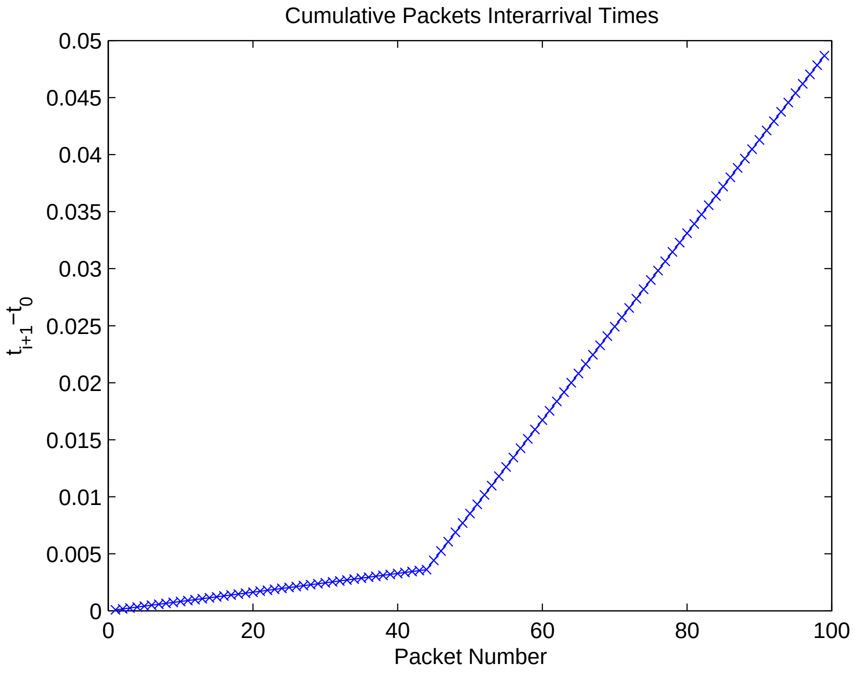
For instance, Figure 3.28(a) shows the inter-arrival times of a train of 100 packets of size 1 KB, with a token-bucket configuration of $CIR = 10$ Mb/s, $PIR = 100$ Mb/s and $B_l = 40$ KB. It can be observed the two stages: packets from 1 to 44 are sent at PIR and from packet 45 to packet 100 are sent at CIR. Therefore, PIR can be estimated using the first sub-train (until the change-point) and the CIR rate using the rest of packets.

But, how can a change-point be detected? Figure 3.28(b) shows the cumulative inter-arrival time of the same packet-train in Figure. 3.28(a). Problems produced by interrupt coalescence [PJD04] present a similar pattern and can be treated in the same way.

In order to calculate the slope, an detect the change, the least square error method is used —see Equation 3.17.



(a)



(b)

Figure 3.28: Packet inter-arrival times distribution of a packet-train with token-bucket ($N = 100$ packets, $B = 1$ KB, $r = 44$, $CIR = 10$ Mb/s, $PIR = 100$ Mb/s and $B_l = 40$ KB) (a) Packet inter-arrival times (b) Cumulative packet inter-arrival times

$$\hat{\beta_1} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{3.17}$$

The algorithm to detect the change-point is shown in Algorithm 1. Note that *slope* function returns the gradient of a line using the method previously described.

Algorithm 1 Change-point detection

```
$cum\_slope = 0$
$win\_slope = 0$
$i = win\_size + 1$
while $cum\_slope \leq win\_slope \times k$ AND $i < N - win\_size$ do
  $win\_slope = slope(N, cum\_iarrivals, i - win\_size, i)$
  $cum\_slope = slope(N, cum\_iarrivals, 1, i)$
  $i = i + 1$
end while
if $i + win\_size == N$ then
  print no change detected
  return $win\_size + 1$
else
  print change detected
  return $i + win\_size$
end if
```

If $r = N$, the link is not saturated by the train —i.e. $B_l > N \cdot B$. Then, every packet is sent at PIR. In this case, the packet-train length has to be increased in order to estimate token bucket parameters. In case that the measured link is not under the influence of token-bucket based mechanisms, the packet-train length may be increased until a hard-limit is reached following a similar algorithm to TCP window adjustment.

Experimental Results and Performance Evaluation

Figure 3.29 shows the estimations of the three parameters (CIR, PIR and burst length) by the proposed algorithm, for the following configuration in the token-bucket : $PIR = 100 Mb/s$ and varying the CIR from 5 Mb/s to 20 Mb/s and the burst length from 1 KB to 200 KB.

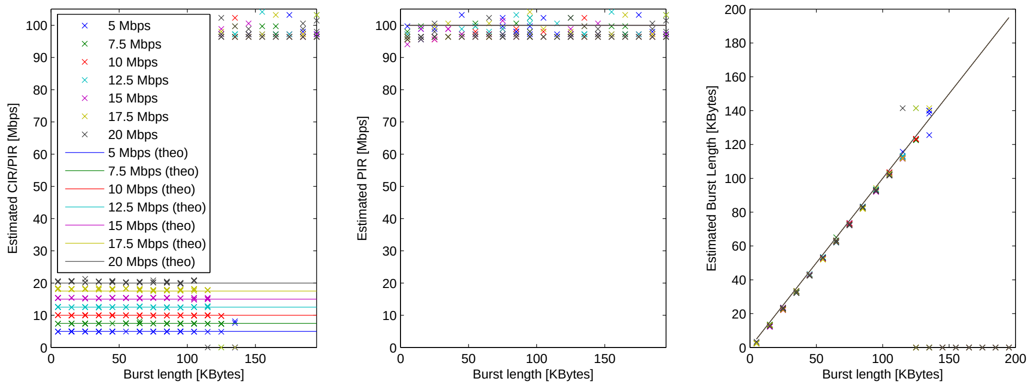


Figure 3.29: Estimated Token-Bucket Parameters: CIR, PIR and Burst length —$N = 100$ packets, $B = 1500$ Bytes, $PIR = 100$ Mb/s and several cases of CIR and B_l

Figure 3.30 shows the bandwidth estimated with the proposed technique (*CIRChecker*) and with other generic tools, such as Speedtest, Iperf (UDP and TCP), Capprobe, a File-transfer tool based on the ETSI guide [Ins08] and a generic Packet-train technique —without change-point detection.

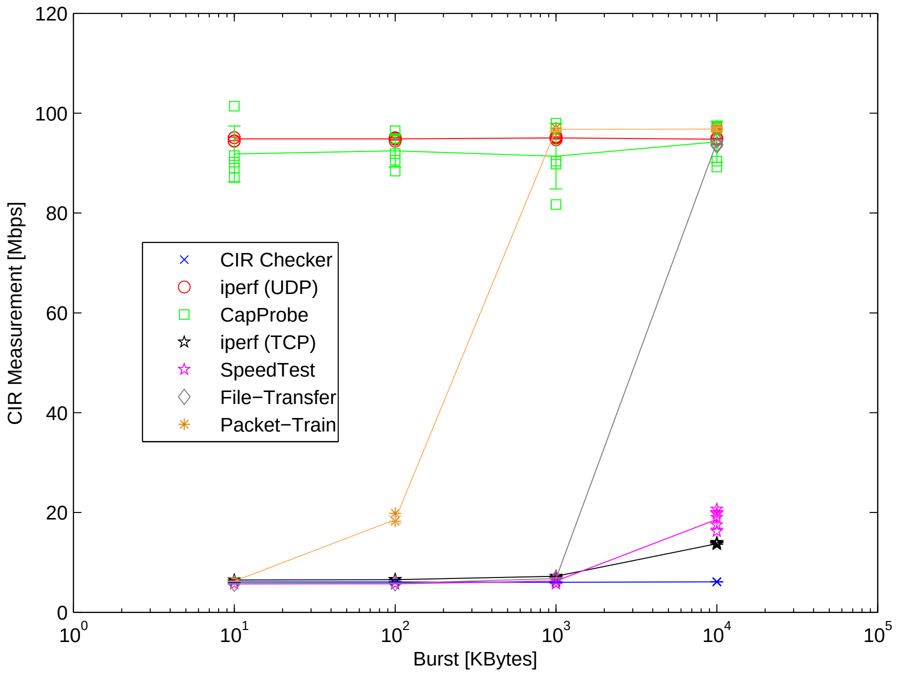


Figure 3.30: CIR Measurement without Cross Traffic —$PIR = 100$ Mb/s, $CIR = 6$ Mb/s and several cases of B_l

3.4 Conclusions

In this chapter a review of the active measurement methods has been carried out. Such review has focused mainly on the file-transfer and packet-pair techniques. As an important result of such review, the idea that some external parameters such as CPU load or interfering traffic have an impact over the quality of measurements has been presented. Based on such idea, an exhaustive study of the influence of CPU and memory load, self-induced traffic over active algorithms has been performed analyzing both low and high-speed scenarios and several operating systems. Such study has revealed that the memory load does not have influence on the measurement whereas the CPU load is very influential on the bandwidth measurement. Specially, file-transfer techniques are more affected by higher CPU loads than packet-pair methods. Attending to the self-induced traffic, the experiments have revealed that both file-transfer and packet-pair techniques are influenced by self-induced interfering traffic being the former method the most influenced one. In the case of packet-pair methods, the study has shown that such measurement method is only effected by CBR UDP cross traffic. To detect and reject polluted measurements, a threshold-based method has been proposed in this section. Such method uses a worst-case scenario approach to provide an upper bound for the measurement error based on the number of self-induced interfering packets and a multinomial model.

Finally, the last contribution of this section is the analysis of the results of the application of traffic rate control mechanisms such as shapers and policers when conducting active measurements. The experiments have shown that packet-pair based methods are sensitive to rate control mechanisms producing erroneous measurements which may lead to QoS parameters overestimation. To tackle this issue, an algorithm to detect and characterize the parameters of token-bucket based rate control mechanism has been presented and compared to other existing solutions achieving the best estimation results.

Chapter 4

Passive Measurements

This chapter provides the background and revises the most relevant works related to passive network measurements. In addition, new passive measurement techniques are proposed and analyzed in order to provide a better integration into Proactive Monitoring (PM) systems. The structure of the chapter is as follows. First, a brief description about different passive measurement methodologies and general concepts is done in Section 4.1. Then in Section 4.2 the most relevant passive measurements techniques are analyzed and discussed pointing out their pros and cons. In Section 4.3 a brief analysis of some hash functions is done attending to performance and empirical collision probability. Such analysis has provided some key ideas used for the proposal of a mechanism for passive packet correlation called Fast Packet Correlation (FPC). Next in Section 4.4, a flow creation and statistics gathering system is proposed and analyzed in terms of functionality and performance. To finalize, some important ideas are presented in Section 4.5 as conclusion.

4.1 Introduction

Passive network measurement is based on the idea of collecting traffic data and processing it to estimate network parameters and analyze the measured network performance and behavior. The collected data can belong to any of these categories:

- Directly captured traffic: for instance Packet Capture (PCAP) traces or packets received by any kind of special hardware.
- Pre-processed data and statistics gathered from devices: such information may be obtained from routers, switches, Network Intrusion Detection Systems (NIDS), etc. For example Multi Router Traffic Grapher (MRTG) data or exported router NetFlow/Internet Protocol Flow Information eXport (IPFIX) flows are representatives of this category.

Depending on the gathered data, the passive monitoring may provide different output information. For instance, analyzing the information provided by a NIDS may produce output data such as the number of detected threats, a list of active malicious Internet Protocol (IP) addresses or the number of suspicious HyperText Transfer Protocol (HTTP) connections per second. On the contrary analyzing low level data such as exported flows, the output estimations are slightly different. In this scenario, for instance, some Quality of Service (QoS) parameter estimations such as packet loss or aggregated available bandwidth can be done as well as other estimations as number of active flows or a list of most active IP addresses. Analyzing all the traffic traversing the monitored links packet-by-packet provide all the previous information and also allows applying different techniques such as Deep Packet Inspection (DPI) or statistical analysis. Usually, such approach is very expensive in terms of both data storage and computational power. Moreover traffic capturing at high-speed rates (10 Gb/s and above) have been proved to be a challenging task, especially when using commodity hardware [GDMR$^+$13, FD10] that is more likely to be found on heterogeneous networks.

The main advantage of passive measurement is the non-intrusiveness on the analyzed network. Usually, using some Switched Port Analyzer (SPAN) port the traffic on the routers and switched may be captured without interfering on production traffic. However, in some cases, some extra traffic may be inserted into the network due to the transportation of data such as exported flows or MRTG data from sources to collector. Nevertheless, such data transfer is considered less intrusive than active measurement traffic injection. Passively estimating QoS parameters in an accurate way, is a complicated task due to the variation of network conditions and the data acquisition interval. In this light, for instance, if a large data gathering interval is selected, abrupt network changes in small time periods may not be detected. On the contrary, if a small data gathering interval is selected a lot of samples will be received and the processing power and time devoted to process such samples increases. In addition to this, passive monitoring may be a challenging task when processing big amount of distributed data due to correlation process. In what follows, a brief state of the art on passive monitoring is presented emphasizing in such aspects and focusing on QoS parameter estimation as in the active case.

4.2 Passive Monitoring Techniques

In this section a review of most popular passive monitoring techniques and systems is done focusing on three different approaches: flow level monitoring, packet level monitoring and MRTG level monitoring.

4.2.1 Flow Monitoring

Flow-level monitoring uses exported NetFlow/IPFIX flow information from routers or switches to either estimate QoS parameters or make assumptions about the state and performance of the monitored network. In this category some approaches to estimate QoS parameters have been proposed. The vast majority of such approaches are focused on estimating the QoS parameters using minimal information provided by either NetFlow based flows or simple IPFIX flows.

Passive bandwidth estimation using flow information has not received so much attention by the research community. In [OSSSP12] an approach based on the creation of flow-level time series is proposed. The proposed flow-level bandwidth estimation method is compared against a packet level analysis achieving results with low bandwidth estimation error (in the bounds of dozens of Mb/s) and bandwidth variance error varying from 0.06 to 66% depending on the scenarios and flow active and inactive timeouts. Such results are good but far from the ones obtained by active measurements.

In the case of latency estimation using flow-level measurements, in [LDK10] a multiflow latency estimator is proposed. Such estimator is based on the idea that packets traversing a link present a positive queuing delay correlation. Using this premise, packets are correlated using a hash-based flow sampling process in several points of the network and an estimator based on average delay of pairs is proposed. Such estimator provides accurate estimates in the bounds of 20% median error for flows greater than 100 packets.

In [LLdVBF04] a flow-based estimation of Bulk Transfer Capacity (BTC) is performed following the metric proposed by [MA01]. Such approach is compared against a packet level analysis and Iperf tool obtaining estimation errors lower than 5%. Also a methodology for estimating Round-Trip Time (RTT) consistent with the one recommended in [AKZ99c] is proposed. Such proposal achieves an accuracy in the order of milliseconds.

In [ZBBC09] several examples of QoS monitoring using IPFIX exported flows are presented. The most relevant comprises the estimation of RTT using a single point methodology and the estimation of One-Way Delay (OWD) using a multi-point approach. In the case of RTT a packet pair matching must be done at observation point to correlate both directions of a data connection —e.g. Domain Name System (DNS), Simple Network Management Protocol (SNMP) or Transmission Control Protocol (TCP) stream. In the case of OWD estimation using IPFIX a multi-point approximation only the packet timestamps and node ids are necessary.

In [RSDC11] a methodology for estimating packet losses on a link based on IPFIX information is proposed. The methodology correlates flow records produced at two different points of the network and creates a superflow containing both records with a temporal restriction. Analyzing such superflows, the packet loss rate can be estimated using the difference between the packets observed in the first observation point and the packets observed in the second observation point. This methodology is evaluated against a packet-level trace obtaining a good precision (mean error less than 8%) especially when combing the estimation with additional routing information to discard disappearing flows. Other loss detection algorithms have evolved to cope with high-speed environments as showed in [FUK$^+$09]. In this case the proposed algorithm is able to detect packet losses in high-speed scenarios using a flow correlation approximation. Such correlation uses the flow 5-tuple and a time window to analyze the number of packets in a flow in two different network points as shown in the previous work. The results obtained by this approach suggest that the precision obtained by the method is near to 100% in the bounds of 10 Gb/s scenarios.

4.2.2 Packet Monitoring

Packet monitoring is based on the use of either on-the-fly captured traffic or packet traces (usually PCAP[1] traces) to estimate QoS parameters.

In [ENUK06], a method to estimate the capacity of a link using tcpdump traces is presented. The proposed method analyzes TCP connections to estimate the capacity of the narrow link of a path using the methodology of packet dispersion. Such estimation takes advantage of TCP delayed ACK strategy as packets are sent in pairs. The methodology is validated both in real and simulated environments obtaining less than 5% overestimated values and less than 18% underestimated values. All the values were compared with an active estimation tool as ground truth.

In the case of passive RTT estimation, in [JD02] a method to estimate the TCP RTT using packet level traces is proposed. The methodology is based on the calculation of the elapsed time between a SYN and the first ACK of a TCP connection with some optimization in the case of HTTP connections and using Maximum Segment Size (MSS) estimation. RTT is only estimated over connections with no losses and no disordered packets to improve accuracy. The method is widely tested using real traces with an indirect verification approach. Such approach is based on the comparison of the calculated RTT values separating the TCP connections by direction —source and destination. The results show that 70%-80% of processed connections

[1] `http://www.tcpdump.org/`

have an absolute difference less than 25 ms.

In [BV02] a method for estimating the packet loss in a passive way is suggested. The methodology uses TCP sequence numbers to detect gaps and retransmissions. The algorithm is refined using the concept of significant packets. Such packets contain a data segment for the first time. Using this approximation, retransmissions losses are not taken into account. Additionally a correction for spurious timeouts is added to avoid marking packets as lost when the RTT increases. Other improvements are proposed to deal with load balancing and packet reordering. The presented method is analyzed both in simulation and real scenarios achieving errors lower than 10% for the 90% of the cases.

Trajectory Sampling and Flow Tracking

Trajectory sampling or packet tracking [DG00, SHSZ10] is a methodology for monitoring packets in large-scale networks using invariant thereof fields as they traverse across different hops. Usually the invariant fields are used to identify uniquely a packet across the network. This technique may be used together with packet records generation to obtain labeled packet information at different network points. Such labels are usually generated by means of a hash function applied over the invariant fields of the packets in such a way that the label remains the same independently of where the label was created.

For TCP/User Datagram Protocol (UDP) over IP packets the selected invariant fields for the hash calculation typically are:

- IP version.
- IP protocol.
- Source and destination IP addresses.
- IP identification field.
- Source and destination TCP/UDP ports.

- First 20 bytes of payload.

Some of the previously commented fields may be variable depending on the network configuration. For example, in domestic networks is more likely to find Network Address Translation (NAT) or Port Address Translation (PAT) technologies which convert IP addresses and TCP/UDP ports in variable fields. Normally, trajectory sampling is used in big data networks such as core ones where such problems do not occur. Other fields may be considered as invariant depending on the network conditions. For instance, the total length, flags and offset IP fields may be used as invariant fields if fragmentation is not so usual or if it is confined to the network edges.

Figure 4.1 shows an example of a packet traversing a set of hops and its corresponding packet records. The packet records must be collected and analyzed on one or several additional machines. Such analysis is very complex in terms of processing power and memory consumption as packets must be correlated attending to several characteristics.

The major drawback of this method lies on the amount of extra traffic generated as per each packet, a record is generated at every router or switch. Typically, a packet record may contain a label (32 bits), the packet size value (16 bits) and a timestamp value (64 bits). Using trajectory sampling, for every packet in a network at least 16 extra bytes are generated. Moreover the number of packet records grows linearly with the number of monitored hops in the network. For example on a 1 Gb/s network, near 1,4 million packets can be generated each second giving an extra traffic of 19,6 MB each second. Such extra traffic increases dramatically if, for example, 50 nodes are being simultaneously monitored —980 MB extra traffic is generated each second. For such reason, packet sampling mechanisms must be applied to reduce the number of analyzed packets. A flow tracking methodology is proposed in Section 4.4 to add a higher abstraction level and reduce the amount of extra traffic generated while reusing flow creation capabilities of modern routers and switches. Such methodology is similar to the one presented in [SRW$^+$08].

Trajectory sampling is useful for a wide variety of network monitoring tasks such as:

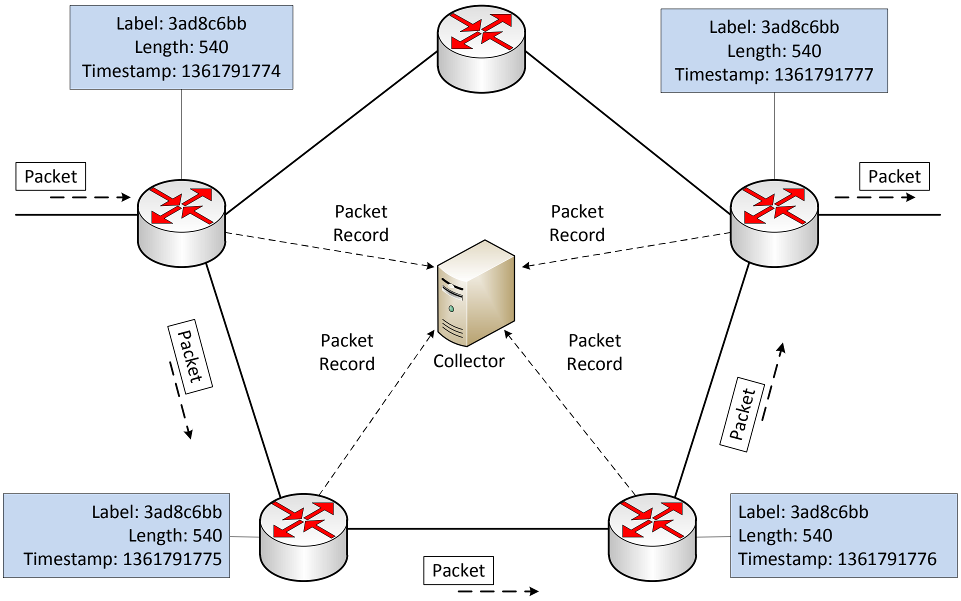


Figure 4.1: Trajectory Sampling Example

- Distributed Denial of Service (DDoS) attacks detection.
- Multi-hop delay and delay variation measurements.
- Multi-hop packet loss measurements.
- Performance analysis of new protocols.
- Active probe packets monitoring.

In [ZMSP03] a methodology for measuring OWD in multi-hop networks is presented. To calculate the delay of a packet across different hops, packets are labeled using a hash function and the OWD is calculated as the subtraction of the destination and source timestamps.

4.3 Packet Correlation and Sampling

One of the key aspects when applying trajectory sampling techniques is the selection of the packet sampling function and the method for correlating sampled information at the collector point. Several studies have been done [MND05, ZMD$^+$09, HSZ08] focusing specially on hash function collision analysis. Additionally, some of such studies have focused on algorithm time analysis under certain constraints while the application of such techniques on high-speed scenarios has not received much attention by the community. On heterogeneous networks where both high-speed and low-speed links may be monitored it is important to obtain the operating limits of these algorithms to adapt the capture system to the network conditions. Moreover, generating a hash value for each packet is not enough to fulfill the requirements of trajectory sampling techniques but also all generated hashes must be correlated at the collector point to infer information. Such correlation and path reconstruction is not a trivial task especially when working with online algorithms in high-speed environments.

4.3.1 Testbed

All the following tests were performed on an standard computer with an Intel i7-860 processor at 2.8 GHz, 8 GB of Random Access Memory (RAM) running a Fedora 18 Linux with a 3.9.6 kernel. For validation purposes, several real packet-level traces were used. Such traces were captured at an OC192 (9953 Mb/s) backbone link of a Tier-1 ISP located between San Jose and Los Angeles and were provided by The Cooperative Association for Internet Data Analysis (CAIDA) [WCA]. Additionally, a packet-level trace comprising 1 day of traffic of a commercial link of a Spanish operator was used.

Table 4.1 summarizes the most relevant characteristics from the used traces.

Trace	Number of packets	Trace duration	Avg. Packet Size (bytes)	Avg. Rate (Mb/s)	Avg. Concurrent Flows
CAIDA	25,476,037	1 min	655	2,321	519,715
Operator	97,092,043	1 day	258	3	4,264

Table 4.1: Trace Sets Information

4.3.2 Hashing

The first step when using trajectory sampling or flow sampling techniques is calculating a hash value for each incoming packet. Typically, the hash is calculated over some predefined packet fields and the data length used for hash calculation normally is in the range of 12-40 bytes. Generating hash values is not a simple task and may introduce an important processing overhead. Additionally, the hash algorithms used in packet and flow sampling must have a very low collision probability. Such constraints reduce the number of hash algorithms that are normally used for packet sampling. Using the previous works [MND05, ZMD$^+$09, HSZ08] as a starting point, three algorithms are used in this section to evaluate the processing performance for packet and flow sampling and correlation. The used algorithms are: Bob-Jenkins implemented in Linux Netfilter code since 2006; CoralReef proposed and used by CAIDA in the flow tool with the same name; and Toeplitz hash proposed by Microsoft and Intel [Mic13, Int12] for Receive-Side Scaling (RSS) in high-speed environments.

To analyze the time consumed by each algorithm when generating a hash value, random byte arrays of different sizes are generated as input for the hash function. Note that, both CoralReef and Toeplitz hashes are applied over 5-tuple (source and destination IP addresses, source and destination ports and protocol) values. In such cases random 5-tuple values are generated. To test the performance of the hash algorithms 1,000,000 unique random hash inputs have been generated. Such inputs have been used to obtain the time devoted to each hash calculation. Figure 4.2 shows the box plot of the hash calculation time for the one million inputs previously generated and the three proposed algorithms. As it can be observed, CAIDA CoralReef hash obtains the best results in terms of per-packet hash time calculation with a median located at 20 ns. To put this figure into perspective, note that, in a 10 Gb/s fully loaded link a 60-byte sized packet is transferred in 67.2 ns :(60 bytes + 4 bytes (CRC)+ 8 bytes (Preamble)+ 12 (Inter-Frame Gap (IFG))) $\cdot 8 \cdot 10^{-10}$. In the case of 1514 bytes, a packet is transferred in 1230.4 ns. In the light of these figure, CAIDA CoralReef hash is able to hash all incoming packets in a fully loaded 10 Gb/s link even in the worst case. The main problem of such approach is that CoralReef hash only accepts 12-byte inputs. This hash function is useful when performing flow-level sampling but it cannot be used for packet sampling as different packets from a same flow produce the same hash output. To solve such problem other algorithms such as Toeplitz or BOB hash may be used.

For comparison purposes, the performance of such algorithms is also showed on Figure 4.2. As it turns out from the analysis, both BOB and Toeplitz hash are able to deal with a 10 Gb/s fully saturated link. In the case of Toeplitz hash, although the algorithm allows the hash calculation over an extended input, depending on the implementation is only applied over the 5-tuple [Int12] —13 bytes. For such reason, BOB hash has emerged as the most recommendable choice for packet sampling whereas CoralReef is the

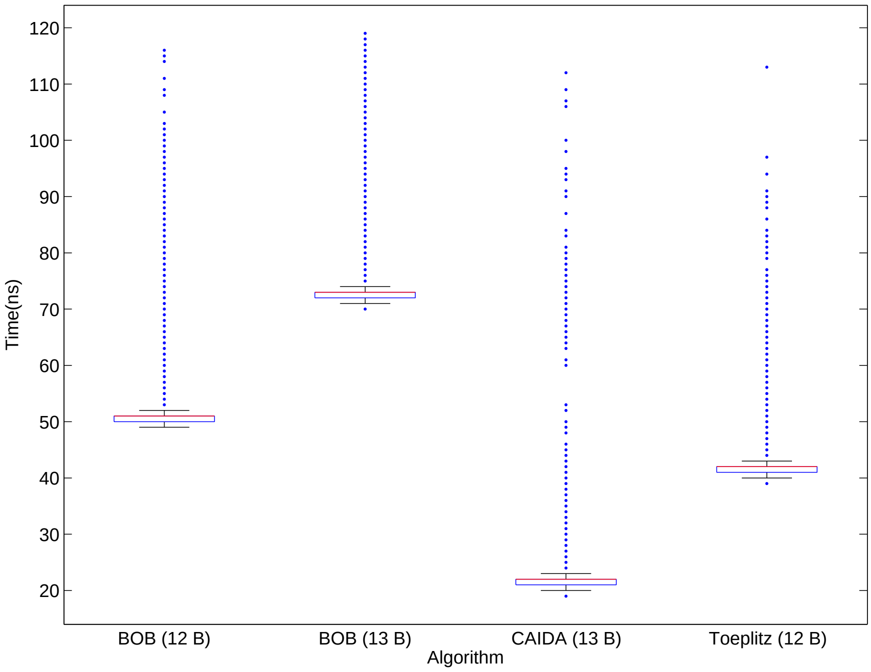


Figure 4.2: Boxplot hash time for different algorithm

most recommendable choice for flow sampling. CoralReef may be replaced by Toeplitz for flow sampling and indexing when hash is hardware calculated an provided by the Network Interface Cards (NIC) as in the case of Intel cards [Int12].

In Figure 4.3 the BOB hash calculation time box plot is shown. Several inputs with different size have been used ranging from one byte to 48 bytes performing 1,000,000 algorithm runs per input size. As it can be observed, the time needed by the BOB hash algorithm increases as the size of the input grows in blocks of 12 bytes. Such behavior is due to the internal mechanisms of the algorithm that groups input bytes in 12-byte chunks to apply logical bitwise operations.

When the BOB algorithm is to be used for packet sampling and correlation, it must be applied over a set of invariant fields that uniquely identify each packet. Such fields are commented in Section 4.2.2 and consist of: IP version (1 byte), IP protocol (1 byte), IP identification field (2 bytes), source and destination IP addresses (8 bytes), source and destination TCP/UDP ports (4 bytes) and 20 first bytes of payload. Putting together all these fields the resulting hash input is 36 bytes long. Observing Figure 4.3, the median for 36-byte inputs is located near 96 ns. In the worst case 10 Gb/s scenario, 60 byte packets, near 10.4 million packets per second may be processed for packet tracking using BOB hash over 36-byte inputs. Note that, trajectory sampling is not designed to be applied over all the traversing traffic of a link but to monitor a reduced set of packets. In this same vein, the obtained results pave the way for high-speed packet tracking systems as near the 70% of the traffic of a 10 Gb/s full-saturated link may be marked using BOB hash.

To evaluate the empirical collision probability, 400,000,000 unique random keys are generated as inputs for the previously presented hash functions. The resulting hash values are analyzed and the empirical collision rate is displayed on Table 4.2. The figures show a similar collision rate between all the analyzed hashes. Note that, in a high speed link in the worst case scenario, the packet rate is 14.88 Million Packets per Second (Mpps). In such case near 380,000,00 unique packets may be received and hashed without any collision resulting in a 25 second time window for packet correlation.

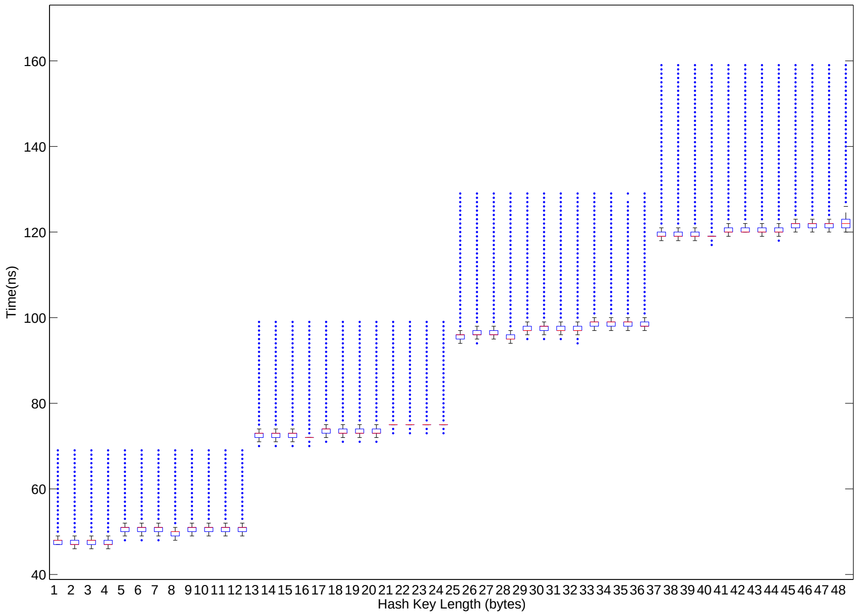


Figure 4.3: Boxplot hash time using BOB

Hash Function	Empirical Collision Rate (%)
BOB (12 Bytes)	4.52
BOB (13 Bytes)	4.52
CAIDA	4.53
Toeplitz	4.51

Table 4.2: Empirical Hash Collision Rate

Hash Function	Packet Sampling	Flow Sampling
BOB	✓	✓
CAIDA	✗	✓
Toeplitz	✗^{2}	✓

Table 4.3: Hash Summary

4.3.3 Packet Correlation

In section 4.3.2, the hashing techniques for packet and flow tracking were presented. When packets are being sampled in a multipoint architecture, packet records are generated at each measurement point. Such records typically contain a hash value calculated over the packet content, the size of the sampled packet, the arrival timestamp of the packet and a unique identifier of the measurement point where the record was generated. Such information is typically transmitted to a collector point for its further analysis. These analyses involve correlating all the received records as well as maintaining the records temporally ordered to obtain reliable statistics. Such process is a demanding task, especially when using general purpose hardware as the one present on heterogeneous environments. In this section an architecture for packet correlation and statistics calculation is presented. The proposed architecture, FPC, is based on a double hashing technique to reduce the memory footprint of the proposal while maintaining a low collision rate and low processing requirements. In addition to the information contained in the standard packet records, the inclusion of the result of a second hash function is proposed. Such hash function should be applied over the 5-tuple data and is used to avoid the collisions produced by the first hash function. The architecture of FPC is shown on Figure 4.4.

For each packet record received, the hash value calculated over the packet content is extracted and used as index in a record hash table. To address the problems derived from the collision of the hash, a linked list is used. Each node of the linked list contains the result of the second hash function calculated over the 5-tuple as well as the packet size. Using such combination of variables, the collision rate is reduced and fewer samples are classified inside a packet stream by mistake. Each node of the list contains a pointer to a data structure used to obtain statistics. Such structure may hold any extra information sent in the packet records. For example, the OWD between each measurement point may be obtained storing the packet timestamp and measurement point identification in a time-ordered list. Other statistics such

[2] Only applied over the 5-tuple

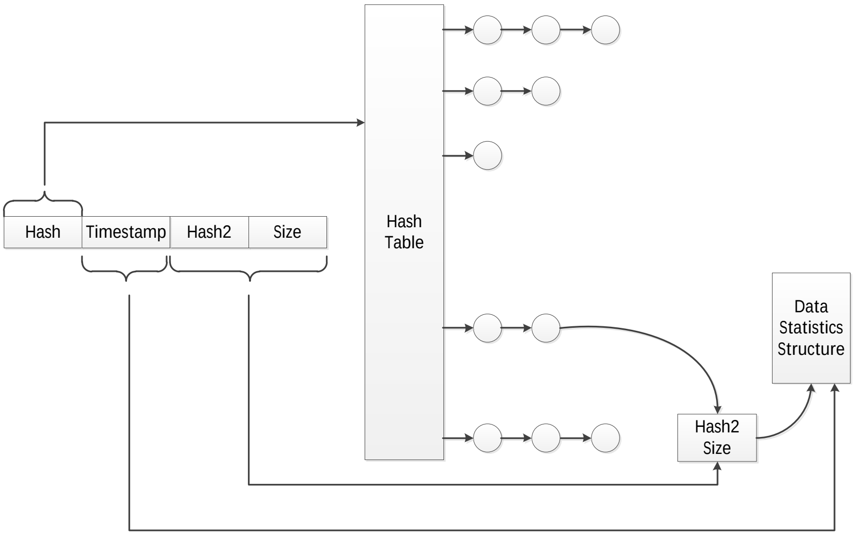


Figure 4.4: Fast Packet Correlation Architecture

as Packet Loss Rate (PLR) may be easily calculated using the generic data structure.

To test the performance of the proposed architecture, CAIDA and Operator trace described at Section 4.3.1 are used. To simulate the multipoint measurements, the traces have been modified decreasing the Time To Live (TTL) value and adding random exponential delays at each simulated hop. In the experiment four hops are simulated and the maximum, minimum and average OWD is calculated. The hashes used for the tests are BOB (applied over the packet content) and Coral Reef —applied over the 5-tuple. In the case of BOB hash the input size is 36 bytes while in the CoralReef case the input data size is 12 bytes. The record hash table used is composed of 2^{24} entries of 8 bytes each which conforms an initial memory footprint of near 128 MB.

Figure 4.5 shows the achieved rate in packets per second of the FPC method applied to calculate the OWD across the four simulated hops in the two proposed traces. For the sake of completeness, the experiments were repeated 1000 times over the previously commented traces. Figures show that between 1.1 and 1.3 Mpps may be processed by FPC method using CAIDA trace and between 0.9 and 0.95 Mpps using the operator trace. Note that the proposed method is not designed to analyze all packets traversing a high-speed link but only the ones selected for tracking. For instance, taking into account the average packet size of the CAIDA trace, 655 bytes, the FPC method is able to process between 6.1 and 6.8 Gb/s. In the case of the operator trace, using the average packet size (258 bytes), the achieved rates are between 1.8 and 1.96 Gb/s. Attending to the empirical collision rate, using the combination of the two hash functions, the obtained results approximate to 1% of collision in both cases.

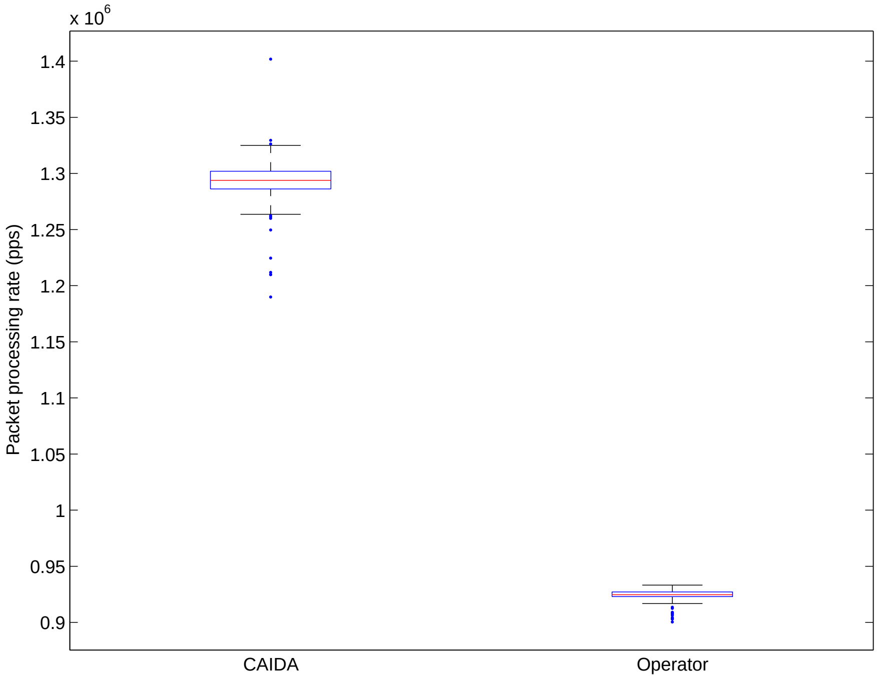


Figure 4.5: FPC effective processing rate (pps) for different traces

4.4 FlowProcess and FlowLib: Flow tracking and analysis

To exploit the potential of the ideas described on the previous sections a software library and an application have been developed. Such software pieces allow the user the online/offline processing of either PCAP or live traffic in a simple and efficient way. Such processing includes:

- Flow creation.
- Flow statistics maintenance.
- Flow and packet sampling using either random or deterministic approaches.
- Flow export using extended NetFlow format or IPFIX.
- General statistics export using MRTG-like format.

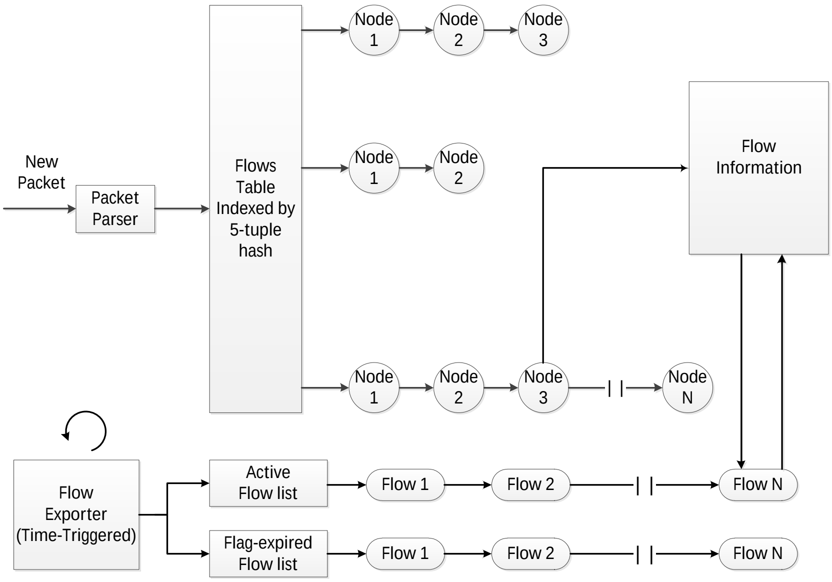


Figure 4.6: FlowProcess Architecture

Figure 4.6 shows the architecture implemented on the FlowProcess and FlowLib tools. Such architecture is optimized to work at high-speeds making use of some improvements such as memory structures pre-allocation and reuse or efficient hash distribution. The workflow of the system is as follows:

1. A packet arrives at the system.

2. In case that packet sampling is used, a sampling function is applied over the arriving packet.

3. If the packet has not been discarded by the sampling function, it is parsed to extract relevant information such as source and destination IP address, source and destination ports, protocol, TCP specific fields (flags, window size) and the first N bytes of payload.

4. A hash function is applied over the 5-tuple of the packet. The result of such function is used as index in a pre-allocated flow table. Each table entry contains a linked list to handle hash collisions. The linked list is composed of nodes each one of which contains a pointer to a flow structure. Both nodes and flow structures are pre-allocated

5. If the flow exists in the linked list, statistics and flow information are updated with the information extracted from the last packet.

6. In case the flow does not exists and flow sampling is active, a sampling function is applied over the previously extracted packet fields. If the sampling function selects the flow for insertion, a node and flow structure are assigned and the node is inserted in the active flow list. Such list contains pointers to the active flow structures and is ordered by last access time facilitating the export and cleanup of the active flows. Each time a flow is created or updated, its corresponding active list node is promoted to the list head.

7. Periodically a garbage collector is executed to cleanup and export flows. The garbage collector traverses trough the active flow list in reverse order checking the last packet timestamp of each flow. If the flow has

not received packets in a certain configurable time period P, the flow is considered expired, its memory structures and nodes are released and its information is exported. Additionally unidirectional TCP flows may be expired when a FIN or RST flag is present on either flow creation or update. In such scenario, nodes are removed from the flow hash table and active flow list and a node is inserted into a separate expired flag flow list which may be analyzed by the garbage collector for flow removal and exportation.

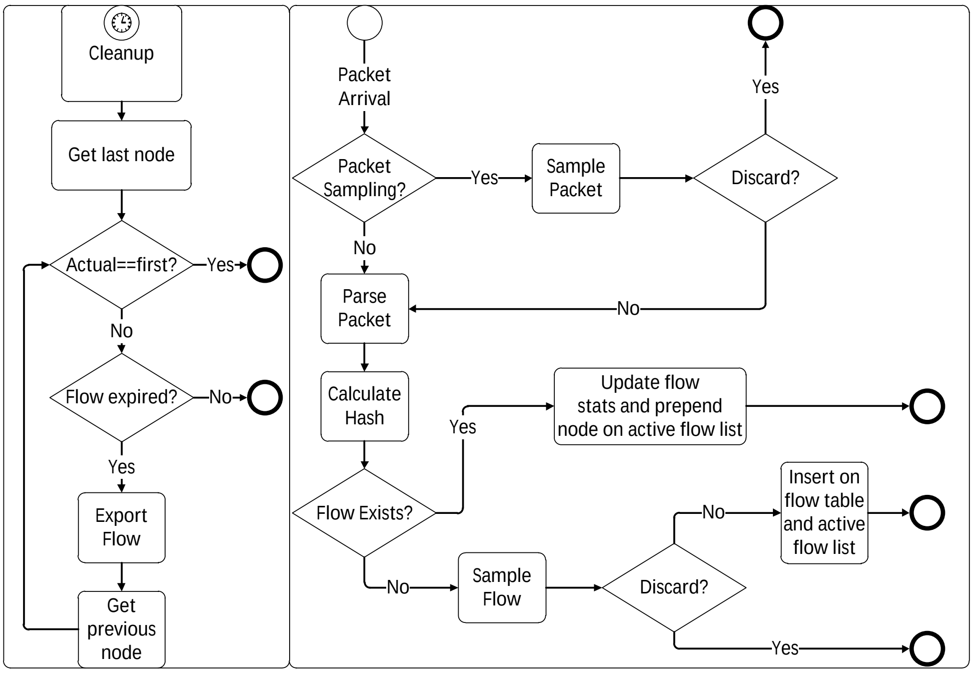


Figure 4.7: FlowProcess Workflow

Figure 4.7 summarizes the process of flow insertion and update as well as the flow cleanup process. FlowProcess an FlowLib can also handle IP fragmented packets for payload extraction and statistics accounting. Additionally, FlowProcess and FlowLib are able to track sessions, that is, bidirectional flows with minimal changes —the hash function used for table index calculation and the flow existence function. Besides the per-flow exported information, FlowProcess and FlowLib are able to export MRTG-like statistics with configurable time granularity. Such statistics include the number of bytes, packets and concurrent flows in the specified time interval as well as the timestamp in UNIX format. Both MRTG-like statistics and flow information can be exported to a file or to a remote collector in an extended NetFlow or IPFIX format.

Using the exported flow information and MRTG-like registers, other applications or utilities may calculate and estimate QoS parameters. The next parameters can be directly calculated from the generated output of FlowProcess or FlowLib:

- Bandwidth: using the MRTG-like statistics with granularity 1 second, the used bandwidth can be obtained. Additionally, flow registers may be used to construct an approximated bandwidth time series with low error as shown in [OSSSP12].

- RTT: using the first N packet timestamps contained in the extended NetFlow format provided by the tool in combination with the correlation of incoming and outgoing flows, RTT may be calculated with microseconds precision. Additionally, the maximum, minimum and average RTT can be calculated depending on the number of stored timestamps.

- Packet Loss: although packet losses may be estimated using passive methodologies based on multipoint measurements, on single point measurements like the ones provided by the tool, only TCP losses can be estimated. Such metric is provided by the tool, based on a simple TCP sequence number and ACK analysis. Using the information exported at several measurement points, the packet loss may be estimated correlation such information as shown in [RSDC11].

Other metrics such as OWD require the use of trajectory sampling techniques at different measurement points. To this end, a software library called Network Measurement Library (NMLib) has been developed as a result of this work. The library integrates FlowLib and adds extended capabilities to implement trajectory sampling for multipoint measurements. Using the implemented capability of hash-based packet sampling, NMLib is able to export per-packet registers containing the timestamp of the packet, the 32-bit calculated hash, the packet size and a per-node unique identifier. Such information is sent to a collector point or stored in a packet register file for further correlation. The collector process implements the FPC method described in Section 4.3. NMLib is described in detail in Chapter 6.

To evaluate the performance of the developed solution, the two traffic traces described on Section 4.3.1 are used as input. Both traces are loaded in a Linux ramdisk in order to avoid the performance degradation produced by disk reading and evaluate only the performance of the flow management system. Using such solution, 100 executions of the program are performed over each trace to obtain the packet per second processing rate of the solution. Note that, the performance of the proposed system depends on the number of concurrent flows, aggregation level and traffic characteristics.

In the performed tests, FlowProcess was producing simultaneously MRTG-like and extended NetFlow records. Every extended NetFlow record contains:

- Source and destination Ethernet addresses.
- Source and destination IP addresses.
- IP protocol field.
- Source and destination TCP/UDP ports.
- Number of bytes and packets.
- Timestamp of the first 10 packets of the flow.
- Size of the first 10 packets of the flow.
- Mean and standard deviation of the flow packet size.
- Number of observed TCP flags (SYN,FYN,RST,ACK,PSH,URG).
- Time between the first observed SYN packet and the first ACK packet.

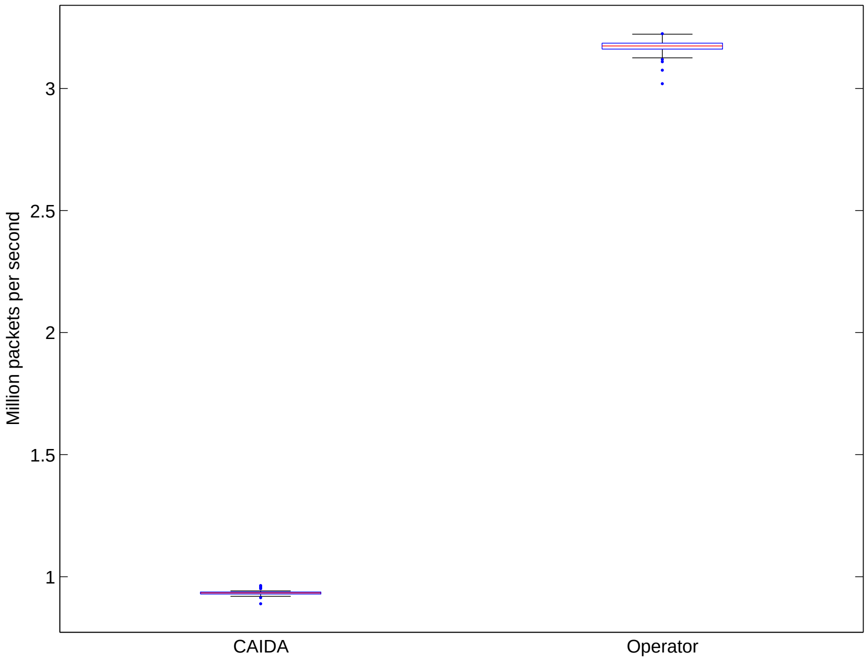


Figure 4.8: FlowProcess performance

- Minimum, maximum and average packet inter-arrival time of the flow.

Figure 4.8 shows the packet processing rate box plot for 100 executions of FlowProcess. As the figures show, in the case of CAIDA trace, near 0.93 Mpps are processed. Taking into account the average packet size, 655 bytes, the average processing throughput is near 4.87 Gb/s. In the case of the operator trace, almost 3.17 Mpps are processed with an average throughput of 6.54 Gb/s. In both cases, the achieved processing rate surpasses the average rate for each trace which shows that the proposed system is able to handle all the traversing traffic of a backbone 10 Gb/s link. As the number of concurrent flows increases, the processing rate decreases due to the overhead of list and table lookups. To solve this problem, the amount of memory dedicated to the flow hash table may be increased, reducing the collision rate and, thus, the average lists lengths. Table 4.4 shows the maximum memory footprint for the 100 executions of FlowProcess over the two different traces. The results show a low memory usage (less than 3 GB in the worst case) which is reachable by any standard computer and fits perfectly into the heterogeneous networks paradigm.

Trace	Avg. Memory (MB)	Std.Dev Memory (MB)
CAIDA	2,890	0.0122
Operator	1,251	0.0428

Table 4.4: FlowProcess memory footprint

4.5 Conclusions

In this chapter, the study of passive monitoring methods has been addressed. Firstly a review of the most popular passive monitoring techniques has been performed focusing in flow-level and packet-level monitoring. Such review has shown that each technique has its benefits and drawbacks and has led the way to the development of multi-granular passive monitoring systems. Such systems allow the application of the best methods to estimate each QoS parameter according to the precision needed and the available resources. In the wake of developing a multi-granular system, flow-based monitoring and packet-tracking techniques have been selected as the leading examples for passive monitoring.

To evaluate the adequacy of packet-tracking techniques to heterogeneous scenarios, a brief analysis of the fundamentals has been presented. First, an initial set of hash functions have been evaluated in terms of performance and collision probability. The results of such analysis have shown that BOB hash function may be used to sample packets and generate packet records achieving a 10.4 Mpps in the worst case 10 Gb/s scenario —60 byte packets. Next, an architecture for packet correlation called FPC has been proposed and presented as a contribution in this chapter. Such proposal has been tested using an example for OWD calculation over real traffic from CAIDA and a Spanish operator. The proposed system has achieved near 6.8 Gb/s processing rates with a standard computer and low base memory footprint —128 MB.

Regarding flow-based approaches, a novel system for flow creation and statistics gathering has been presented. Such system, called FlowProcess, makes use of efficient memory structures and optimization techniques to produce extended NetFlowIPFIX records and MRTG-like statistics that may be used to estimate QoS parameters. The proposal has been tested with real traffic achieving rates between 4.87 and 6.54 Gb/s.

Finally, to obtain an efficient multi-granular passive monitoring system that fits into the PM methodology the integration of FPC and FlowProcess into a measurement library called NMLib has been done constituting the third contribution of this chapter.

Chapter 5

Traffic Pollution Detection

This chapter provides the background and revises the most relevant works related to the traffic pollution detection. Additionally, a method for traffic pollution detection using minimal information from time series is provided along with its corresponding validation. The structure of the chapter is as follows. First, a brief description about traffic pollution detection and general concepts is done in Section 5.1. Then in Section 5.2 the fundamentals of the traffic pollution detection model are presented along with an initial validation of the hypotheses. In Section 5.3, an online algorithm for traffic pollution detection based on time series analysis is presented and validated. Finally, in section 5.4 some conclusions and ideas are presented.

5.1 Introduction

Nowadays, communication networks transport different kinds of protocols and services complicating the traffic monitoring. Moreover, in recent years new technologies with special network condition requirements have arisen such as Voice over Internet Protocol (VoIP), Internet Protocol Television (IPTV). Usually, such services are provisioned over dedicated networks to minimize the impact of interfering traffic and control the traffic in a strict way. Such scenario is even more common in corporate environments where dedicated resources are used to provide this kind of services. One example is the case of bank networks where some Virtual Local Area Network (VLAN) or subnets are devoted to electronic bank transactions which need reliable, secure and low-latency communications. In such transactional environment, the traffic is controlled and restricted to a small set of protocols and services. Thus, incorporating external extra traffic may pollute the controlled environment producing undesirable effects. Unfortunately, avoiding the pollution is a difficult task as such transactional networks comprise different devices such as Automated Teller Machines (ATM), Point of Sales (POS) or standard computers. Moreover, since such networks are not completely and physically isolated from other networks some polluting traffic may slip into the network due to maintenance (backup process), misconfigured devices and applications or security flaws.

In the case of VoIP networks, if polluting traffic is generated, the One-Way Delay (OWD) and Packet Loss Rate (PLR) may be increased, with the subsequent performance degradation of the ongoing calls. In both domestic and corporate scenarios it is very likely to find voice and Internet traffic mixed up together on the same access router. In such case, it is important to assure that data traffic does not pollute voice traffic.

Several works [BKPR02, SDTG10, XCF12, LSK10, TMH11] have addressed similar problems focusing especially on anomaly and attacks detection. On one hand, some of these works are based on Principal Component Analysis (PCA) or similar techniques and require a pre-training in order to obtain reliable results. On the other hand, other works need large amount of data and the complexity of the proposed methods is computationally unaffordable in heterogeneous environments. Additionally, few of such algorithms may work in an online way which increases the time needed until a network manager notices the problems. At the moment of the writing, no literature addressing specifically the traffic pollution detection was found.

In what follows, a description of the root causes and the fundamentals of pollution traffic detection are presented.

The first step for polluting traffic detection consists on the detection of the spikes in the transferred or received bytes time series and their root causes. Such spikes may be produced by:

- Increases in the number of concurrent flows: the creation of several new flows that generate small or medium amounts of traffic may contribute to the increase of the global byte series.
- Increases in the rate of one or few flows: the increase in the bits per second rate of one or few flows may increase the overall byte time series.

In the first case, if the aggregated bytes series suffers an increment and the number of concurrent flows has increased, the increment on the overall traffic is not due to polluting traffic unless the increment on the number of concurrent flows is excessive in a short time period —e.g. SYN flood or similar attacks. In the second case, if the number of aggregated bytes increases and the number of concurrent flows remains constant in a short time period, the increment may be due to the rate increase of one or a small group of flows. In that case, such flows may be considered as polluting traffic. Effectively detecting such situations is not a trivial task and may consume a lot of resources (both processor and memory) analyzing flow rates individually and correlating such information with the number of concurrent flows and aggregated bytes rate. To simplify this task, a model to detect polluting traffic based on the variance of the flow sizes is presented. Using such simplification, the analysis process is reduced to the calculation of the squared bytes sum without additional process or resources.

In this light, an algorithm for online detection of polluting traffic presence is also presented in this chapter. The main advantages of the proposed method are:

- It can be applied to all the observed traffic regardless of the transport protocol —Transmission Control Protocol (TCP) or User Datagram Protocol (UDP).
- Requires minimal information, namely: bytes, number of concurrent flows and flow size variance time series of a given link.
- Requires minimal computational power and memory which fits perfectly into the heterogeneous environment paradigm.

5.2 Initial Hypotheses

Let $x(n), n = 0, \ldots, m$ denote a discrete-time traffic stream that represents unidirectional traffic on a link, measured in bps. In order to determine what are the time epochs at which the baseline traffic is being polluted, it is used a discrete-time stream which represents the flow size variance and is denoted by $y(n), n = 0, \ldots, m$. Let also define a discrete-time stream that represents the number of concurrent flows on a link, denoted by $f(n), n = 0, \ldots, m$.

The main hypotheses of the model are:

1. $f(n)$ remains stable over a given time period t

2. $x(n)$ and $y(n)$ are strongly correlated

Taking into account the previous hypotheses, if an increment is observed in the flow size variance stream over a time period t and the number of concurrent flows is stable within a predefined limit, then an increment in the byte traffic stream must be observed. Such increment is produced by the increment of the rate of one flow or a small group of flows which suggest the presence of polluting traffic such as intensive HyperText Transfer Protocol (HTTP) downloads.

For validation purposes some real-scenario traces are used. The traces were collected in a commercial core bank network which contains traffic from several services and networks. Such traces contain electronic bank transactions mixed up with batch traffic from backups using File Transfer Protocol (FTP), XCOM data transfer protocol and Lightweight Directory Access Protocol (LDAP) traffic along with HTTP and HyperText Transfer Protocol Secure (HTTPS). Additionally, VoIP traces collected at the same bank network are used in the validation process. Such traces contain Skinny Client Control Protocol (SCCP)/Real-time Transport Protocol (RTP) traffic in combination with batch TCP traffic from users. Some Video on Demand (VoD) and commercial traces from a Spanish operator are also used on the validation process. In all cases, additionally to the minimal data required by the algorithm, the NetFlow output of the traces and the Packet Capture (PCAP) traces are used to as ground truth to verify the nature of the polluting traffic.

All the used traces have a 24 hour duration which covers all the possible load scenarios presented in a day. Table 5.1 shows the main information about the trace sets used.

Trace Set	Number of traces	Avg. Rate (Mb/s)	Max Rate (Mb/s)	Additional Info
Bank1	30	6.4	82.3	LDAP,FTP,XCOM and bank transactions
Bank2	30	3.7	78.6	LDAP,FTP,XCOM and bank transactions
Bank3	30	3.5	68.3	LDAP,FTP,XCOM and bank transactions
Bank4	30	0.1	5.9	LDAP,FTP,XCOM and bank transactions
VoIP1	5	1.5	75	SCCP,RTP,FTP and HTTP traffic
VoIP2	5	0.4	16	SCCP,RTP,FTP and HTTP traffic
VoD	2	33.11	145	RTP and HTTP
Operator	4	3	17.3	-

Table 5.1: Characteristics of the traffic data sets

For each set of traces, the cross-correlation value between $x(n)$ and $y(n)$ is calculated at hourly intervals to assure the applicability of the model to different types of traffic. Note that, the cross correlation value is only calculated over the time periods in which the number of concurrent flows remains stable. The closer to one the cross-correlation value is, the more correlation exists between $x(n)$ and $y(n)$.

Figure 5.1 shows the Empirical Cumulative Distribution Function (ECDF) of the cross-correlation value calculated over the bank traces described on Table 5.1. The cross-correlation value is calculated using the time series of aggregated bytes and flow size variance corresponding to 30 days of bank traffic. Note that the value of the cross-correlation value at lag 0 is greater than 0.7 for the 70% of the cases.

Figure 5.2 shows the ECDF of the cross-correlation value over the VoIP,VoD and commercial traffic traces. Attending to the figures it can be observed that the value of the cross-correlation value at lag 0 is greater than 0.7 for the 75% of the cases. All the presented results support the applicability of

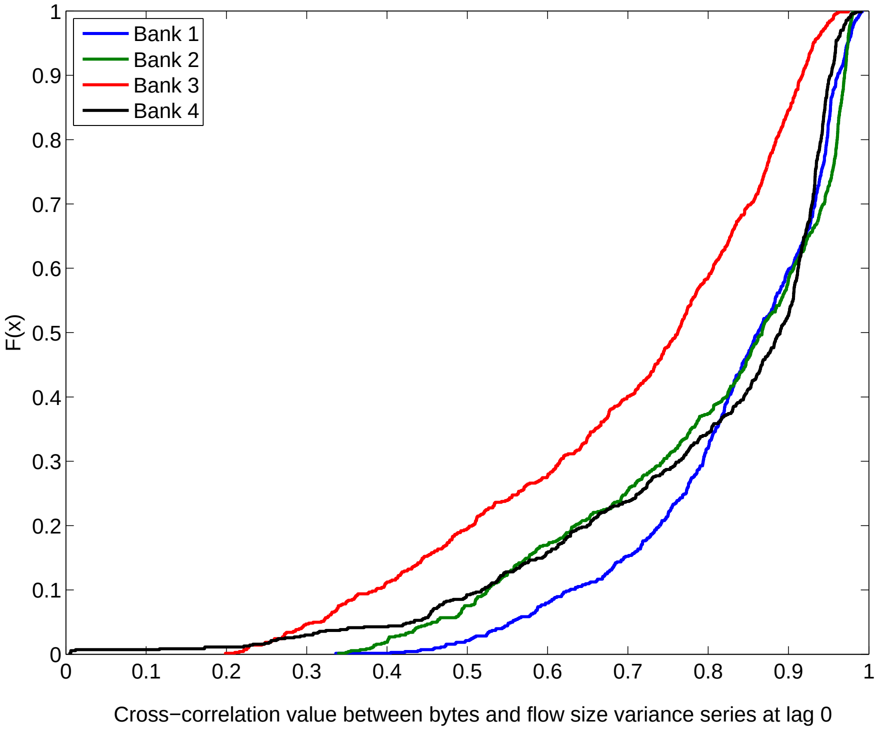


Figure 5.1: ECDF cross correlation value between bytes and flow size variance at lag 0 for bank traces

the proposed model. In the next section, an online algorithm for pollution detection based on the verified hypotheses is presented.

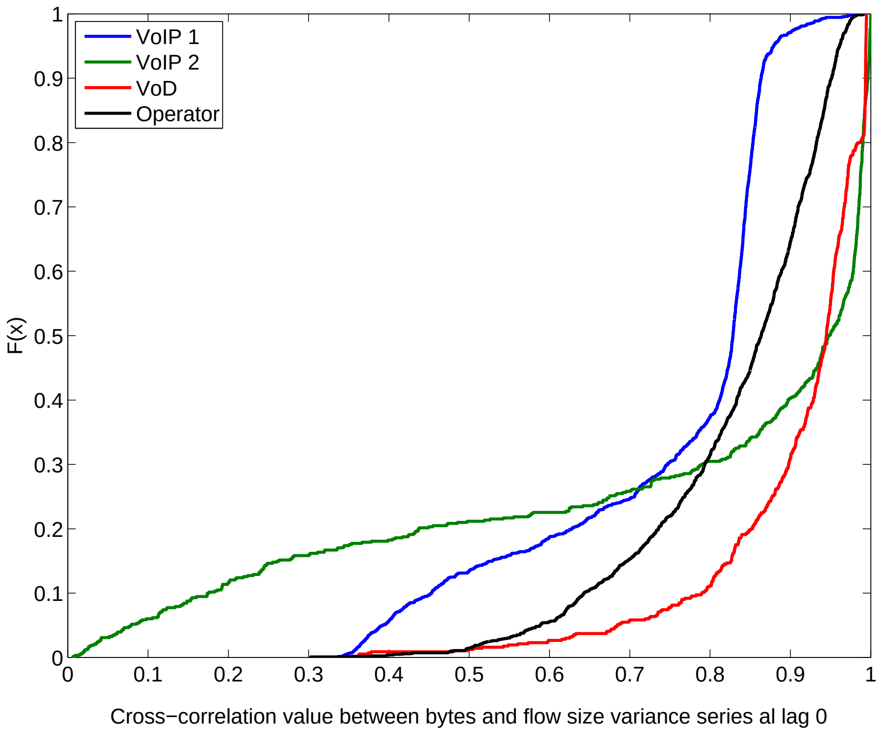


Figure 5.2: ECDF cross correlation value between bytes and flow size variance at lag 0 for other traces

5.3 Online Algorithm

In this section, an online algorithm is proposed based on the idea behind the model presented in the previous section. The algorithm follows a sliding window approach to detect spurious increments in the bytes and the flow size variance for each time step. The proposed time step in this case is one second as is the standard minimum granularity provided by tools such as Multi Router Traffic Grapher (MRTG). The input parameters of the algorithm are:

- t: size of the time window.
- bytes time series.
- flow size variance time series.

The output parameter of the algorithm is:

- estimator: a value for the estimator of the presence of polluting traffic. Such parameter takes either zero or unity value if there is no polluting traffic or if there is polluting traffic respectively.

Algorithm 2 shows the pseudocode of the proposed algorithm for its further implementation. Note that the function percentile represents the percentile procedure calculation. The method is based on a similar approach of the spear detection method described in [Pal09]. Such method relies on the calculation of the difference between the value of an observed point and the average value of the N closest previous and subsequent values. Equation 5.1 shows the method used to calculate the spear coefficient at point i.

$$SC(i) = \frac{x_i - \frac{\sum_{k=1}^{N} x_{i-k}}{N} + x_i - \frac{\sum_{k=1}^{N} x_{i+k}}{N}}{2} \tag{5.1}$$

To detect global spears and eliminate local increments, a threshold based method is proposed. Such threshold is based on the capacity of the monitored link. Usually, such data is previously known and in case of absence it can be estimated observing the weekly traffic pattern. Using such approximation, only points with a spear coefficient greater than a certain capacity percentage are marked as polluted instants.

Note that, in Algorithm 2, the concurrent flow stability test is not showed for simplicity. Such stability test is based on the use of Cumulative Sum (CUSUM) methods along with a threshold derived from the observation of the number concurrent flows of the monitored network.

Algorithm 2 Online Traffic Pollution Detection

```
nsamples=0
while new samples do
  Add new sample to byte vector
  Add new sample to flow variance vector
  nsamples++
  if (nsamples > t) then
    $x_0$=bytes[1:(t/2)-1]
    $x$= bytes[t/2]
    $x_1$=bytes[(t/2)+1:t]
    $y_0$=var[1:(t/2)-1]
    $y$=bytes[t/2]
    $y_1$=bytes[(t/2)+1:t]
    $e_0$=((x-mean($x_0$))+(x-mean($x_1$)))/2
    $e_1$=((y-mean($y_0$))+(y-mean($y_1$)))/2
    if ($e_0$ > byte_threshold) OR ($e_1$ >variance_threshold) then
      estimator(nsamples)=1
    else
      estimator(nsamples)=0
    end if
  else
    estimator(nsamples)=0
  end if
end while
```

5.3.1 Algorithm Validation

To validate the proposed algorithm, using NetFlow records and original PCAP files, the time intervals with polluted traffic presence have been identified creating a ground-truth time series of the polluted time intervals. Note that, the baseline traffic is known in each scenario and it can be differentiated from polluting traffic. The applied threshold is based on the previously known capacity of each of the monitored links.

For each trace of the trace set described in Section 5.2, the algorithm has been executed using a threshold of 30% of the link's capacity and a time interval t of 60 seconds. To represent the validation results, Receiver Operating Characteristic (ROC) graphs are used. Note that, in these kind of graphs the False Positives Rate (FPR) or (1-accuracy) is represented on the x-axis and the True Positives Rate (TPR) or sensitivity is represented on the y-axis. Such values are calculated using the False Positive (FP),False Negative (FN),True Positive (TP) and True Negative (TN) values. Equations 5.2 and 5.3 show how these ratios are calculated. In ROC graphs, the lower the value of FPR is, the better the algorithm is. Similarly, the higher the TPR value is, the better the algorithm is.

$$TPR = \frac{TP}{TP + FN} \tag{5.2}$$

$$FPR = \frac{FP}{FP + TN} \tag{5.3}$$

Figure 5.3 shows the ROC graph for the results of the application of the online algorithm over the bank traces. The results show a high TPR ranging from 50% to 100% with most of the samples located between 90% and 100%. Attending to the FPR, promising results are achieved ranging from 0% to 1.4%.

Figure 5.4 shows the ROC graph for the results of the application of the online algorithm over the VoIP, VoD and commercial traffic traces. In this case, the achieved TPR ranges from 80% to 100% with most of the samples located between 98% and 100%. The obtained FPR in this case, ranges from

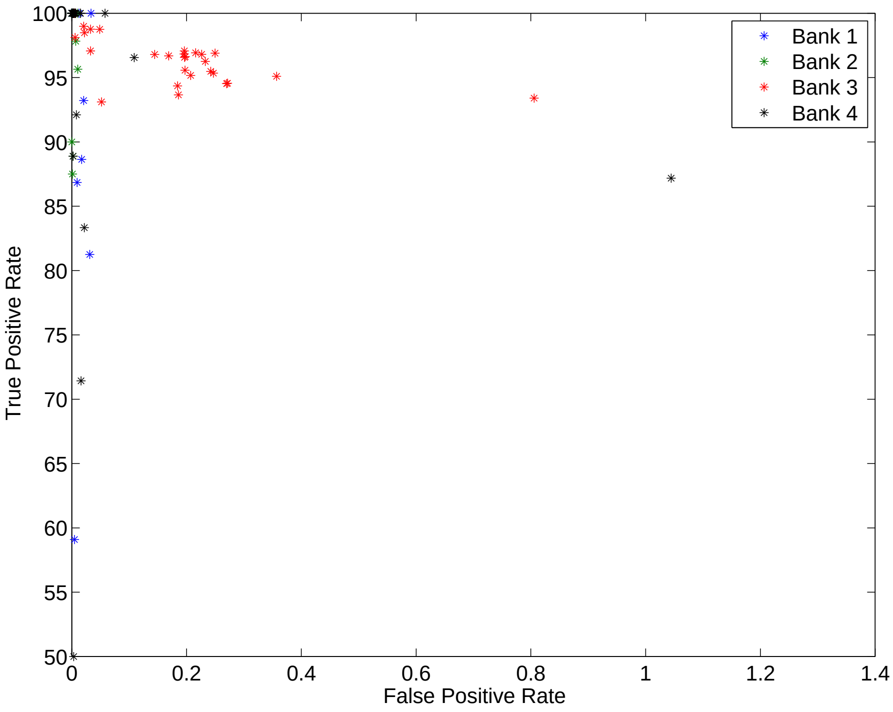


Figure 5.3: ROC graph for bank traces using th=30%

0% to 3% with most part of the samples located between 0% and 0.5%.

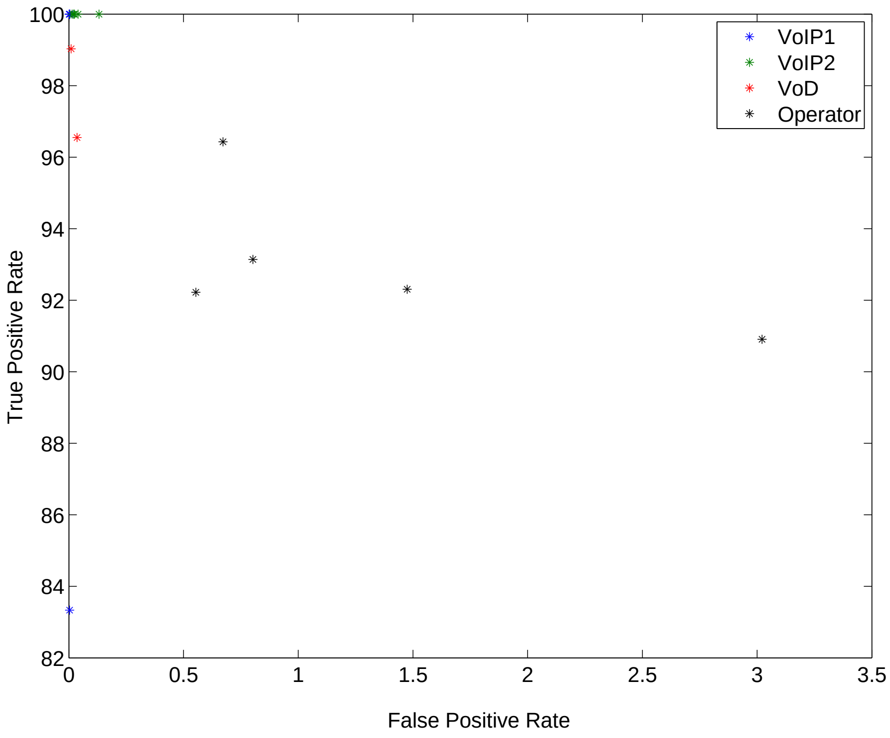


Figure 5.4: ROC graph for other traces using th=30%

In the light of the obtained results, the proposed method is able to detect traffic pollution time intervals based on the analysis of the bytes and flow size variance time series with FPR lower than 3%. These promising results are supplemented with a low memory footprint (only the last 60 samples are necessary) and Central Processing Unit (CPU) usage due to the simplicity of the algorithm.

5.4 Conclusions

In this section a method for traffic pollution detection based on time series analysis has been presented. The proposed method is based on the existing correlation between the flow size variance and the aggregated byte rate. First, a brief review of the fundamentals and similar works has been done. Next, the hypotheses of the detection method have been proposed and explained. Such hypotheses have been validated by means of cross-correlation tests using real traffic traces obtained on a bank and a Spanish operator. Then a simple algorithm for traffic pollution detection based on a time series spear identification method has been proposed. Such algorithm makes use of the aggregated bytes and flow size variance time series to detect the time interval in which polluting traffic exists. Finally, the algorithm has been validated using the real traffic traces obtaining promising results with FPR lower than 3% and an average TPR greater than 90%. The developed algorithm fits perfectly in the heterogeneous networks paradigm as makes use of minimal resources in terms of memory and CPU usage.

Chapter 6

Proactive System

This chapter describes the difficulties presented when integrating active and passive measurements into a proactive system. First in Section 6.1 a brief introduction to the proactive measurement selection problem is presented. Next, in Section 6.2 a simple algorithm for measurement triggering is presented. Such algorithm makes use of time series analysis and statistical methods to obtain an estimator of the time epochs in which active monitoring must be carried out. In Section 6.3, Proactive Network Monitoring Framework (PNMF) is presented and described. This tool allows the integration of active and passive measurements into a simple and powerful Proactive Monitoring (PM) system. Finally, in Section 6.4 some conclusions are presented and the chapter is closed.

6.1 Introduction

In chapters 3 and 4, different measurement methodologies are presented. Using active techniques only, reliable measurements may be obtained at the expense of generating extra traffic and interfering on the measured network. On the other hand, passive techniques provide measurements that lack of precision and specificity. For example, the estimation of some Quality of Service (QoS) parameters such as Round-Trip Time (RTT) or Packet Loss Rate (PLR) is very complicated as few information is gathered at several measurement points. Such information must be correlated and analyzed in order to obtain QoS parameter estimations. In case that the monitored network is composed by a large number of measurement points, a prodigious amount of data is produced and sampling or tracking methodologies must be applied in order to focus on certain traffic. Such techniques mask the behavior of the network and degradation on the quality of the communications in certain points may not be detected. To tackle this task while maintaining a trade-off between the reliability of the measurements and the intrusiveness in the monitored network, PM is proposed. PM goes beyond the simple combination of the active and passive methodologies and adds more functionality than Reactive Measurement (REM) systems. The main idea behind the PM is planning the measurements in such a way that the switching between measurement methods is done automatically when a change is detected. For example, a system may be passively monitoring a set of reduced QoS parameters (base parameters) such as available bandwidth of one or more networks or One-Way Delay (OWD) over a small set of paths or connections until a change is detected or a certain threshold is exceeded. In that moment active measurements over the problematic networks are triggered to obtain estimations of other QoS parameters such as PLR or RTT in addition to base estimation parameters —i.e. available bandwidth or OWD. Depending on the obtained results the active tests are performed until the base parameters return to their standard values or a certain monitoring time is expired. PM triggering is useful not only when anomalous situations are presented but also when dimensioning must be done in non-production or test environments. The use of the PM triggering mechanisms allow the operators and network designers to easily perform tests and push the network to the limit while obtaining reliable measurements of QoS parameters.

6.2 Measurement Planning: switching from passive to active

Triggering between passive and active measurements methodologies before certain situations are produced is a desirable behavior for PM systems. Such measurements must be controlled in order to reduce the impact on the network and the necessary running time. To tackle this task, a simple and efficient method based on passive monitoring data must be applied to increase the event response time and reduce the computational complexity. In this light, a simple triggering mechanism is proposed based on the analysis of the byte time series and simple statistical features.

In order to determine the time epochs when active measurements must be executed an approach based on the Jensen-Shannon Divergence (JSD) is used. The JSD provides a metric of the similarity of two given probability distributions P and Q. This method makes use of the Kullback-Leibler Divergence (KLD) to provide a symmetric and finite value of the similarity of the distributions. Equations 6.1, 6.2 and 6.3 show how JSD and KLD are calculated. According to [Lin91] the value of JSD is bounded by [0,1].

$$JSD(P \parallel Q) = \frac{1}{2}KLD(P \parallel M) + \frac{1}{2}KLD(Q \parallel M) \tag{6.1}$$

$$M = \frac{P+Q}{2} \tag{6.2}$$

$$KLD(P \parallel Q) = \sum_i ln\left(\frac{P(i)}{Q(i)}\right)P(i) \tag{6.3}$$

Let $x(n), n = 0, \ldots, m$ denote a discrete-time traffic stream that represents unidirectional traffic on a link, measured in bps. Let also define W as a window length value and $x_k(n)$ as the k-th window for the input stream $x(n)$ with W samples. Calculating the JSD value over the distributions of adjacent sample windows (D_{k-1} and D_k), a metric for detecting changes over $x(n)$ value is obtained. If $\sqrt{JSD(D_{k-1} \parallel D_k)}$ value approaches to one, the distributions of the monitored parameter (unidirectional traffic rate) are different and the time interval defined by the bounds of x_k is a candidate for measurement triggering. In other case, the distribution remained stable and no triggering must be done. In order to isolate spurious changes and avoid detecting non-critical changes, a threshold rejection technique is applied over the $\sqrt{JSD(D_{k-1} \parallel D_k)}$.

Figure 6.1 shows the value of measurement triggering estimator (square root JSD value) of the Bank 4 trace described in Table 5.1 with W=300 seconds and a threshold of 0.6. As it can be observed, the value of the estimator surpasses the given threshold at the time epochs where significant changes are observed on the traffic rate series.

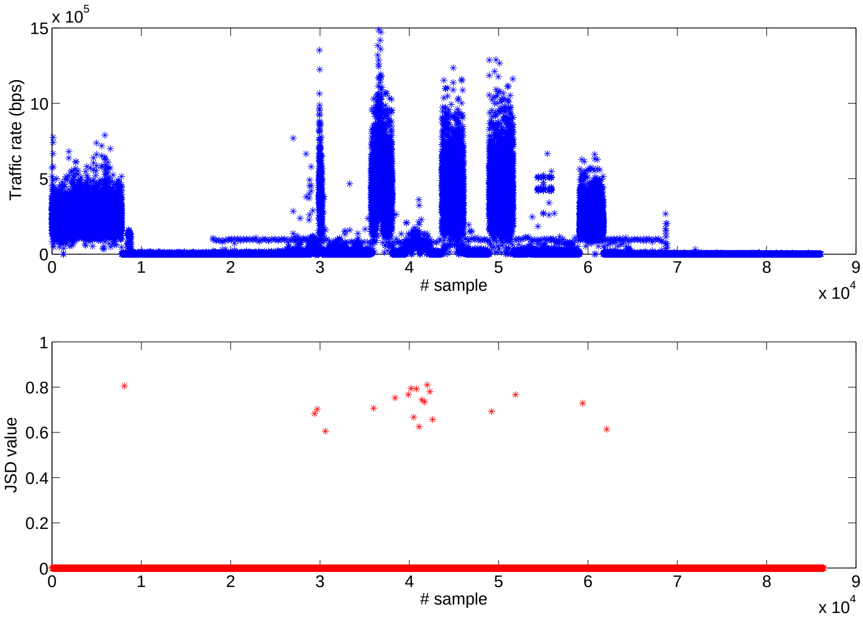


Figure 6.1: Measurement triggering estimator value for Bank 4 trace with W=300 s and th=0.6

The proposed method not only identifies the time epochs where active monitoring must be launched but also the moments that such monitoring must finalize. Such detection is automatically done by the very nature of the JSD-based algorithm. To decide which time intervals must trigger an active monitoring task and which must not, a simple threshold method may be used. When applying the proposed method to the traffic rate parameter a threshold based on the capacity may be selected —e.g. the existing average traffic rate exceeds the 80% of monitored links capacity. Using such a simple mechanism increases the effectiveness of the PM systems as early problem detection can be done.

Algorithm 3 Measurement triggering algorithm

```
while new samples do
    Add new sample to parameter vector
    nsamples++
    if (nsamples > 2*W) AND (mod(nsamples,N) ==0 ) then
        $x_{k-1}$=samples[1:W]
        $x_k$=samples[W+1:2*W]
        $D_{k-1}$=hist($x_{k-1}$)/sum(hist($x_{k-1}$))
        $D_k$=hist($x_k$)/sum(hist($x_k$))
        estimator=JSD($D_{k-1}$,$D_k$)
        if estimator > estimator_threshold then
            if  mean($x_k$) >parameter_threshold then
                if no active monitoring present then
                    launch active monitoring task
                end if
            end if
            if active monitoring present then
                stop active monitoring task
            end if
        end if
    end if
end while
```

The proposed method has been applied over the traffic rate parameter but it can be applied to other parameters, for instance OWD or PLR, depending on the monitoring scenario and the QoS requirements. Algorithm 3 shows the triggering algorithm pseudocode for a generic monitored parameter. Note that the operator *hist(x)* returns the distribution of the vector x among bins and the operator *sum(x)* returns the sum of the elements in the vector x.

6.3 Proposed System

As a result of the work of this thesis, a measurement framework called PNMF has been developed. Such framework groups all the techniques developed and presented in Chapters 3, 4 and 5 providing a set of tools useful for distributed network monitoring. In this section, an overview of the system architecture, modules and implemented features is done.

PNMF is composed by three different modules, namely: probe module, collector module and frontend module. Figure 6.2 shows an example of deployment of PNMF modules.

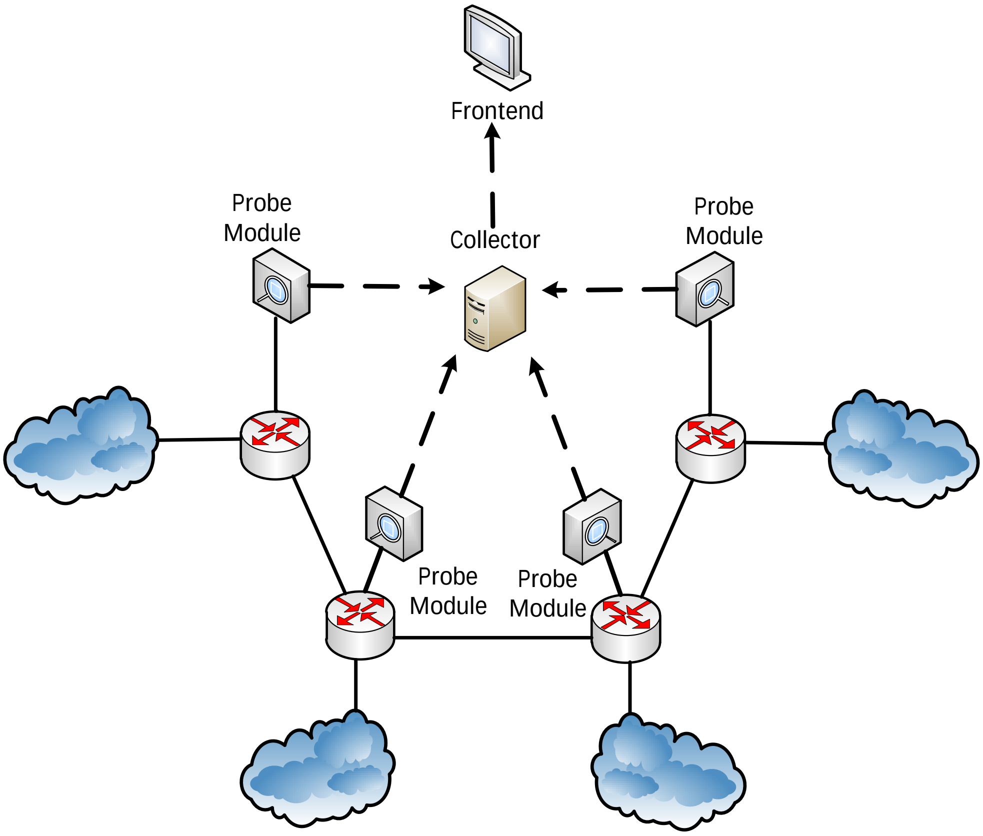


Figure 6.2: PNMF architecture

6.3.1 Probe Module

The probe module is deployed in each of the network points to monitor. The main task of this module is passively obtaining data from the network or performing active tests based on a given set of parameters. The probe module has two operating modes:

- Passive Mode: such mode consists on a packet capture process that creates and maintains a flow table in the same way the Cisco routers do. Whenever a flow expires, it is exported using an extensible mechanism that allows the communication of the information to the collector using different formats such as NetFlow v5 or Internet Protocol Flow Information eXport (IPFIX). Additionally, the passive mode constantly monitors the incoming and outgoing traffic rate and number of packets to generate Multi Router Traffic Grapher (MRTG)-like statistics which are exported to the collector. The exportation time is configurable to adjust the aggregation granularity. The passive methods used are based on the contributions explained in Chapter 4.
- Active Mode: in this mode, the probe module is able to either generate or receive measurement traffic to perform active measurements. Such measurements include the techniques proposed on Chapter 3 to estimate the capacity, OWD, RTT and packet loss including the suggested improvements to calculate the accuracy of the measurements depending on the estimation method used and the active estimation of shaping parameters. Once the measurements are done, the results are exported to the collector for further processing.

Figure 6.3 shows the distribution of the different elements that conform the passive part of the probe module. Basically, the passive probe module captures incoming packets using a raw mmaped socket. Such approach has limited performance specially when monitoring high-speed links. In case that high-speed capture is needed, different capture modules such as the ones described in [GDMR$^+$13] may be easily attached to the probe module to improve capture performance.

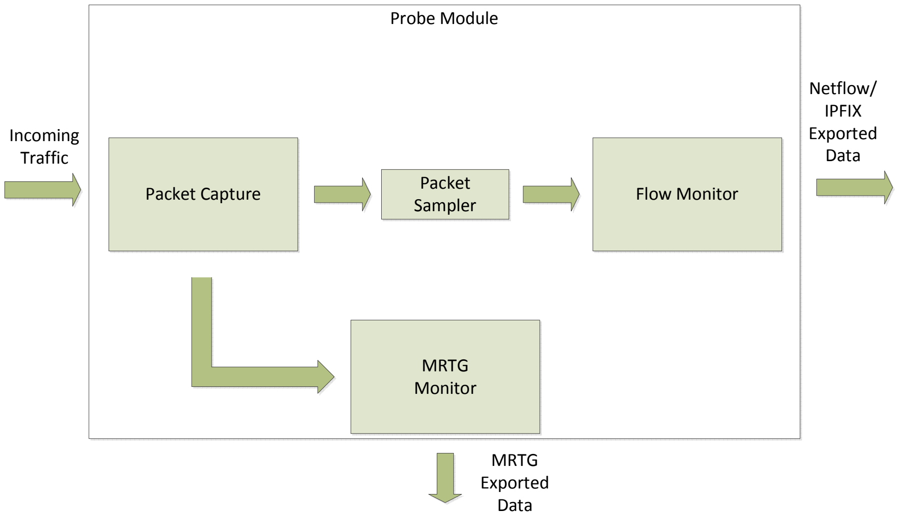


Figure 6.3: Passive Probe Module

In Figure 6.4 the distribution of the different elements that conform the active part of the probe module are shown. Active module is in charge of performing measurements using different types of traffic. To this end, both the Transmission Control Protocol (TCP) measurement and packet train generator submodules are used. The former generates TCP traffic executing a HyperText Transfer Protocol (HTTP) file download from a specific server as stated in Section 3.2.1. The latter sends User Datagram Protocol (UDP) packet trains based on the methods presented in Sections 3.2.2 and 3.2.3. All the measurements parameters such as packet train length, packet size, destination server or HTTP parameters are received from the network and forwarded to the adequate submodule by the active measurement scheduler submodule. In the case of UDP measurement a probe may work as either sender or receiver. If the probe works as receiver the packet train analyzer submodule is used. Such submodule receives UDP packet trains and analyzes them to estimate network parameters such as capacity, OWD or PLR. Once the measurement either packet-train or HTTP download is finished the results are forwarded to the measurement quality quantifier submodule. This part of the active probe module analyzes the results obtained by the previous measurement submodules and applies the techniques specified in Section 3.3 to provide an estimation of the validity of the measurements based on analyzed parameters such as Central Processing Unit (CPU) load, concurrent traffic or available memory. Once the quality of the measurements is quantified, all the information including the quantification and the measurements themselves are sent to the active measurement scheduler submodule. The scheduler sends all the measurement information to the collector for storage and further analysis.

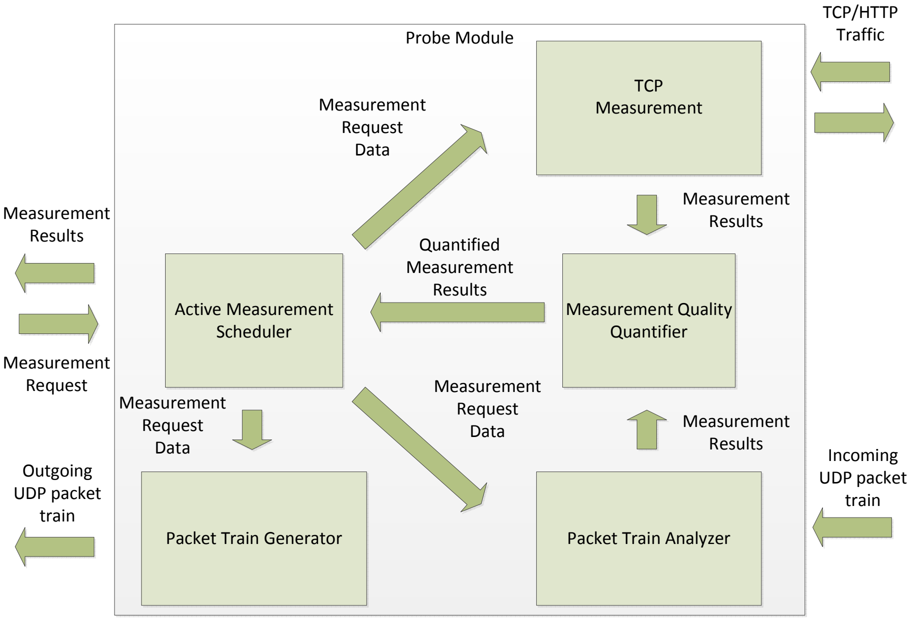


Figure 6.4: Active Probe Module

6.3.2 Collector Module

The collector module is responsible for gathering and storing measurement results and statistics that may be used in different type of analyses. Additionally, the collector module is able to perform some operations over the received data to obtain extra information. Such extra operations are:

- Hash-based packet correlation: using the gathered information from different probes across the network, the collector module correlates and orders temporarily the packet records to analyze the OWD along a network end-to-end path. The used method, Fast Packet Correlation (FPC), is described in Chapter 4.
- Hash-based flow correlation: using the NetFlow/IPFIX information generated by probe modules, a correlation between flows across the network can be performed. Using such correlation the PLR may be estimated using the number of packets of each flow at every hop.
- Time series analysis: using the methods described in Chapters 4 and 5, the collector module performs traffic pollution detection, traffic bursts detection and link characterization using the MRTG-like information (number of bytes, variance of the size of the flows and packets and flows per second) gathered from the probes.

All the data received and generated by the collector module can be stored using different mechanisms. By default, all the information is stored on disk using different files per node and analysis. Such mechanism may be extended to store all the collected and analyzed data in a database to ease the access and organization. Additionally the collector module exchanges information with the frontend module to display and represent the collected and analyzed data. Figure 6.5 shows the collector module architecture.

6.3.3 Frontend Module

The frontend module is in charge of representing and organizing the gathered information in a ordered and easy way. Additionally, the module serves as an interface to control and manage measurement nodes as well measurements. Regarding measurement nodes, the frontend module allows:

- Visualizing measurement network topology.

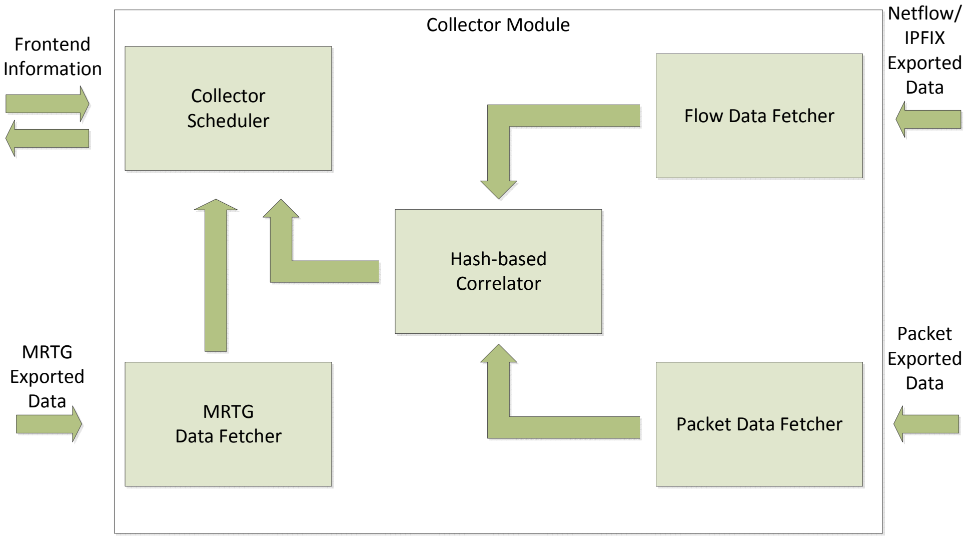


Figure 6.5: Collector Module

- Deploying and undeploying measurement nodes.
- Visualizing measurement node information: name, interfaces and Internet Protocol (IP) and Ethernet addresses.
- Visualizing per node MRTG-like information: number of bytes, packets and concurrent flows.

Regarding measurements, the frontend module allows:

- Deploying and undeploying collector modules in selected nodes.
- Deploying and undeploying probe modules in selected nodes specifying different parameters such as sampling rate, collector IP address or packet fields used in hash calculation. Additionally, Berkeley Packet Filter (BPF) filters can be applied in packet capture.
- Executing and visualizing active measurements between selected nodes defining parameters such as number of packets in train, packet size.
- Tracking packets based on hash correlation to display OWD between nodes or endpoints.

The frontend module is in charge of calculating the time epochs where active measurements must be executed using the algorithm described in 6.2. Active measurements can also be executed in a timely fashion in order to gather data from the QoS of the network at specific moments of the day.

Figures 6.6 and 6.7 show the interface of the PNMF Frontend module. Using this interface, all the deployed nodes can be controlled in a centralized way. Figure 6.7 shows how the topology of the measurement nodes is constructed. As new nodes are added, the topology is reconstructed and displayed in a simple and clean way.

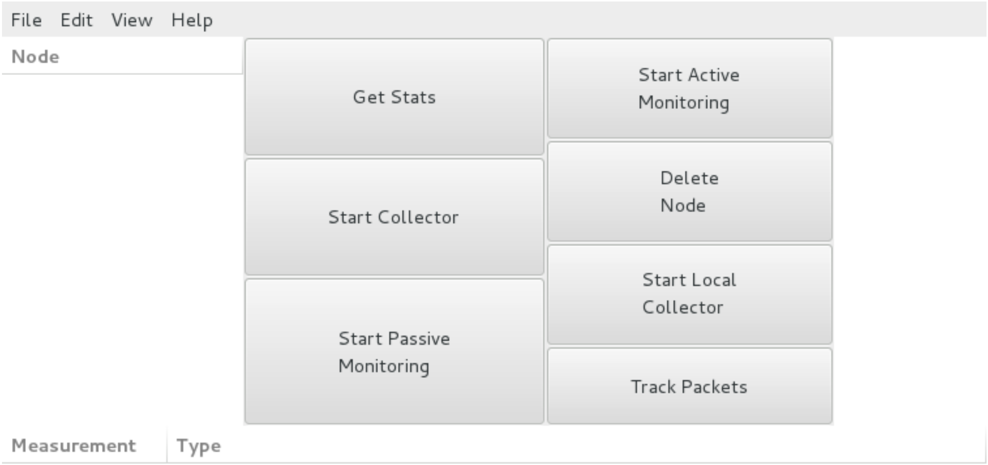


Figure 6.6: PNMF Frontend

Figure 6.8 show the MRTG-like display interface for the byte time series of a given node. This interface allows, also, the visualization of the packet and number of concurrent flows time series.

In Figure 6.9 the OWD time series between two connected nodes is shown. The OWD value is calculated using the hash-based sampling method along with FPC technique for correlation on the collector.

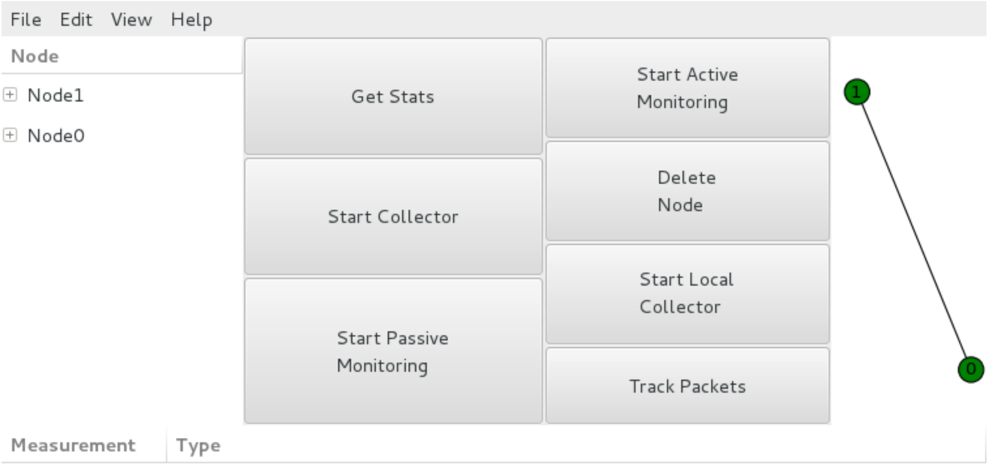
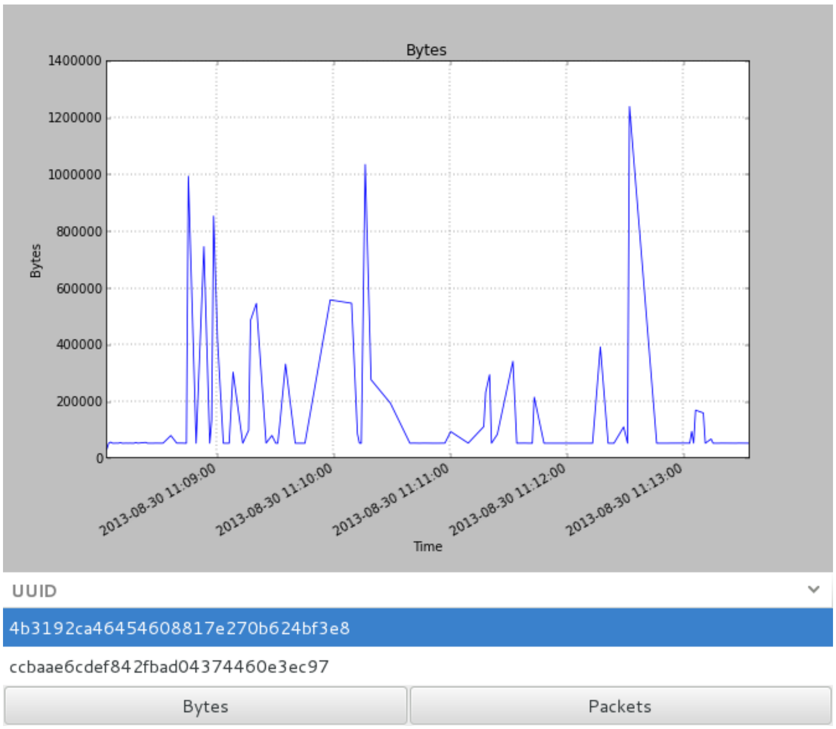


Figure 6.7: PNMF Frontend with nodes added



4b3192ca46454608817e270b624bf3e8

ccbaae6cdef842fbad04374460e3ec97

Figure 6.8: PNMF byte time series

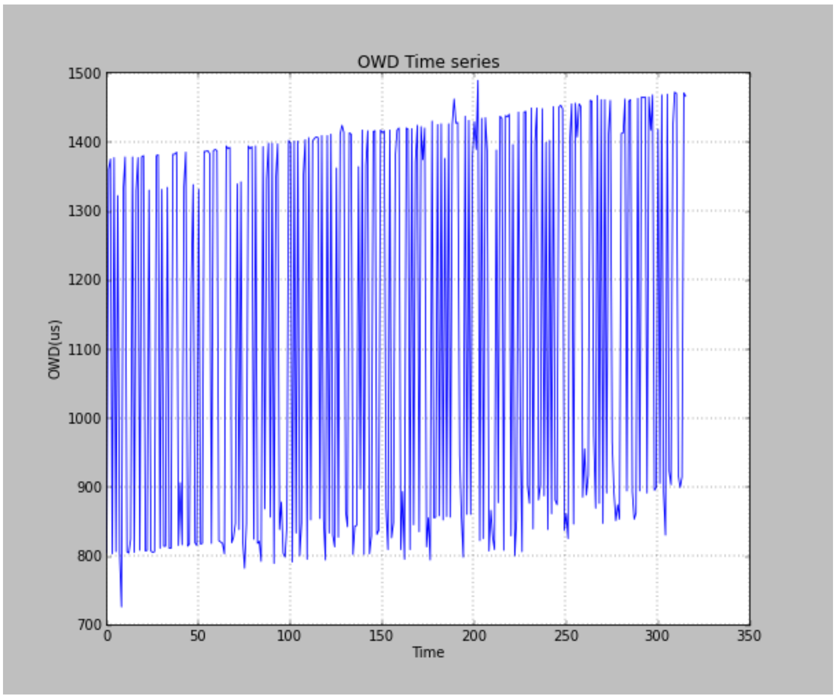


Figure 6.9: PNMF OWD value between two nodes time series

6.4 Conclusions

In this chapter, a brief description about the main features and requirements of a PM system has been presented. First, the triggering between active and passive measurements problem is presented. Next, a simple method based on time series analysis along with JSD calculation is proposed. Such method is able to determine the changes on the reference parameter and launch active measurements when certain threshold is exceeded.

Additionally, in this chapter, PNMF is presented. PNMF is a monitoring framework that allows the integration of active and passive measurements in a simple and centralized system. The system is described along with its architecture focusing on the mechanisms described on the previous chapters and the QoS measurements that is capable of carrying out. This chapter closes the circle of the proactive monitoring describing a successful implementation of a system applicable to heterogeneous networks due to the simple and efficient principles followed for both active and passive monitoring.

Chapter 7

Conclusions

This chapter is devoted to summarize the main results of this Ph.D. thesis. First, in Section 7.1, the main contributions of this thesis are presented. Next, the industrial applications of the work and its deployment in different commercial projects are presented in Section 7.2. Finally, in Section 7.3, future works and research lines for continuing with this work are presented.

7.1 Main Contributions

This thesis has tackled the problem of Proactive Monitoring (PM) on heterogeneous networks. Traditional monitoring works have focused on passive and active network monitoring independently. In this thesis, the idea of combining both types of measurement methodologies into a proactive system has been presented focusing specifically on heterogeneous networks. Such type of networks present great diversity of protocols, technologies and systems with dissimilar requirements and restrictions. In this line, this work has focused on the development of new active and passive monitoring methodologies that are enough simple and powerful to work on heterogeneous environments. Additionally, the problems derived from the combination of active and passive measurements into one proactive system have been addressed in order to provide systems that are able to detect and identify abnormal situations in an early stage and provide Quality of Service (QoS) parameter estimations.

The main conclusions from these contributions are presented at the end of their respective chapters in this thesis. However, the following list shows the key ideas and conclusions derived from this work.

(i) **Active measurement methods are influenced by external parameters such as Central Processing Unit (CPU) load, self-induced interfering traffic and traffic control mechanisms.** Chapter 3 has shown that traditional measurement methods such as file-transfer and packet-pair and derivatives are affected by external parameters leading to an erroneous estimation of QoS parameters. Specifically, this thesis has revealed that CPU load is very influential on the bandwidth measurement. Concretely, file-transfer techniques have been proved to be more affected by higher CPU loads than packet-pair methods. Attending to the self-induced traffic, the study in Chapter 3, has revealed that both file-transfer and packet-pair techniques are influenced by self-induced interfering traffic being the former method the most influenced one. In the case of packet-pair methods, the experiments have revealed that such measurement method is only effected by Constant Bit Rate (CBR) User Datagram Protocol (UDP) cross traffic. Using the gathered information from experiments, a threshold-based measurement rejection method has been proposed. Such method makes use of a worst-case scenario approach to provide an upper bound for the measurement error based on the number of self-induced interfering packets and a multinomial model. Additionally, it has been demonstrated the influence of the traffic rate control mechanisms such as shapers and policers when conducting active measurements. In Chapter 3, the experiments have shown that packet-pair based methods are sensitive to rate control mechanisms producing erroneous measurements which may lead to QoS parameters overestimation. To tackle this issue, an algorithm to detect and characterize the parameters of token-bucket based rate control mechanisms has been presented and compared to other existing solutions achieving the best estimation. The results of the performed studies led to the development of an active measurement tool called QoSPoll[1] freely available.

Finally, the contributions in this chapter have led to the following publications (presented in chronological order):

- J. Aracil, J. Ramos, J.E. López de Vergara, L. de Pedro and S. López, *Appliance for the certified measurement of the bandwidth of a network access and method for the calibration thereof*, International Patent PCT/ES2010/070269, Spain, 28/04/2010.
- J. Aracil, J. Ramos, P.M. Santiago, J.E. López de Vergara, L. de Pedro, S. López, I. González and F.J. Gómez, *Method for estimating the parameters of a control element such as a token bucket*, International Patent PCT/ES2011/070239, Spain, 09/04/2011.
- J. Ramos, P.M. Santiago, J. Aracil and J.E. López de Vergara, *On the effect of concurrent applications in bandwidth measurement speedometers*, Computer Networks (2011), Vol. 55, Issue 6, pp. 1435-1453. JCR Impact Factor 1.2 (Q2).

(ii) **Packet correlation and flow monitoring can be performed at multi-Gb/s rates with simple hardware adaptable to heterogeneous environments**. In Chapter 4, packet correlation and flow monitoring systems are analyzed. To correlate packets and monitor flows, hash functions are normally used. The experiments in Chapter 4 have revealed that BOB hash function may be used to sample packets and generate packet records achieving a 10.4 Mpps in the worst case 10 Gb/s scenario with an empirical collision rate of 4.5%. Taking into account such results, a packet correlation architecture for methods based on trajectory sampling has been proposed. Such architecture, called Fast Packet Correlation (FPC), has been tested with real traffic achieving a 6.8 Gb/s processing rate with an error of 1% and low base memory footprint —128MB. FPC has been used to estimate the One-Way Delay (OWD) across several hops showing a real example of the use of the proposed architecture.

[1] `http://danu.ii.uam.es:8080/QoSPollPro/`

Additionally, a system for flow creation and statistics gathering has been presented. Such system, called FlowProcess, makes uses of efficient memory structures and optimization techniques to produce extended NetFlow/Internet Protocol Flow Information eXport (IPFIX) records and Multi Router Traffic Grapher (MRTG)-like statistics that may be used to estimate QoS parameters. The approach has been tested with real traffic achieving rates up to 6.5 Gb/s. Moreover, due to the need of multi-granularity on heterogeneous networks, the integration of FPC and FlowProcess into a measurement library called Network Measurement Library (NMLib) has been presented. Such library conforms the base of the PM system proposed on Chapter 6.

(iii) **Traffic byte rate and flow size variance are strongly correlated when the number of concurrent flows remains stable.** In Chapter 5, a study of the correlation between the traffic byte rate and the flow size variance has been carried out. Such study has been performed using real traces from operator and bank networks and has revealed a high degree of correlation between the traffic byte rate and the flow size variance when the number of concurrent flows remains stable. Using such idea as starting point, an online algorithm for traffic pollution detection has been developed based on the analysis of traffic byte rate and flow size variance time series. The algorithm has been validated using real traffic traces obtaining promising results with False Positives Rate (FPR) lower than 3% and an average True Positives Rate (TPR) greater than 90%. The presented algorithm fits perfectly in the heterogeneous networks paradigm as makes use of minimal resources in terms of memory and CPU usage.

(iv) **Proactive measurement triggering can be done in an online way using simple statistical methods.** In Chapter 6, the problem of triggering between active and passive measurements is addressed. Detecting short-term changes over a variable or set of variables is interesting as may help in early detection problem and dimensioning. In this light, a method for triggering between measurement methods is presented. Such method is based on the analysis of a reference variable (e.g. traffic byte rate) at fixed-size intervals to detect changes in the distribution of the samples. Such detection is done using Jensen-Shannon Divergence (JSD) as an estimator of the difference between the sample distribution of two adjacent intervals. A simple analysis has revealed the effectiveness of this method to change the measurement method.

(v) **Implementation of a proactive system called Proactive Network Monitoring Framework (PNMF)**. In Chapter 6, the description of the implemented proactive system is presented. Such system puts together all the ideas from Chapters 3, 4 and 5 to conform a reliable and simple system for proactive monitoring focused on QoS parameter estimation and suitable for its deployment in heterogeneous environments.

7.2 Industrial Applications

The results and applications of this thesis are being currently exploited by Naudit HPCN [NAU13]. Naudit is a technology-based startup created as a spin-off from two universities: Universidad Autónoma de Madrid (UAM) [2] and Universidad Pública de Navarra (UPNA), and it is part of its Campus of International Excellence[3]. Its shareholders include both universities as well as Spanish National Research Council (CSIC) by way of Madrid Science Park (Parque Científico de Madrid). Naudit along with Fundación de la Universidad Autónoma de Madrid (FUAM)[4] have carried out several innovation and technology transfer projects. Among Naudit clients, there are public organisms like Spanish Industry Ministry, telecom operators like Movistar, multinational banking groups like BBVA, industrial companies like Airbus or important energy producers. Specifically, the following results of this thesis are directly applied in the industry:

[2] http://www.uam.es/ss/Satellite/es/1242657608103/listadoCategorizado/Spin-offs_de_la_UAM.htm

[3] http://campusexcelencia.uam-csic.es/

[4] http://www.fuam.es

- **QoSPoll:** the methodology and software described in Chapter 3 is currently deployed in a commercial network from the Spanish operator Movistar. The software is in charge of quality monitoring in the Nuevo servicio Ethernet de Banda Ancha (NEBA) service with link speeds up to 100 Mb/s and more than 30 measurement points.

- **QoSInspec:** the file-transfer method and the algorithm for measurement rejection based on the value of external parameters have been implemented in a commercial tool for link quality validation following [Ins08] guidelines. Such tool is in use at the present time by the Spanish Industry Ministry as the official tool for domestic link monitoring and Service-Level Agreement (SLA) validation.

- **FlowProcess:** the flow analysis tool described in Chapter 4 is currently in use for network traffic monitoring and statistics gathering at bank network from BBVA-Bancomer.

7.3 Future Work

The work conducted in this thesis has opened new research lines for future work in the field of the proactive monitoring in heterogeneous environments. In what follows, some future research topics are presented:

- **Active measurement methods and high-speed environments:** Although, active monitoring on high-speed environments has been addressed in this work, 10 Gb/s links are nowadays quite common and the suitability of the classical measurement methods must be analyzed and validated. Moreover, the use of advanced functionalities that present modern Network Interface Cards (NIC)s such as hardware timestamping paves the way for accurate active monitoring in high-speed environments. Additionally, the influence of external parameters in the quality of the measurements in such scenarios must be analyzed in order to determine if packet-pair methods are enough resilient at high-speeds.

- **Flow monitoring optimization:** Despite the promising results shown in Chapter 4 for flow monitoring, new optimization techniques and architectural changes must be introduced in order to reach 10 Gb/s speeds and beyond. Moreover the integration with high-speed capture systems such as the ones proposed in [GDMR+13] must be studied in order to provide efficient online monitoring systems.

- **New hash techniques and packet sampling:** This work has shown the performance and collision rate for a set of selected hash functions. New hash functions are developed every day. Analyzing such functions and their applicability to packet sampling and flow monitoring is a key aspect to improve the efficiency of the presented systems. Moreover, a deep study on the effects of distributed sampling over the estimation of the QoS must be done in order to improve large-scale distributed monitoring systems.

- **Enhanced traffic pollution detection methods:** In this thesis an algorithm for traffic pollution detection has been presented. Such algorithm is only applicable whenever concurrent flow number remains stable over a given time period. Detecting pollution traffic when new legitimate flows are being created is a challenging task that must be addressed in the future. Such work is applicable to several popular fields such as network security and attack detection.

- **Advanced proactive multi-parameter triggering mechanisms:** In Chapter 6, a mono-parameter method for measurement triggering in proactive systems has been presented. Creating a multi-parameter model may increase the efficiency of the proactive systems by launching active measurements in a smarter way. Additionally, the use of other statistical tools to determine the triggering instants and their performance compared with the solution proposed in this work is a future work line.

- **Performance and efficiency of proactive monitoring systems on large-scale networks:** the proposed proactive system in Chapter 6 has been tested in a limited environment with a low number of nodes. Deploying the system in large-scale networks with hundreds of nodes is a paramount importance task in order to analyze the performance and verify the applicability of the proposed system in such environments.

Conclusiones

Este capítulo está dedicado a resumir los resultados e ideas principales presentados en esta tesis doctoral. A continuación se presentan las contribuciones principales del trabajo así como las aplicaciones industriales del mismo. Para terminar, se presenta una colección de ideas para trabajos futuros derivados de las líneas de investigación planteadas en este trabajo.

Contribuciones Principales

Esta tesis ha tratado el problema de la monitorización proactiva en redes heterogéneas. Tradicionalmente, los trabajos de monitorización se han centrado en medidas de red activas o pasivas de manera independiente. En esta tesis, la idea de la combinación de ambos tipos de medidas en un sistema proactivo de monitorización ha sido presentada haciendo especial hincapié en su aplicabilidad a las redes heterogéneas. Este tipo de redes presentan gran diversidad de protocolos, tecnologías y sistemas con requisitos y restricciones muy dispares. En esta línea, este trabajo se ha centrado en el desarrollo de nuevas metodologías de monitorización activas y pasivas que sean suficientemente simples y potentes para funcionar en entornos heterogéneos. Adicionalmente, los problemas derivados de combinar medidas activas y pasivas han sido tratados para proporcionar sistemas que sean capaces de detectar situaciones anómalas en etapas tempranas y proporcionar estimaciones de los parámetros de calidad de servicio.

Las conclusiones principales de estas contribuciones se han presentado al final de sus correspondientes capítulos, sin embargo, a continuación se resumen las ideas más importantes de esta tesis.

(i) **Los métodos de medida activos se ven influidos por parámetros externos tales como la carga de CPU, el tráfico auto-inducido o los mecanismos de control de tráfico.** En el Capítulo 3 se ha mostrado que los métodos de medida tradicionales tales como la descarga de fichero o los métodos de pares de paquetes y sus derivados se ven afectados por la influencia de parámetros externos produciendo estimaciones erróneas de los parámetros de calidad de servicio. Específicamente, esta tesis ha mostrado que la carga de CPU es muy influyente en la medida del ancho de banda. En concreto, los métodos de descarga de fichero se han mostrado más afectados que los métodos basados en pares de paquetes. Atendiendo al tráfico auto-inducido, el estudio en el Capítulo 3 ha revelado que tanto los métodos de descarga de fichero como los métodos de pares de paquetes se ven afectados por el tráfico interferente siendo los primeros más sensibles. En el caso de los métodos de pares de paquetes, los experimentos han mostrado que solamente se ven afectados por tráfico cruzado UDP CBR. Usando la información obtenida de estos experimentos, se ha desarrollado un método de descarte de medidas basado en umbrales. Este método hace uso de una aproximación de caso peor para proporcionar un límite superior para el error de la medida basándose en el número de paquetes de tráfico auto-inducido y un modelo multinomial.

Adicionalmente, se ha demostrado la influencia de los mecanismos de control de tasa de tráfico como shapers y policers sobre los métodos activos. En el Capítulo 3, los experimentos han mostrado que los métodos de pares de paquetes son sensibles a los mecanismos de control de tráfico y producen medidas erróneas que llevan a la sobreestimación de algunos parámetros de calidad de servicio. Para solucionar este problema, se ha propuesto un algoritmo para detectar y caracterizar los parámetros de los mecanismos de control basados en Token Bucket y se ha comparado con otras soluciones previas obteniendo los mejores resultados. Los resultados de los estudios realizados han llevado a la creación de una herramienta gratuita de medida activa llamada QoSPoll[5].

Finalmente, las contribuciones en esta área han producido las siguientes publicaciones (presentadas en orden cronológico):

- Javier Aracil Rico, Javier Ramos de Santiago , Jorge E. López de Vergara Méndez, Luis de Pedro Sánchez, Sergio López Buedo, *Aparato para la medición certificada del ancho de banda de un acceso de red y método de calibración del mismo (APPLIANCE FOR THE CERTIFIED MEASUREMENT OF THE BANDWIDTH OF A NETWORK ACCESS AND METHOD FOR THE CALIBRATION THEREOF)*, Patente Internacional PCT/ES2010/070269, España, 28/04/2010.
- Javier Aracil Rico, Javier Ramos de Santiago, Pedro M. Santiago del Río, Jorge E. López de Vergara Méndez, Luis de Pedro Sánchez, Sergio López Buedo, Iván González Martínez, Francisco Javier Gómez Arribas, *Método para estimar los parámetros de un elemento de control de tipo Tocken-Bucket (METHOD FOR ESTIMATING THE PARAMETERS OF A CONTROL ELEMENT SUCH AS A TOKEN BUCKET)*, Patente Internacional PCT-/ES2011/070239, España, 09/04/2011.
- Javier Ramos, Pedro M. Santiago, Javier Aracil, Jorge E. López de Vergara , *On the effect of concurrent applications in bandwidth measurement speedometers*, Computer Networks (2011), Vol. 55, Issue 6, pp. 1435-1453. Indice de impacto JCR 1.2 (Q2).

(ii) **Se puede realizar correlación de paquetes y monitorización de flujos a tasas multi-Gb/s con hardware simple adaptable a entornos heterogéneos.** En el Capítulo 4, la correlación de paquetes y los sistemas de monitorización de flujos han sido analizados. Para correlar paquetes y monitorizar flujos, normalmente se hace uso de funciones hash. En los experimentos de este capítulo, se ha mostrado que la función hash BOB puede ser usada para muestrear paquetes y generar

[5] `http://danu.ii.uam.es:8080/QoSPollPro/`

registros de paquetes a una tasa de 10.4 Mpps en el peor escenario de 10 Gb/s con una tasa de colisión de 4.5%. Tomando en cuenta estos resultados, se ha propuesto una arquitectura de correlación de paquetes para los métodos de *t*rajectory sampling. Esta arquitectura, llamada FPC, ha sido evaluada con tráfico real obteniendo una tasa de proceso de aproximadamente 6.8 Gb/s con un error del 1% y una huella base de memoria mínima — 128 MB. Como ejemplo práctico, se ha usado FPC para realizar el cálculo del retardo en un sentido de cada paquete a través de varios saltos.

Adicionalmente, se ha presentado un sistema de creación de flujos y recolección de estadísticas. Este sistema, llamado FlowProcess, hace uso de estructuras de memoria eficientes y técnicas de optimización para producir registros extendidos NetFlow/IPFIX y estadísticas similares a MRTG que pueden ser utilizadas para estimar parámetros de QoS. La propuesta se ha evaluado con tráfico real alcanzando tasas de proceso de 6.5 Gb/s. Además, debido a la necesidad de medidas con múltiple granularidad en redes heterogéneas, se ha realizado la integración FPC y FlowProcess en una librería llamada NMLib. Esta librería es la base para el sistema de monitorización proactiva presentado en el Capítulo 6.

(iii) **La tasa de tráfico en bytes y la varianza del tamaño de los flujos están fuertemente correlados cuando el número de flujos concurrentes se mantiene estable.** En el Capítulo 5 se ha realizado un estudio de la correlación entre la tasa de tráfico en bytes y la varianza del tamaño de los flujos. Este estudio se ha realizado con tráfico real de un operador y una red bancaria y ha revelado un alto grado de correlación entre las dos magnitudes cuando el número de flujos se mantiene estable. Usando esta idea como punto de partida, se ha desarrollado un algoritmo *online* que permite detectar tráfico contaminante tomando en cuenta la tasa de tráfico en bytes y la varianza del tamaño de los flujos. El algoritmo ha sido validado usando trazas con tráfico real obteniendo una tasa de falsos positivos menor al 3% y una tasa de verdaderos positivos por encima del 90%. El algoritmo propuesto se adapta perfectamente al entorno heterogéneo ya que hace un uso mínimo de memoria y CPU.

(iv) **La selección entre los diferentes tipos de medidas en sistemas proactivos puede ser realizada de manera *online* utilizando métodos estadísticos sencillos.** En el Capítulo 6 se aborda el problema del cambio entre medidas activas y pasivas. Detectar cambios a corto plazo en los valores de una variable o un conjunto de variables es interesante ya que puede ayudar a la detección temprana de problemas y al dimensionado. En este sentido, se ha propuesto un método para cambio entre medidas activas y pasivas basado en el análisis de una variable de referencia, por ejemplo la tasa de tráfico expresada en bytes. Este método hace uso de intervalos de tamaño fijo para detectar cambios en las distribuciones de los valores de la variable de referencia. Para ello, se utiliza la divergencia de Jensen-Shannon como estimador de la diferencia en las distribuciones de los valores de dos intervalos adyacentes. Un análisis simple ha revelado la efectividad de este método para realizar cambios en el método de medida usado.

(v) **Implementación de un sistema proactivo llamado PNMF.** En el Capítulo 6 se ha presentado la descripción del sistema proactivo implementado. Este sistema reúne todas las ideas de los Capítulos 3, 4 y 5 para conformar un sistema de monitorización proactiva simple y fiable centrado en proporcionar estimaciones de parámetros de calidad de servicio para su despliegue en redes heterogéneas.

Aplicaciones Industriales

Los resultados y aplicaciones de esta tesis están siendo actualmente explotados por Naudit HPCN [NAU13]. Naudit es una empresa de base tecnológica creada por una spin-off de dos universidades: la Universidad Autónoma de Madrid (UAM)[6] y la Universidad Pública de Navarra (UPNA), y forma parte

[6] `http://www.uam.es/ss/Satellite/es/1242657608103/listadoCategorizado/Spin-offs_de_la_UAM.htm`

de su Campus de Excelencia Internacional[7]. En su accionariado participan, además de ambas universidades, el Consejo Superior de Investigaciones Científicas (CSIC), a través del parque científico de Madrid.

Naudit junto con la Fundación de la Universidad Autónoma de Madrid (FUAM)[8] han llevado a cabo varios proyectos de innovación y transferencia tecnológica. Entre los clientes de Naudit se encuentran organismos públicos como el Ministerio de Industria del Gobierno de España, operadoras de telecomunicaciones como Movistar, grupos bancarios multinacionales como el BBVA, compañías del sector industrial como Airbus o importantes grupos energéticos. Concretamente, los siguientes resultados de la tesis son aplicados directamente en la industria:

- **QoSPoll:** la metodología y el software descrito en el Capítulo 3 está actualmente desplegado en una red comercial del operador español Movistar. El software se encarga de la monitorización de la calidad en el servicio NEBA el cual cuenta con enlaces con velocidades de hasta 100 Mb/s y más de 30 puntos de medida.

- **QoSInspec:** el método de descarga de fichero y el algoritmo de rechazo de medidas contaminadas por el impacto de factores externos han sido implementados en una solución comercial para la validación de la calidad de los enlaces siguiendo las directrices descritas en [Ins08]. Esta herramienta está actualmente en uso por el Ministerio de Industria de España y es la herramienta oficial para monitorización de enlaces domésticos y validación de acuerdos de nivel de servicio.

- **FlowProcess:** la herramienta de análisis de flujos presentada en el Capítulo 4 se usa en la actualidad en la red bancaria de BBVA-Bancomer para monitorizar el tráfico que circula y obtener estadísticas.

[7] `http://campusexcelencia.uam-csic.es/`

[8] `http://www.fuam.es`

Trabajo Futuro

El trabajo realizado en esta tesis ha abierto nuevas líneas de investigación para trabajos futuros en el campo de la monitorización proactiva en entornos heterogéneos. A continuación se presentan algunos temas de investigación futuros:

- **Métodos de medida activos y entornos de alta velocidad:** A pesar de que la monitorización en entornos de alta velocidad ha sido abordada por este trabajo, los enlaces de 10 Gb/s son muy comunes hoy en día y se deben analizar y validar los métodos de medida clásicos para ver si son aplicables en estos escenarios. Además, el uso de las características avanzadas que presentan las tarjetas de red modernas como el marcado temporal por hardware abre el camino a nuevos métodos de monitorización activa en entornos de alta velocidad con gran precisión. Adicionalmente, el impacto de parámetros externos en la calidad de las medidas en estos escenarios debe ser analizado para determinar si los métodos de pares de paquetes son suficientemente inmunes a altas tasas.

- **Optimización de la monitorización de flujos:** A pesar de los prometedores resultados para la monitorización de flujos mostrados en el Capítulo 4, se deben aplicar nuevas técnicas de optimización y cambios arquitecturales para alcanzar velocidades de 10 Gb/s y superiores. Además, se debe estudiar la integración con sistemas de captura de alta velocidad como los mostrados en [GDMR$^+$13] para proporcionar sistemas efectivos de monitorización *online*.

- **Nuevas técnicas hash y muestreo de paquetes:** En este trabajo se ha mostrado el rendimiento y la tasa de colisión de un conjunto de funciones hash seleccionadas. Sin embargo, cada día se desarrollan nuevas funciones hash. Analizar estas funciones y su aplicabilidad al muestreo de paquetes y la monitorización de flujos es un aspecto clave para mejorar la eficiencia de los sistemas presentados. Además se debe realizar un estudio profundo del impacto del muestreo distribuido en la estimación de parámetros de calidad de servicio para mejorar los sistemas de monitorización distribuida a gran escala.

- **Métodos de detección de contaminación de tráfico mejorados:** En esta tesis se ha propuesto un algoritmo para la detección de contaminación de tráfico. Este algoritmo solo puede aplicarse cuando el número de flujos concurrentes se mantiene estable. Detectar la contaminación de tráfico cuando flujos legítimos están siendo creados es un reto que debe ser abordado en el futuro. Este trabajo es aplicable a varios campos muy populares como la seguridad de red o la detección de ataques.

- **Mecanismos avanzados multi-parámetro para el cambio entre medidas en entornos proactivos:** En el Capítulo 6, se ha presentado un método monoparamétrico para el cambio entre medidas en sistemas proactivos. Crear un sistema multi-paramétrico puede incrementar la eficiencia de los sistemas proactivos manejando mejor los momentos en los que se ejecutan las medidas activas. Adicionalmente, el uso de otras herramientas estadísticas para determinar los instantes de cambios entre tipos de medida así como su rendimiento en comparación con el método propuesto conforman una línea de trabajo futuro.

- **Rendimiento y eficiencia de los sistemas de monitorización proactiva en redes a gran escala:** el sistema proactivo propuesto en el Capítulo 6 ha sido probado en un entorno limitado con un número pequeño de nodos. Desplegar el sistema en una red a gran escala con cientos de nodos es una tarea de gran importancia para analizar el rendimiento y verificar la aplicabilidad de la solución propuesta en este tipo de entornos.

References

[AAMD06] D. Antoniades, A. Athanatos, D. Papadogiannakis, E.P. Markatos, and C. Dovrolis, *Available bandwidth measurement as simple as running wget*, Proceedings of the 7th International Conference on Passive and Active Network Measurement (Adelaide, Australia), PAM'06, March 2006. 40

[AKZ99a] G. Almes, S. Kalidindi, and M. Zekauskas, *RFC 2679: A One-way Delay Metric for IPPM*, 1999. 11, 13

[AKZ99b] ———, *RFC 2680: A One-way Packet Loss Metric for IPPM*, 1999. 13, 14

[AKZ99c] ———, *RFC 2681: A Round-trip Delay Metric for IPPM*, 1999. 12, 88

[AP08] M. Allman and V. Paxson, *A reactive measurement framework*, Passive and Active Network Measurement, Lecture Notes in Computer Science, vol. 4979, 2008, pp. 92–101. 2

[BKM$^+$09] A. Bulut, N. Koudas, A. Meka, A.K. Singh, and D. Srivastava, *Optimization techniques for reactive network monitoring*, IEEE Transactions on Knowledge and Data Engineering **21** (2009), no. 9, 1343 –1357. 2

[BKPR02] P. Barford, J. Kline, D. Plonka, and A. Ron, *A signal analysis of network traffic anomalies*, Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment (Marseille, France), IMW '02, November 2002, pp. 71–82. 112

[BMSDM12] S. Basso, M. Meo, A. Servetti, and J.C. De Martin, *Estimating packet loss rate in the access through application-level measurements*, Proceedings of the 2nd ACM SIGCOMM Workshop on Measurements Up the Stack (Helsinki, Finland), W-MUST '12, 2012, pp. 7–12. 40

[BV02] P. Benko and A. Veres, *A passive method for estimating end-to-end TCP packet loss*, Proceedings of the IEEE 2002 Global Telecommunications Conference ((Taipei, Taiwan)), GLOBECOM '02, vol. 3, November 2002, pp. 2609 – 2613. 14, 90

[CFEK06] K. Cho, K. Fukuda, H. Esaki, and A. Kato, *The impact and implications of the growth in residential user-to-user traffic*, ACM SIGCOMM Computer Communication Review **36** (2006), no. 4, 207–218. 48

[CFGS11] B. Constantine, G. Forget, Ruediger Geib, and R. Schrage, *RFC 6349: Framework for TCP Throughput Testing*, 2011. 12

[Cla04] B. Claise, *RFC 3954: Cisco Systems NetFlow Services Export Version 9*, 2004. 26

[Cla08] ———, *RFC 5101: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*, 2008. 27

[Cor11] C.J. Corrado, *The exact distribution of the maximum, minimum and the range of multinomial/dirichlet and multivariate hypergeometric frequencies*, Statistics and Computing **21** (2011), no. 3, 349–359. 173

[DA03] L. Deri and Netikos S. P. A., *Passively monitoring networks at gigabit speeds using commodity hardware and open source software*, Proceedings of the 4th International Conference on Passive and Active Network Measurement (La Jolla, California, USA), PAM'03, April 2003. 3

[DC02] C. Demichelis and P. Chimento, *RFC 3393: IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)*, 2002. 12

[DG00] N. G. Duffield and M. Grossglauser, *Trajectory sampling for direct traffic observation*, ACM SIGCOMM Computer Communication Review **30** (2000), no. 4, 271–282. 90

[dRCGDA13] P.M.S. del Rıo, D. Corral, JL Garcıa-Dorado, and J. Aracil, *On the impact of packet sampling on Skype traffic classification*, Poceedings of the IFIP/IEEE International Symposium on Integrated Network Management, IM '13, May 2013. 27

[DRM01] C. Dovrolis, P. Ramanathan, and D. Moore, *What do packet dispersion techniques measure?*, Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (Anchorage,Alaska, USA), INFOCOM '01, vol. 2, April 2001, pp. 905–914. 42, 44

[DRM04] C. Dovrolis, P. Ramanathan, and D. Moore, *Packet-dispersion techniques and a capacity-estimation methodology*, IEEE/ACM Transactions on Networking **12** (2004), 963–977. 42, 45

[Duf04] N. Duffield, *Sampling for passive Internet measurement: A review*, Statistical Science **19** (2004), 472–498. 27

[EGE02] J. Elson, L. Girod, and D. Estrin, *Fine-grained network time synchronization using reference broadcasts*, ACM SIGOPS Operating Systems Review **36** (2002), no. SI, 147–163. 34

[ENUK06] T. En-Najjary and G. Urvoy-Keller, *PPrate: A passive capacity estimation tool*, Proceedings of the 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (Vancouver,Canada), E2EMON '06, April 2006, pp. 82 – 89. 89

[FAM01] K. Fujimoto, S. Ata, and M. Murata, *Statistical analysis of packet delays in the Internet and its application to playout control for streaming applications*, IEICE Transaction on Communications **E84** (2001), no. 6, 1504–1512. 48, 67

[FD10] F. Fusco and L. Deri, *High speed network traffic analysis with commodity multi-core systems*, Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference (Melbourne, Australia), IMC '10, November 2010, pp. 218–224. 86

[FDL+01] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P.e Owezarski, D. Papagiannaki, and F. Tobagi, *Design and deployment of a passive monitoring infrastructure*, Evolutionary Trends of the Internet, Lecture Notes in Computer Science, vol. 2170, 2001, pp. 556–575. 34

[FUK+09] A. Friedl, S. Ubik, A. Kapravelos, M. Polychronakis, and E.P. Markatos, *Realistic passive packet loss measurement for high-speed networks*, Traffic Monitoring and Analysis, Lecture Notes in Computer Science, vol. 5537, 2009, pp. 1–7. 14, 15, 89

[GDMR+13] J.L. García-Dorado, F. Mata, J. Ramos, P.M. Santiago del Río, V. Moreno, and J. Aracil, *High-performance network traffic processing systems using commodity hardware*, Data Traffic Monitoring and Analysis, Lecture Notes in Computer Science, vol. 7754, 2013, pp. 3–27. 3, 86, 131, 145, 153

[GHH+09] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, *Flow processing and the rise of commodity network hardware*, ACM SIGCOMM Computer Communication Review **39** (2009), no. 2, 20–26. 3

[HAN02] T.J. Hacker, B.D. Athey, and B. Noble, *The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network*, Proceedings of the 16th IEEE International

Parallel and Distributed Processing Symposium (Washington, DC, USA), IPDPS '02, 2002, pp. 314–. 41

[HG99a] J. Heinanen and R. Guerin, *RFC 2697: A Single Rate Three Color Marker*, 1999. 19

[HG99b] ———, *RFC 2698: Two Rate Three Color Marker*, 1999. 20

[HG03] K.S.J. Hielscher and R. German, *A low-cost infrastructure for high precision high volume performance measurements of web clusters*, Computer Performance Evaluation. Modelling Techniques and Tools, Lecture Notes in Computer Science, vol. 2794, 2003, pp. 11–28. 35

[HMn07] A. Hernandez and E. Magaña, *One-way delay measurement and characterization*, Proceedings of the 3rd IEEE International Conference on Networking and Services (Athens,Greece), ICNS '07, June 2007, p. 114. 11

[HSZ08] C. Henke, C. Schmoll, and T. Zseby, *Empirical evaluation of hash functions for multipoint measurements*, ACM SIGCOMM Computer Communication Review **38** (2008), no. 3, 39–50. 93, 94

[IDVFE10] S. Ickin, K. De Vogeleer, M. Fiedler, and D. Erman, *The effects of packet delay variation on the perceptual quality of video*, Proceedings of the 35th IEEE Conference on Local Computer Networks, LNC '10, October 2010, pp. 663 –668. 13

[Ins08] European Telecommunications Standards Institute, *Speech Processing, Transmission and Quality Aspects (STQ);User related QoS parameter definitions and measurements;Part 4: Internet access*, 2008. 39, 40, 65, 81, 144, 152

[Int12] Intel, *82599 10 Gbe controller datasheet*, `http://www.intel.com/content/www/us/en/ethernet-controllers/82599-10-gbe-controller-datasheet.html`, 2012. 95, 97

[IS08] IEEE Instrumentation and Measurement Society, *IEEE standard 1588-2008: Standard for a precision clock synchronization protocol for networked measurement and control systems*, 2008. 29, 31, 33

[ITF04] Y. Ito, S. Tasaka, and Y. Fukuta, *Psychometric analysis of the effect of end-to-end delay on user-level QoS in live audio-video transmission*, Proceedings of the 2004 IEEE International Conference on Communications (Paris, France), ICC '04, vol. 4, June 2004, pp. 2214–2220. 48, 67

[Jac88] V. Jacobson, *Congestion avoidance and control*, ACM SIGCOMM Computer Communication Review **18** (1988), no. 4, 314–329. 42

[JD02] H. Jiang and C. Dovrolis, *Passive estimation of TCP round-trip times*, ACM SIGCOMM Computer Commuication Review **32** (2002), no. 3, 75–88. 89

[Joh03] A. Johnsson, *On the comparison of packet-pair and packet-train measurements*, Proceedings of the 2003 Swedish National Computer Networking Workshop (Arlandastad,Sweden), SNCNW '03, 2003. 44

[Kes91] Srinivasan Keshav, *A control-theoretic approach to flow control*, ACM SIGCOMM Computer Communication Review **21** (1991), no. 4, 3–15. 42

[LAM$^+$11] V. Lopez, J.L. Anamuro, V. Moreno, J.E. Lopez De Vergara, J. Aracil, C. Garcia, J.P. Fernandez-Palacios, and M. Izal, *Implementation of multi-layer techniques using FEDERICA, PASITO and OneLab network infrastructures*, Proceedings of the 17th IEEE International Conference on Networks (Singapore,Singapore), ICON '11, December 2011, pp. 89 –94. 34

[LDK10] Myungjin L., N. Duffield, and R. Kompella, *Two samples are enough: Opportunistic flow-level latency estimation using*

NetFlow, Proceedings of the 29th Annual Joint Conference of the IEEE Computer and Communications Societies (San Diego,CA,USA), INFOCOM '10, March 2010, pp. 1 –9. 88

[Lin91] J. Lin, *Divergence measures based on the shannon entropy*, IEEE Transactions on Information Theory **37** (1991), 145–151. 127

[LLdVBF04] D. Lpez, J.E. Lpez de Vergara, L. Bellido, and D. Fernndez, *Monitoring an academic network with netflow*, Proceedings of the 10th EUNICE Open European Summer School: Advances in fixed and mobile networks (Tampere, Finland), EUNICE '04, June 2004, pp. 63–70. 88

[LSK10] H. Liu and M. Sik Kim, *Real-time detection of stealthy DDoS attacks using time-series decomposition*, Proceedings of the 2010 IEEE International Conference on Communications, ICC '10, May 2010, pp. 1–6. 112

[MA01] M. Mathis and M. Allman, *RFC 3148: A Framework for Defining Empirical Bulk Transfer Capacity Metrics*, 2001. 11, 88

[MBG00] B. Melander, M. Bjorkman, and P. Gunningberg, *A new end-to-end probing and analysis method for estimating bandwidth bottlenecks*, Proceedings of the 2000 IEEE Global Telecommunications Conference (San Francisco, CA, USA), GLOBECOM '00, vol. 1, November 2000, pp. 415–420. 44

[MBG02] B. Melander, M. Björkman, and P. Gunningberg, *Regression-based available bandwidth measurements*, Proceedings of the 2002 SCS/IEEE Symposium on Performance and Evaluation of Computer and Telecommunications Systems (San Diego, CA, USA), SPECTS '02, July 2002. 44

[Mic13] Microsoft, *Receive Side Scaling*, `http://msdn.microsoft.com/en-us/library/windows/hardware/ff567236(v=vs.85).aspx`, 2013. 95

[Mil85] D.L. Mills, *RFC 958: Network Time Protocol (NTP)*, 1985, Obsoleted by RFCs 1059, 1119, 1305. 29

[Min99] N. Minar, *A survey of the NTP network*, `http://alumni.media.mit.edu/~nelson/research/ntp-survey99/html/`, December 1999. 30

[MMBK10] D. Mills, J. Martin, J. Burbank, and W. Kasch, *RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification*, 2010. 29

[MMI$^+$05] D. Morato, E. Magana, M. Izal, J. Aracil, F. Naranjo, F. Astiz, U. Alonso, I. Csabai, P. Haga, G. Simon, J. Steger, and G. Vattay, *The European Traffic Observatory Measurement Infrastructure (ETOMIC): a testbed for universal active and passive measurements*, Proceedings of the 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Trento,Italy), TRIDENTCOM '05, February 2005, pp. 283 – 289. 34

[MND05] M. Molina, S. Niccolini, and N.G. Duffield, *A comparative experimental study of hash functions applied to packet sampling*, Proceedings of the 19th IAC International Teletraffic Congress (Beijing, China), ITC '05, August 2005, pp. 1–10. 93, 94

[NAU13] *Naudit High Performance Computing and Networking,*, 2013, `http://www.naudit.es`. 143, 151

[Ols05] R. Olsson, *Pktgen the Linux packet generator*, Proceedings of the 7th Linux symposium (Ottawa, Canada), vol. 2, July 2005, pp. 11–25. 43

[OR98] T. Oetiker and D. Rand, *MRTG: The Multi Router Traffic Grapher*, Proceedings of the 12th USENIX Conference on System Administration (Boston, MA, USA), LISA '98, December 1998, pp. 141–148. 28

[OSSSP12] R. O. Schmidt, A. Sperotto, R. Sadre, and A. Pras, *Towards bandwidth estimation using flow-level measurements*, Dependable Networks and Services, Lecture Notes in Computer Science, vol. 7279, 2012, pp. 127–138. 87, 105

[Pal09] G.K. Palshikar, *Simple Algorithms for Peak Detection in Time-Series*, Proceedings of the 1st IIMA International Conference on Advanced Data Analysis, Business Analytics and Intelligence (Gujarat, India), June 2009. 118

[Pax96] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics PhD. Thesis*, `http://www.eecs.berkeley.edu/Pubs/TechRpts/1997/CSD-97-945.pdf`, 1996. 42

[PG93] A.K. Parekh and R.G. Gallager, *A generalized processor sharing approach to flow control in integrated services networks: the single-node case*, IEEE/ACM Transactions on Networking **1** (1993), no. 3, 344–357. 23

[PJD04] R. Prasad, M. Jain, and C. Dovrolis, *Effects of interrupt coalescence on network measurements*, Proceedings of the 5th International Conference on Passive and Active Network Measurement (Antibes Juan-les-Pins, France), PAM '04, April 2004, pp. 247–256. 58, 70, 78

[PKP^{+}06] A. Papadogiannakis, A. Kapravelos, M. Polychronakis, E. P. Markatos, and A. Ciuffoletti, *Passive end-to-end packet loss estimation for grid traffic monitoring*, Proceedings of the CoreGRID Integration Workshop (Krakow, Poland), CGIW '06, 2006. 14

[PRTV10] A. Pescape, D. Rossi, D. Tammaro, and S. Valenti, *On the impact of sampling on traffic monitoring and analysis*, Proceedings of the 22nd IAC International Teletraffic Congress Teletraffic Congress (Amsterdam, The Netherlands), ITC '10, 2010, pp. 1 –8. 27

[QBC$^+$08] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer, *RFC 5102: Information Model for IP Flow Information Export*, 2008. 27

[QZCZ04] J. Quittek, T. Zseby, B. Claise, and S. Zander, *RFC 3917: Requirements for IP Flow Information Export (IPFIX)*, 2004. 27

[RSDC11] F. Ricciato, F. Strohmeier, P. Dorfinger, and A. Coluccia, *One-way loss measurements from IPFIX records*, Proceedings of the 1st IEEE International Workshop on Measurements and Networking (Anacapri, Italy), M&N '11, October 2011, pp. 158 –163. 88, 105

[SBDR10] J. Sommers, P. Barford, N. Duffield, and A. Ron, *Multiobjective monitoring for SLA compliance*, IEEE/ACM Transactions on Networking **18** (2010), no. 2, 652–665. 34

[SDTG10] F. Silveira, C. Diot, N. Taft, and R. Govindan, *ASTUTE: detecting a different class of traffic anomalies*, ACM SIGCOMM Computer Communication Review **40** (2010), no. 4, 267–278. 112

[SHSZ10] T. Santos, C. Henke, C. Schmoll, and T. Zseby, *Multi-hop packet tracking for experimental facilities*, ACM SIGCOMM Computer Communication Review **40** (2010), no. 4, 447–448. 90

[SMRD06] A. Seuret, F. Michaut, J.P. Richard, and T. Divoux, *Networked control using GPS synchronization*, Proceedings of the

2006 American Control Conference (Minneapolis, MN, USA), ACC '06, June 2006, p. 6 pp. 34

[SRW$^+$08] V. Sekar, M.K. Reiter, W. Willinger, H. Zhang, R.R. Kompella, and D.G. Andersen, *CSAMP: a system for network-wide flow monitoring*, Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (Berkeley, CA, USA), NSDI'08, 2008, pp. 233–246. 91

[TB08] B. Trammell and E. Boschi, *RFC 5103: Bidirectional Flow Export Using IP Flow Information Export (IPFIX)*, 2008. 27

[TMH11] G. Thatte, U. Mitra, and J. Heidemann, *Parametric methods for anomaly detection in aggregate traffic*, IEEE/ACM Transactions on Networking **19** (2011), no. 2, 512–525. 112

[WCA] C. Walsworth, k.c. Claffy, and D. Andersen, *The CAIDA UCSD Anonymized Internet Traces 2009 - 17 December*, `http://www.caida.org/data/passive/passive_2009_dataset.xml`. 94

[WN10] G. Wang and T.S.E. Ng, *The Impact of Virtualization on Network Performance of Amazon EC2 Data Center*, Proceedings of the 29th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '10, 2010, pp. 1163–1171. 64

[XCF12] Q. Xu, D. Cheng, and Y. Fu, *Traffic feature distribution analysis based on exponentially weighted moving average*, Proceedings of the 2nd IEEE International Conference on Computer Science and Automation Engineering (Zhangjiajie, China), CSAE '12, vol. 1, May 2012, pp. 535–539. 112

[ZBBC09] T. Zseby, E. Boschi, N. Brownlee, and B. Claise, *RFC 5472: IP Flow Information Export (IPFIX) Applicability*, 2009. 88

[ZH07] W. Zhang and J. He, *Modeling end-to-end delay using Pareto distribution*, Proceedings of the 2nd IARIA International Conference on Internet Monitoring and Protection (Silicon Valley, USA), ICIMP '07, July 2007, p. 21. 48, 67

[ZMD+09] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, *RFC 5475: Sampling and Filtering Techniques for IP Packet Selection*, 2009. 93, 94

[ZMSP03] T Zseby, L Mark, C Schmoll, and G Pohl, *Passive one-way-delay measurement and data export*, Proceedings of the 2003 International Workshop on Inter-domain and Performance Simulation (Salzburg, Austria), IPS '03, February 2003. 93

List of Publications

Publications Directly Related to this Thesis

1. J. Aracil, J. Ramos, J.E. López de Vergara Méndez, L. de Pedro Sánchez and S. López Buedo, *Aparato para la medición certificada del ancho de banda de un acceso de red y método de calibración del mismo (APPLIANCE FOR THE CERTIFIED MEASUREMENT OF THE BANDWIDTH OF A NETWORK ACCESS AND METHOD FOR THE CALIBRATION THEREOF)*, International Patent PCT/ES2010/070269, Spain, 28/04/2010.

2. J. Aracil, J. Ramos, P.M. Santiago del Río, J.E. López de Vergara Méndez, L. de Pedro Sánchez, S. López Buedo, I. González Martínez and F.J. Gómez Arribas, *Método para estimar los parámetros de un elemento de control de tipo Token-Bucket (METHOD FOR ESTIMATING THE PARAMETERS OF A CONTROL ELEMENT SUCH AS A TOKEN BUCKET)*, International Patent PCT/ES2011/070239, Spain, 09/04/2011.

3. J. Ramos, P.M. Santiago del Río, J. Aracil and J.E. López de Vergara, *On the effect of concurrent applications in bandwidth measurement speedometers*, Computer Networks (2011), Vol. 55, Issue 6, pp. 1435-1453.

Publications in Topics Related to this Thesis

1. V. Moreno, P.M. Santiago del Río, J.Ramos, J.J. Garnica, J.L. García-Dorado, *Batch to the Future: Analyzing Timestamp Accuracy of High-Performance Packet I/O Engines*, IEEE Communications Letters (2012), Vol. 16, Issue 11, pp. 1888-1891.

2. J.L. García-Dorado, F. Mata, J. Ramos, P.M. Santiago del Río, V. Moreno and J. Aracil, *Chapter 1: High-performance network traffic processing systems using commodity hardware*, Data Traffic Monitoring and Analysis, Lecture Notes in Computer Science 7754 (2013), pp. 327.

3. P. M. Santiago Río, J. Ramos, A. Salvador, J. E. López de Vergara, J. Aracil, A. Cuadra and M. Cutanda, *Application of Internet Traffic Characterization to All-Optical Networks*, Proceedings of the 12th International Conference on Transparent Optical Networks (Munich, Germany), ICTON '10, June 2010.

4. P. M. Santiago Río, J. Ramos, J.L. García-Dorado, J. Aracil, A. Cuadra and M. Cutanda, *On the processing time for detection of Skype traffic*, Proceedings of the 2nd International Workshop on Traffic Analysis and Classification (Istanbul, Turkey), IWCMC2011-TRAC, July 2011.

5. I. Csabai, A. Fekete, P. Hága, B. Hullár, G. Kurucz, S. Laki, P. Mátray, J. Stéger, G. Vattay, F. Espina, S. García-Jimenez, M. Izal, E. Magaña, D. Morató, J. Aracil, F.J Gómez, I. González, S. López-Buedo, V.Moreno and J. Ramos, *ETOMIC Advanced Network Monitoring System for Future Internet Experimentation*, Proceedings of the 6th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Berlin, Germany), TRIDENTCOM '10, May 2010.

6. J. Fullaondo, P. M. Santiago Río, J. Ramos, J.L. García-Dorado, and Javier Aracil, *AP-CAP framework: Monitorizando a 10 Gb/s en hardware de propósito general*, in Actas de las X Jornadas de Ingeniería Telemáatica, (Santander, Spain), JITEL '11, September 2011.

7. F. Mata, J.Ramos, A.Cuadra, A.Ferreiro and N.Gómez, *Monitorización de tráfico IP para el control de calidad de servicio en entornos convergentes*, in Actas de las XIX Jornadas Telecom I+D (Madrid,Spain), November 2009.

Appendix A

Memory and CPU load programs

Memory load program

```
allocate memory;
while TRUE do

end while
```

CPU load program

```
float f1,f2;
while TRUE do
    open file;
    read file;
    close file;
    f1*f2;
end while
```

Appendix B

Multinomial Minimum Distribution Calculation

The distribution for the minimum amount of packets which drops in a gap when m packets are scattered across $N-1$ gaps it is hard to obtain. It is worth noticing that the amount of all possibilities of distributing packets into the gaps increases quickly. For instance, 10 gaps and 50 packets gives over 12 billion possibilities. A faster way to calculate the minimum distribution has been proposed in [Cor11].

As is proposed in [Cor11], the problem of multinomial distribution can be posed as a stochastic process which represents the gap filling process. Fig. B.1 shows the process diagram, which reads as follows: the generic state i/s_i gives the number of packets, s_i, which fill the first i gaps. For instance, the state labeled 1/0 means that neither packet have dropped into the first gap. The diagram in Fig. B.1 also gives the transition probabilities, in general, from state s_{k-1} (s_{k-1} packets in the first $k-1$ gaps) to state s_k (s_k packets in the first k gaps). This probability is shown in eq. B.1. For instance, transition from state 0/0 to state 1/0 occurs with probability $\left(\frac{N-2}{N-1}\right)^m$ because this transition occurs when any of m packets drops into the first gap.

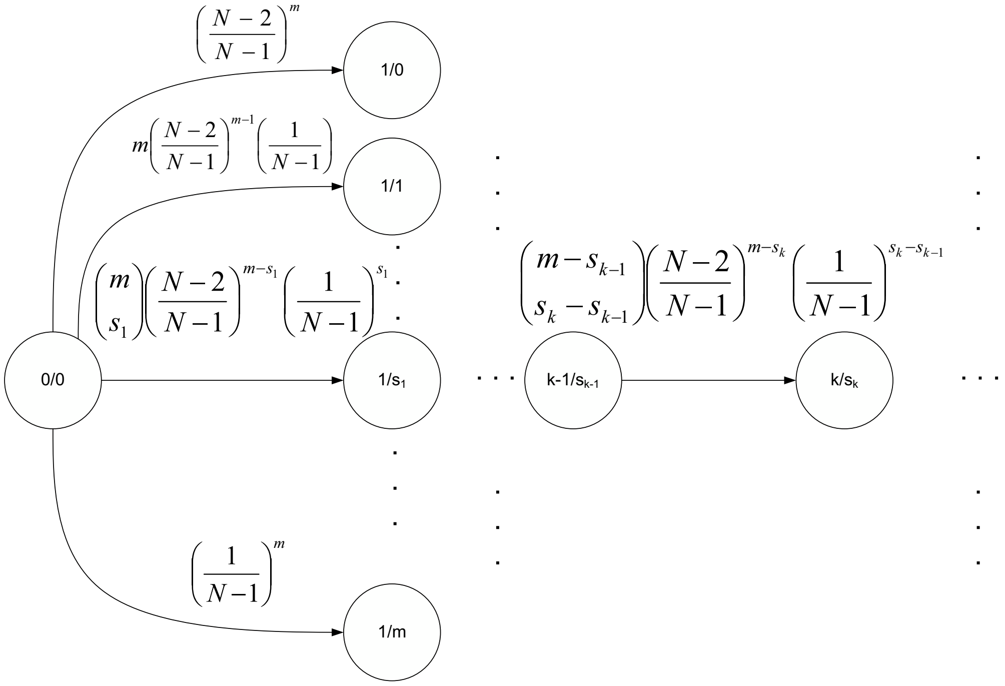

$\left(\frac{N-2}{N-1}\right)^{m}$
1/0
$m\left(\frac{N-2}{N-1}\right)^{m-1}\left(\frac{1}{N-1}\right)$
1/1
$\binom{m}{s_1}\left(\frac{N-2}{N-1}\right)^{m-s_1}\left(\frac{1}{N-1}\right)^{s_1}$
$\binom{m-s_{k-1}}{s_k-s_{k-1}}\left(\frac{N-2}{N-1}\right)^{m-s_k}\left(\frac{1}{N-1}\right)^{s_k-s_{k-1}}$
0/0
1/s$_1$
k-1/s$_{k-1}$
k/s$_k$
$\left(\frac{1}{N-1}\right)^{m}$
1/m

Figure B.1: Stochastic process related to multinomial distribution

$$P(s_k|s_{k-1}) = \left\{ \begin{array}{ll} \binom{m-s_{k-1}}{s_k-s_{k-1}} \left(\frac{1}{N-1}\right)^{s_k-s_{k-1}} \left(\frac{N-2}{N-1}\right)^{m-s_k} & \text{for } s_k \geq s_{k-1} \\ 0 & \text{otherwise} \end{array} \right\} \tag{B.1}$$

Let Q_k denote the stochastic matrix determining the transitions between the previous states, which s_{k-1}, s_k entry contains the probability shown in eq. B.1. The product $Q_1 \cdot \ldots \cdot Q_k$ represents the convolution distribution of the sum $s_k = n_1 + \ldots + n_k$. At $k = m$, the convolution distribution is degenerated on m (only takes the value m with probability one).

In order to compute the probability that no gap contains less than 1 packet, it must be set zero the transition probability between states $s_{k-1}/k-1$ and s_{k-1}/k i.e. set $P(s_k|s_{k-1}) = 0$ whenever $s_k = s_{k-1}$. Therefore, the product $Q_1 \cdot \ldots \cdot Q_m$ gives $P\left(\min_{i=1,\ldots,N-1} n_i \geq 1\right)$ exactly.

Thus, when we obtain a measurement with the packet-train method, we know the length of the packets train (N) and the total amount of interfering packets(m) and then, we can calculate the measurement rejection probability.

These measurement rejection probabilities can be computed beforehand and stored. Thus, the measurement application does not consume processing time to calculate the measurement rejection probability. It will simply deem as invalid those measurements with a measurement rejection probability higher than ϵ.

Index