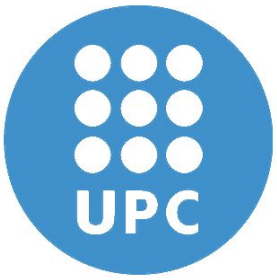
Towards End-to-End SLA Assessment



Author: René Serral-Gracià

Advisor: Jordi Domingo-Pascual

Computer Architecture Department

Technical University of Catalonia

A thesis presented to the Technical University of Catalonia in fullfillment of the requirements for the degree of doctor in Computer Science

2009 January

Abstract

The introduction of a new set of services to the Internet impose a series of tight conditions on the network usage: there is need of more bandwidth, bounded packet losses or limited delays on data delivery.

Until now, the only considered parameter was the bandwidth. Nevertheless, this paradigm is changing because of the more demanding services being deployed, namely VoIP, Videoconferencing or Video-Streaming.

The above requirements are easily met on reduced environments or small networks, where the resources are managed by a single administrative unit, and Quality of Service (QoS) mechanisms are not difficult to deploy. Nevertheless, the problem has a much more difficult solution when dealing with Inter-Domain environments. There, the traffic policies, the network usage, and the resources in general are managed by different administrative units, which most likely will have different goals and policies when distributing the available resources.

On top of this, every day more and more customers are requiring guaranties for their services, and operators want to know the network status all the time. Therefore, there is an increasing need of monitoring whether the network can provide the desired levels of quality. Or, what is the same, if the network is complying with the contracted Service Level Agreements (SLA).

This thesis has the major goal of developing an on-line distributed SLA Assessment infrastructure, which will give live reporting about the quality the network is providing. We focus on the Intra-Domain study, and outline several alternatives for extending this to the Inter-Domain area.

To develop such infrastructure, some challenges have to be addressed. First there is the need of specifying the techniques to be used in order to monitor and analyse the traffic in real-time. Second, there is the study of the resources in terms of

bandwidth required to perform the assessment. And third, there is a trade-off between the accuracy and the resources to be decided.

We solve the problem of the resource requirements by proposing a set of innovative sampling techniques, which make an efficient use of the resources.

Specifically, the main contributions of this work are:

- We develop a Network Parameter Acquisition System (NPAS), that is a scalable distributed SLA Assessment infrastructure that monitors the QoS of the sensible flows on the network.
- We analyse the current measurement techniques, proposing ways of using the knowledge acquired from metrics behaviour and Internet in general to extend and optimise NPAS.
- We optimise NPAS by using static and dynamic adaptive sampling techniques in order to control the resources needed by the system.
- In parallell we improve classical Quality of Experience (QoE) algorithms in order to work with current network technologies and adapt them to our NPAS, delivering a system which can assess the user perceived quality of specific flows.
- Finally we propose a novel way of decoupling the metric computation from the quality of the network by smart distance algorithms over Inter-Packet Arrival Times. With this decoupling we can assess the QoS of a network with minimum metric computation, with the consequent boost in performance of the system.

Resum

La introducció d'un nou conjunt de serveis a Internet, ha imposat una sèrie de noves restriccions a l'ús i comportament de la xarxa: ara es necessita més amplada de banda, un control sobre els paquets perduts o bé un llindar màxim amb el retard en l'entrega de les dades.

Fins ara, l'únic paràmetre que es considerava a l'hora de dimensionar una xarxa era l'amplada de banda. De totes formes, aquest paradigma està canviant, principalment a causa dels requeriments dels nous serveis posats en funcionament, per exemple VoIP, Videoconferència o vídeo streaming.

El requeriments necessaris són relativament senzills d'aconseguir en entorns controlats i reduïts, on els recursos són gestionats per una sola unitat d'administració, i els mecanismes per la provisió de la Qualitat de Servei (QoS) no són difícils de posar en funcionament. De totes formes, el problema té una solució molt més complicada quan estem tractant en entorns Inter-Domini. En aquest entorn les polítiques de tràfic, l'ús de la xarxa i dels recursos en general estan gestionats per diferents unitats administratives, el que molt possiblement implica que tindran diferents polítiques de gestió de tràfic i objectius diferents en la gestió de la seva xarxa i dels seus recursos.

A sobre de tot això, cada dia més i més clients demanen a la xarxa garanties per als seus serveis. En paral·lel els operadors volen saber quin és l'estat de la seva xarxa en tot moment. Per això, hi ha una necessitat creixent de monitoritzar si la xarxa pot o no pot oferir el nivell de qualitat desitjada. O bé, el que és el mateix, si la xarxa està complint amb el contracte de serveis (Service Level Agreement – SLA).

Aquesta tesi té el principal objectiu de desenvolupar un sistema distribuït per analitzar el SLA *"en directe"*. Aquest sistema donarà informació en tot moment sobre la provisió de qualitat donada per la xarxa. El nostre estudi se centra en un

entorn Intra-Domain, però s'introdueixen una serie d'alternatives que poden ser utilitzades per ampliar el sistema a un entorn Inter-Domain.

Per desenvolupar aquesta infraestructura hi ha alguns reptes que s'han de resoldre. En concret, primer hi ha la necessitat d'especificar les tècniques utilitzades per monitoritzar i analitzar el tràfic en temps real. Segon, és necessari estudiar els recursos, en termes d'amplada de banda, que són necessaris per poder realitzar la monitorització. I finalment, s'ha de tenir en compte que hi ha una solució de compromís entre la precisió del sistema i els recursos necessaris per obtenir la informació necessària.

El problema dels recursos es resol mitjançant la proposta d'una sèrie de tècniques de mostreig de tràfic innovadores, i que permeten fer un ús eficient dels recursos.

En concret les contribucions d'aquesta tesis són:

- Desenvoluparem un *Network Parameter Acquisition System* (NPAS), que és una infraestructura per la monitorització de tràfic distribuïda i escalable.
- Analitzem les actuals tècniques de mesures, proposant maneres de fer-les servir per tal d'ampliar i optimitzar NPAS.
- Optimitzem NPAS mitjançant tècniques de mostreig de tràfic tant estàtic com dinàmic, per tal d'ajustar els requeriments de recursos del sistema.
- De forma complementària proposem un sistema per millorar mecanismes clàssics de Qualitat de l'Experiència (QoE). El nostre objectiu és adaptar els mecanismes existents al nou paradigma d'Internet, i ajustar-los al nostre NPAS. Permetent saber la percepció des del punt de vista de l'usuari dels fluxes de xarxa.
- Finalment, proposem un sistema innovador de desacoblar el càlcul de les mètriques de xarxa de la seva qualitat. El que fem en aquest cas, és proposar una sèrie d'algorismes per analitzar el temps d'arribada entre paquets (IPAT). Amb aquest desacoblament aconseguim que es pugui informar de la QoS d'una xarxa minimitzant el càlcul de les mètriques, amb el conseqüent increment en en rendiment del sistema.

The Wheel of Time turns, and Ages come and pass, leaving memories that become legend. Legend fades to myth, and even myth is long forfotten when the Age that gave it birth comes again. In one Age, called the *Internet* Age by some, an Age yet to come, an Age long past, a wind rose above the office named D6-111. The wind was not the beginning. There are neither beginnings nor endings to the turning of the Wheel of Time. But it was *a* beginning.

Robert Jordan, The Wheel of Time, Books 1-11.

Arrangements by *René Serral*, Towards End-to-End SLA Assessment, Book 1.

To Caterin, my life.

Acknowledgements

Després de tants anys i tanta feina feta, la llista de gent que ha participat directa o indirectament en aquesta tesi és massa gran per llistar-la.

De totes maneres, m'agradaria donar les gracies a Jordi Domingo per donar-me la oportunitat de ser aquí i poder fer allò que més m'agrada professionalment, investigar. De la mateixa manera voldria agrair als meus companys d'agonies, l'Albert i el Pere, per haver-me aguantat durant tots aquestos anys en infinites discussions sobre un tema o altre per tal de millorar el treball que s'estava realitzant al moment.

També voldria donar les gracies molt especialment al Marcelo i al Xavi, que durant tots els viatges i amb la feina a fer han sabut treure el millor de mi mateix en tot moment, tant des del punt de vista personal com professional.

Canviant de registre, voldria agrair al David i a tots els mandrils per ajudar-me a trencar la rutina i donar-me una via d'escapament amb esmorzars, dinars i festorres molt agradables, a la vostra manera també heu aportat un petit gra d'arena a la realització d'aquesta feina.

No puc acabar la dedicatòria sense agrair als meus pares i germana, Jordi, Avelina i Ma Pilar, i als meus avis i padrí Francisca, Joan i Rafel pel suport moral que sempre m'han donat, molts cops sense saber-ho... També, i de forma molt destacada vull agrair a la meva parella, Caterin, que ha aguantat els meus mals humors, monòlegs i les meves paranoies durant més anys dels que em mereixo. Caterin la meitat de la feina d'aquí es teva també!.

Contents

List of Figures

List of Tables

Glossary

AB	Available Bandwidth
ABA	Abandon Rate
AC	Admission Control
AF	Assured Forwarding
ASA	Average Speed to Answer
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
BTC	Bulk Transfer Capacity
CBR	Constant Bit Rate
COPS	Common OpenPolicy Service
CoS	Class of Service
CRC	Cyclic Redundancy Check
DSCP	Differentiated Services Code Point
EF	Expedited Forwarding
EuQoS	End-to-End QoS over heterogeneous networks
FCR	First Call Resolution
FEC	Forward Equivalent Class
FTP	File Transfer Protocol
GPS	Generic Positioning System
HTTP	HyperText Transfer Protocol
IDRP	Intra-Domain Reporting Protocol
IDSE	Inter-Domain Subscriber Entity
IDSP	Inter-Domain Subscriber Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPAT	Inter-Packet Arrival Time
IPDV	Inter Packet Delay Variation
IPPM	IP Performance Metrics
IPTD	IP Transfer Delay
ISP	Internet Service Provider
ITU	International Telecomunications Union
LBC	Link Bandwidth Capacity
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label Switching Router
MD	Measurement Domain
ME	Monitoring Entity
MMFM	Monitoring Measurements and Fault Management
MMS	Measurement and Monitoring System
MOS	Mean Opinion Score
MPLS	Multi-Protocol Label Switching
MTU	Maximum Transfer Unit
NPAS	Network Parameter Acquisition System
NTP	Network Time Protocol
OWD	One-Way Delay
P2P	Peer to Peer
PDB	Per-Domain-Behaviour
PE	Processing Entity
PHB	Per Hop Behaviour
PLR	Packet Loss Ratio
PPS	Pulse Per Second
QoE	Quality of Experience
QoS	Quality of Service
RDS	Reference Distribution Set
RED	Random Early Detection

GLOSSARY

RFC	Request For Comments
RSVP	ReSerVation Protocol
SLA	Service Level Agreement
SLS	Service Level Specification
TAT	Turn Around Time
TCP	Transport Control Protocol
ToS	Type of Service
TSC	TimeStamp Counter
TSF	Time Service Factor
TTL	Time To Live
TW	Time Window
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VCI	Virtual Circuit Identifier
VID	Valid IPAT Distribution
VPI	Virtual Path Identifier
WFQ	Weighted Fair Queueing
XML	eXtensible Markup Language

1

Introduction

Internet's usage has increased greatly during the last years. Such growth has evolved in many protocols, applications and services which demand a more intensive use of the network resources.

Network resources typically have been counted only in terms of bandwidth. But as services get more involved, so do the constraints they impose on the network. Usually, such constraints include network quality parameters, for example packet losses, transmission delays, or delay variations. They *measure* the degree of quality present in a network, and in some cases their impact on the perceived quality by the users.

Transmission delays interfere in the fact that some services rely in interactivity, packet losses are the main reason of service disruption, and finally delay variations, impact both on the interactivity and in the quality of the transmission.

There are mainly two different kinds of services affected by this: human-machine interaction (e.g. remote shell connection). And human-human interaction (e.g. videoconferencing). When dealing with such applications, it is mandatory to keep those metrics within specific bounds to get such interactivity. Obviously, no conversation (or interaction) can be held if there are big one-way delays between the emission of the message and its reception.

All the above discussion, leads to the need of the network administrators to provide, by some means, the guarantee of the delivered service to their users. On the other hand, the users will require some mechanisms to validate such provisioning. This assessment framework sets the main axis where all our research work will be focused.

Objectives

The main goal of this thesis is to develop a scalable end-to-end infrastructure for on-line SLA assessment, namely the *Network Parameter Acquisition System* (NPAS). We focus on the intra-domain analysis, overviewing mechanisms that can extend our work to inter-domain environments.

The progress towards the full architecture will be done incrementally from the most simple approach, to the introduction of the more advanced techniques which perform the SLA assessment. Improving aspects such as resource requirements from the base system.

The presented methodology will deliver a versatile mechanism to study the network's performance, and whether the network is complying with the desired Service Level Agreement (SLA) or not. This is accomplished at two different levels: first, it will provide objective network quality assessment, by using the classical network performance metrics. And second, it will also deliver user level information, namely subjective quality assessment by using the well-known Mean Opinion Score (MOS).

All the traffic monitoring tasks are performed by means of passive traffic collection, in this regard we assume during this whole thesis that the system can cope with the collection of all the required packets from the network to perform the analysis.

Motivation

The main motivation to perform this detailed analysis and research about SLA assessment is three-fold: *i)* The impressive growth of real-time flows on the Internet forces the network to have mechanisms to assess the proper delivery of such traffic. *ii)* Currently there is no clear and scalable solution to the problem. And *iii)* Tightly related with the above point, lately the governments are pushing the adoption of new standards in order to demand from the service providers guaranties that they are providing the required network quality the customers are paying for.

Regarding scalability, the solutions currently present are mostly based on active traffic measurements, which tend to give a coarse result about the network quality, moreover they are not designed for continuous monitoring. However, there is a framework with similar goals as ours (perfSONAR [97]). Many efforts are being invested in this platform in order to produce

a reliable active SLA assessment tool. We present an alternative solution which aims at giving much more accurate on-line information about the status of the network.

Related to the standardisation efforts, the *European Telecommunications Standards Institute* (ETSI) plans to standardise the set of constraints and testing methodology which will be mandatory in a near future for any service provider [2]. This will help both the customers and Internet Service Providers (ISP) in the hard task of assessing (for ISPs) or having guaranties (for Customers) about the real quality and behaviour of the network.

Our work is aimed at filling the gap between the specifications on what, how and where the ISP should monitor their networks to the actual task of monitoring it. As a bonus this methodology is designed to work on-line. Nevertheless, we do not plan to specify and implement all the techniques and restrictions presented in [2]. Our focus is at resolving the broad range of challenges found in such a distributed monitoring infrastructure.

Contributions

This thesis is structured into different building blocks which incrementally construct a full-fledged SLA Assessment solution. Here we present the relevant contributions on each one of the different areas of this work.

We separate here the different contributions depending on the part of the whole SLA assessment system it fits: *Preliminary work*, *Base system*, *Optimisations*, *User level* and *Traffic Pattern analysis*.

Preliminary work

Here we have the contributions related to background work we performed to understand the way network measurements work, these publications do not have direct relation with the main contribution of the thesis, but deliver basic knowledge about network measurements, which we used to design the rest of the work.

1. *Serral-Gracià, R.*, and Gil, Marisa: **A Linux Networking Study**, *In Operating System Review, Volume 38 Number 3 (SIGOPS ACM)*, July 2004.

2. Cabellos-Aparicio, A., *Serral-Gracià, R.*, Jakab, L., and Domingo-Pascual, J.: **Measurement Based Analysis of the Handover in a WLAN MIPv6 Scenario**, *Passive and Active Measurements (PAM)*, LNCS 3431, 207–218, 2005.

3. *Serral-Gracià, R.*, Cabellos-Aparicio, A., Julian-Bertomeu, H., and Domingo-Pascual, J.: **Active measurement tool for the EuQoS project**, *MOME 3rd International Workshop on Internet Performance, Simulation, Monitoring and Measurements (IPS-MoMe)*, 2005.

4. Jakab, L., *Serral-Gracià, R.*, and Domingo-Pascual, J.: **A Study of Packet Losses in the EuQoS Network**, *MOME 4rd International Workshop on Internet Performance, Simulation, Monitoring and Measurements. IPS-MoMe*, 2006.

5. *Serral-Gracià, R.*, Jakab, L., and Domingo-Pascual, J.: **Out of Order Packets Analysis on a Real Network Environment**, *2nd Conference on Next Generation Internet Design and Engineering*, 2006.

6. *Serral-Gracià, R.*, Domingo-Pascual, J., Bęben, A., and Owezarski, P: *Chapter 2 - QoS Measurements in IP-based Networks* in **End-to-End Quality of Service Over Heterogeneous Networks**, *Springer, Eds: Braun, Torsten, and Staub, Thomas*, Aug 2008.

These six contributions detail the most interesting aspects of network measurements. Specifically, in 1 we study the constraints found in commodity hardware when trying to monitor high amounts of packets in the network. In 2 we focus on the study of the different effects into the measurements of wireless access technology. While in 3 the research is centred on the effects of large testbeds with different access technologies in network measurements. In 4 and 5 we study the metrics behaviour for QoS assessment. And finally in 6 we performed an overview of the most important measurement techniques in the area of Quality of Service.

In order to understand Inter-Domain technologies and behaviour we explained and deployed a full QoS aware routing solution in inter-domain systems in:

1. Masip-Bruin, X., Yannuzzi, M., *Serral-Gracià, R.*, et al.:, **The EuQoS System: A Solution for QoS Routing in Heterogeneous Networks**., *IEEE Commun. Mag 45(2)* 96–103, 2007.

Base System

The base work which sets the starting point of the global contribution of this thesis is:

1. *Serral-Gracià, R.*, Barlet-Ros, P-, and Domingo-Pascual, J.: **Coping with Distributed Monitoring of QoS-enabled Heterogeneous Networks**, *4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks (QoSIP - IT-NEWS*, 142–147, 2008.

Where we detail all the basic methodology and most important building blocks for SLA assessment. This sets the grounds of our proposal, which will be enhanced by the different optimisations and techniques that form the main contribution of this thesis.

Optimisations

The most interesting contributions of this thesis fall in this area. They focus on optimisations over the basic system in terms of required resources (bandwidth) for the SLA assessment.

1. *Serral-Gracià, R.*, Barlet-Ros, P., and Domingo-Pascual, J.: **Distributed Sampling for On-line QoS Reporting**, *Local and Metropolitan Area Network (LANMAN)*, Sep 2008.

2. *Serral-Gracià, R.*, Cabellos-Aparicio, A., and Domingo-Pascual, J.: **Network performance assessment using adaptive traffic sampling**, *IFIP Networking* LNCS 4982, 252–263, May 2008.

3. *Serral-Gracià, R.*, Cabellos-Aparicio, A., and Domingo-Pascual, J.: **Packet loss estimation using distributed adaptive sampling**, *End-to-End Monitoring (E2EMon)*, Apr 2008.

As a first optimisation we present in 1 a Static Sampling approach which is used in order to reduce the amount of required resources. Given the low accuracy in packet loss estimation, we develop in 2 and 3 two similar Adaptive Sampling algorithms which improve significantly the packet loss estimation while keeping a good estimation of the rest of the metrics.

User level

On a related topic we used our acquired knowledge, this time for enhancing our SLA assessment system in order to deliver higher layer information, namely Quality of Experience. The main contribution is:

1. *Serral-Gracià, R.*, Jakab, L., and Domingo-Pascual, J.: **Measurement Based Call Quality Reporting**, *2nd Workshop on Network Measurements in 32nd IEEE Conference on Local Computer Networks (LCN)* 997–1004, 2007.

Where we present an improvement over the classical definition of the Mean Opinion Score (MOS) with the aim of adapting it to current packet switched networks, fitting it to the main system.

Traffic Pattern analysis

1. *Serral-Gracià, R.*, Labit, Y., Domingo-Pascual, J. and Owezarski, P.: **Towards efficient SLA Assessment**, *Pending Acceptance Infocom*, 2009.

In this contribution we change the typical paradigm of metric inference for SLA Assessment and focus on a first approach to metric-less SLA Assessment infrastructure. In this contribution we still require reduced information about the metrics but we set the basis in order to fully remove them.

The rest of the document is structured as follows. Chapters 2 and 3 give some general concepts required in order to understand the background related with this thesis, specifically we introduce QoS concepts and present the basis of network testing. We follow the discussion in 4 where we introduce the general challenges we must consider previous to design a robust SLA Assessment system, in that chapter we also detail the preliminary contributions of this thesis since their goal is to acquire more knowledge about the limits and best practices on network monitoring.

The core of the thesis begins in Chapter 5 where we present the basis and the core of our contribution, stating the main building blocks and the detailed description of the protocols involved in our passive SLA assessment system. We optimise and enhance this base system on the rest of the chapters. Specifically we have mainly two kinds of optimisations: The first is focused on the optimisation of metric computation, while the second is centred on the change of paradigm we present in our disruptive approach, which decouples metric estimation from SLA assessment, we accomplish this by means of Inter Packet Arrival Time (IPAT) analysis. In detail:

Chapter 6 presents the most basic optimisation, which is based on a time based classification that permits to reduce the required bandwidth due to control traffic.

Chapter 7 further improves the resource utilisation by inserting traffic sampling to the system. This uncovers a new problem that is the lack of accuracy of the system for low rate flows in the estimation of packet losses. To overcome this limitation, in Chapter 8 we present two different adaptive sampling techniques that deliver much better accuracy with even further reducing the required resources.

In Chapter 9 we extend all the above techniques to a higher level, the subjective quality assessment, specially for adapting our system to VoIP quality assessment using the Mean Opinion Score (MOS).

As a a final enhancement of the system presented in this thesis, in Chapter 10, we present a totally novel approach which separates the classical tie between metrics and network quality, here we present a different methodology which uses IPAT as trigger for SLA assessment.

Finally in chapter 11 we conclude and draw the further lines of research left open in this work.

2

Quality of Service

As Internet continues its expansion, more and more services are deployed. These services usually have very specific demands of the network. Some of such demands are typically defined through metrics such as bandwidth, packet losses, etc. [123].

All these constraints end up on the need of better network resource management, and therefore with the improvement of the user experience using such services.

From the network point of view all this *sensible* data produced by the applications need a special treatment while travelling towards its destination. All the used policies, forwarding methods and traffic classification algorithms are part of what is called *Quality of Service* (QoS).

This chapter details the basis of QoS Provisioning, specially Integrated and Differentiated Services. Later we focus on some proposals in inter-domain extensions for provisioning end-to-end QoS. The chapter concludes with a discussion about the Quality of Experience (QoE), that is the application level point of view of QoS.

2.1 Quality of Service Provisioning

QoS is the set of mechanisms where the network traffic is treated in a special fashion. By default Internet uses the *Best Effort* [74] mechanism for treating the traffic. This mechanism is based in the simplest algorithm, where the packets do not have any priorities, so they are processed in the order they keep arriving to the gateway.

Given that *Best Effort* is clearly insufficient for demanding services, the community has developed several ways of dealing more fairly with the network resources. All the mechanisms in use nowadays for such provisioning can be classified on the following types:

- *Integrated Services [19]*: this method is based on a controlled resource reservation mechanism for the selected traffic.

- *Differentiated Services [15]*: this mechanism does not reserve resources as the above methodology, here the traffic is aggregated on different classes, where each class has its own contract with the network.

- *Multi-Protocol Label Switching [105]*: MPLS adds a new layer on the protocol stack for easing the bottleneck on the core routers. Takes the best of DiffServ and IntServ.

2.1.1 Integrated Services

Integrated Services (IntServ) is the IETF's scheme to introduce QoS support over IP networks. It provides extensions to the best-effort service model to allow control over end-to-end packet delays. IntServ is a per-flow, resource reservation model. Its key building blocks include resource reservation and admission control, which will be discussed later.

Different applications behave differently depending on its network constraints. That is the reason why the IETF defines three different kinds of application profiles:

- *Elastic applications:* applications which use TCP as a transport protocol, which adapts the network use to the bandwidth availability. This applications do not have usually any timing constraints. Examples can be FTP transfers or HTTP connections.

- *Tolerant real-time applications:* in this category fall the voice/video streaming kind of applications. Here there are bandwidth constraints, but the delay of the information delivery is not critical, since there is not interactivity between the user and the application.

- *Intolerant real-time applications:* VoIP or videoconferencing are good examples of applications in this category. These have tight bandwidth constraints, necessity of small packet delivery delays and few packet losses. The main reason being the interactivity between the parts using the application (telephony conversation, tele-meeting, etc.).

As can be noted the above categories have a different set of requirements of the network. The IntServ framework defines two different services besides *Best Effort*: *Controlled Load* and *Guaranteed Service*.

Controlled load: This class of service is prepared for controlling the Tolerant real-time applications shown above, which have a bounded resource requirements. The constraint is that such applications need some kind of bandwidth reservation, *"as if no other traffic is crossing the link"*. To do this the application must perform some kind of resource reservation on the network.

Moreover, for guaranteeing the proper provisioning there is need of an admission control (AC) system. AC has the goal of avoiding the overload of the network with the corresponding degradation of the service, more details about AC can be found in Section 2.1.1.3. The Controlled load, implicitly bounds the resource requirements of the traffic. Eventhough due to potential variability on the traffic profile, this class of traffic can adapt to some burstiness.

Guaranteed service: Guaranteed service forces the traffic belonging to the class to strict reservation of resources. It provides assurance of a certain end-to-end bandwidth and upper and lower bounds for delay.

This class of service is used for Intolerable Real-time traffic, which has the traffic profile of CBR (Constant Bit Rate) or rt-VBR (Real-Time Variable Bit Rate) . Its implementation on actual gateways is done as a token bucket, which forces a constant bandwidth with very limited support for bursty traffic (not usual on such applications). The main issue with this solution is its difficulty for deploying the service on shared medium networks, where specific access technology dependent solutions must be developed.

2.1.1.1 Signalling

For having a guaranteed service, is necessary to setup the connection along the whole path which the packets will follow to reach its destination. This requires to have a reliable mechanism to perform such reservation, the signalling.

Nowadays, the most common system used for this signalling is RSVP (ReSerVation Protocol) , the in-deep description of this protocol is out of the scope of this document, for more information see [80], [20] and [128].

A signalling protocol has to monitor all the time the used resources of the network, because it has to decide whether it accepts a new connection or not. Related with that, another important point, is the fact that the best QoS which can be offered, always will be as much as good as the one offered by the weaker node.

The typical steps the signalling protocols use are:

1. The application asks for network resources.

2. Each element checks if it can handle the demanded QoS.

3. Each network element accepts or denies the allocation.

The most used signalling protocols used with IntServ differ from the ones proposed on other technologies (such as ATM) in the sense that use a soft connection approach, where the reservation only lasts for a given time (times out).

2.1.1.2 Reservation Specs

Another important part of IntServ is the reservation itself, when an application needs to reserve resources from the network it needs to specify the kind of constraints it has. IntServ solves this with the *Reservation Specs*. There are two kinds of Reservation Specs:

- *FilterSpecs:* FilterSpec specifies the set of data which has to receive QoS. The rest of the traffic is considered *Best Effort*.

- *FlowSpecs:* Is the QoS specified for being used by the Admission Control and the scheduler during packet forwarding. In-deep description of Flow Specs is out of the scope of this document.

2.1.1.3 Admission Control

The network resources are finite, that means limitations on maximum numbers of flows, bandwidth constraints, etc. All this forces the IntServ framework to monitor the state of the resources and to compute the viability of accepting new flows.

The problem is aggravated by the fact that this control has to be monitored on each hop of the IntServ enabled network independently, because each hop can have different resources or flows on a given time.

Another task of the Admission Control is to renegotiate the Specs of new incoming flows in the case that there aren't enough resources available.

Usually the policy behind this is: better not to give a service at all than giving it defective.

2.1.1.4 Traffic Shaping and Policing

When all the connection is setup, and all the resources are allocated, there is another issue to consider: decisions to make when the flows do not comply with the Specs.

When a flow sends more traffic than is allowed to, the gateway has to *Shape* the traffic forcing it to comply with the specs. The rest of the traffic which does not fit on this shaping can be treated in several ways:

- Drop the packets using algorithms such as RED [48].
- Treat the rest of the traffic as *Best Effort*.
- Mark the packets for later dropping in case of congestion.
- Take care that the non-conformant traffic does not affect the conformant one.
- Do nothing.

2.1.1.5 Issues with IntServ

The main problem in IntServ is scalability, the fact that this framework monitors and reserves resources for each flow is unbearable on big topologies like Internet, where a core router can have hundreds of thousands of simultaneous flows. The memory and computational power required for maintaining information too high.

Another minor problem found on IntServ is that this framework forces the application to know the traffic parameters before even starting the transmission. This constraint can be easily achieved on some protocols, but is almost unpredictable on others, where burst traffic is more common.

2.1.2 Differentiated Services

As seen in the previous section, IntServ problem relies on scalability. For solving this issue the Differentiated Services (DiffServ) framework was designed.

DiffServ solution to QoS Provisioning takes a more generic approach when classifying traffic. IntServ based its approach in a-priori resource reservation on a per flow basis, with all the problems related with this. On the other hand, DiffServ rather gives priority to the aggregated traffic by taking precedence in the gateway, but without actual resource reservation.

This solution permits to use the free bandwidth, when not used by higher priority classes, by any traffic passing the link.

Such methodology enables efficient use of the bandwidth, with a much more scalable infrastructure. The drawback of this approach, though, is the lack of control over specific flows.

Without the big overhead of resources management for handling all the flows, the mechanism, increases the forwarding performance, thus doing of DiffServ a proper candidate to be used on high load links, where millions of flows are forwarded constantly, namely, the core network.

2.1.2.1 Traffic Classification

One of the main strengths of DiffServ is its ability to aggregate many flows into classes. This permits to ease flow management to the point of Class management.

DiffServ as stated on [91] defines different classes of service, where the traffic is classified depending on its characteristics, there are two main classes: *Expedited Forwarding (EF)* and *Assured Forwarding (AF)*. In the first class falls all the traffic which need low losses, low latency, low jitter and bandwidth assurance inside the whole DiffServ Domain. While Assured Forwarding is developed for the traffic with weaker network constraints.

Such classification into classes, where several flows will belong to the same class, can be achieved by marking the packets which belong to a class. This marking has to keep compatibility with any router or host which is not aware of the DiffServ technology. This is accomplished by the use of the *Differentiated Services Code Point (DSCP)* field on the IP header. Part of the older *Type of Service (ToS)* field.

ToS: This field has 8 bits. Its goal is to change the default forwarding policies of the gateway for special traffic. There is only a consensus on the first 3 bits of the whole field, which indicate the IP precedence. It ranges from 111 (max priority) meaning Network control, to 000 (minimum priority) for routine traffic. A full discussion about these bits can be found at [32].

The rest of the bits are divided in two parts, one part with 4 bits where there isn't consensus about its meaning. These are defined on [8], but due to the lack of standardisation, generally, are ignored by the routers. Its meaning is shown on Table 2.1. The remaining bit is marked as unused and will be left for future use.

ToS	*Semantics*
1000	Minimise delay
0100	Maximise throughput
0010	Maximise reliability
0001	Minimise monetary cost
0000	Normal service

Table 2.1: RFC-1349 ToS values

DSCP: Given the issues with the ToS field, IPPM Working group decided to provide a more generic framework for the ToS field and created the DiffServ CodePoint [91], with the aim of providing a facility so that packets marked with specific DSCP have a defined performance or forwarding behaviour at each hop.

DSCP uses 6 bits of the ToS field, with the other two left for future use. This field has been splitted into pools of Codepoints, with a different goal for each pool as shown in Table 2.2.

Pool	*Codepoint Space*	Assignment
1	xxxxx0	Standard action
2	xxxx11	Experimental/local action
3	xxxx01	Experimental/local action (subject to standardisation)
Default	000000	Best-effort forwarding
	xxx000	For IP precedence compatibility

Table 2.2: RFC-2474 DSCP

The table shows the general form of the field, currently there is only standardised the DSCP for *standard action*, leaving the rest for testing on experimental networks. Another important point is that the codepoint is backward compatible with the IP precedence on the old ToS. It also has backwards compatibility with the default Internet behaviour, which leaves the value 000000 to the best effort traffic.

All in all, DSCP's limitation are given by the maximum classes available, that is 64. The upside of this limitation is that each domain can specify the service provided autonomously.

Despite the 8bit available for Classes of Service (CoS), from the practical point of view only a subset is used. Table 2.3 shows the classical division into CoS as described in [64], the

table details each CoS in terms of metric constraints. Details about the metrics and its definition can be found in Section 3.1.

Network Parameter	QoS Classes					
	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5
IPTD	$100ms$	$400ms$	$100ms$	$400ms$	$1s$	U
IPDV	$50ms$	$50ms$	U	U	U	U
IPLR			1×10^{-3}			U
IPER			1×10^{-4}			U

Table 2.3: QoS class definitions and performance objectives. IPTD: IP Transfer Delay; IPDV: IP Delay Variation; IPLR: IP Loss Ratio; IPER: IP Error Ratio

2.1.2.2 Per Hop Behaviour

The continuous changes of services, and service constraints, on the Internet forced DiffServ to define classes and not services. Classes define general behaviours which are not tied to actual services. In fact, the service, or services, will be mapped to a class which will receive a special treatment at each hop of the whole DiffServ domain. This behaviour definition is called Per-Hop-Behaviour (PHB).

PHB specifies the traffic constraints, such as reserved bandwidth to the class, which will be formed by different flows or services, named Behaviour Aggregates (BA).

PHB does not define how the packets will be treated at the gateway, because that is addressed by using queueing mechanisms, such as WFQ or CBQ. PHB just defines how the final result has to be, complying with the specified contract or Service Level Agreement (SLA).

When several PHB are similar, they end up forming which is called a PHB Group. An example of PHB Groups is Assured Forwarding.

The only difference between classes is DSCP, which detailed meaning depends on the actual implementation on the gateway. This can differ greatly among administrative domains, thus arising the need of some compatibility mapping across them. This mapping is called Per-Domain-Behaviour (PDB) defined on [92].

PDB defines the SLA of the packets on the ingress and egress points of a domain through the Service Level Specification (SLS). The actual result of this is a DSCP mapping between two administrative units (domains).

2.1.2.3 Service differentiation

In a DiffServ environment, the packets arrive usually with a zero DSCP, and is the DiffServ router who will need to mark the packets given some predefined conditions. For this marking to be possible, DiffServ has to provide some hints on which possible classes are available for sending the aggregates to.

Besides of Best-Effort, DiffServ defines two different categories, Expedited Forwarding (EF) [66] and Assured Forwarding (AF) [54]. Those classes have a specific DSCP belonging to the Pool 1 of the DSCP classification.

Expedited Forwarding: Also known as Premium Service. DiffServ does not standardise such services, only the PHBs, but IETF decided to present some sample services. EF being their first example, where the EF PHB gives its traffic low loss, low latency, low jitter and bandwidth assurance.

Given those tight constraints, EF traffic has to be tightly limited to avoid starvation of other aggregates. When EF is used it has absolute priority regardless the other traffic on the link, when EF exceeds the assigned bandwidth for its contract, it is immediately discarded to avoid congestion.

This service should be used only on reduced traffic with very tight constraints. The reserved Codepoint for this aggregate is 101110.

Assured Forwarding: The EF class is unique. On the other hand AF has the particularity of being a PHB Group, this way a service provider can classify its customers depending on their contracted service, AF defines four classes (sometimes three of them named after the sports medals: Gold, Silver and Bronze) and on each class there are three drop precedences of the exceeding traffic as depicted on table 2.4

Drop Precedence	*Class AF1*	*Class AF2*	*Class AF3*	*Class AF4*
Low (1)	001 010	010 010	011 010	100 010
Medium (2)	001 100	010 100	011 100	100 100
High (3)	001 110	010 110	011 110	100 110

Table 2.4: AF Drop precedence

The deployment of AF (together with EF) requires some admission control for not reserving more resources than the actually available.

Each class has allocated its own resources, each packet has assigned a queue which depends on its contract, when the network is congested, the administrator can decide to increase the drop precedence for customers with less charging fees given that AF only provides soft guarantees.

2.1.3 Multi Protocol Label Switching (MPLS)

MPLS as stated on [105] and [104] has the aim to combine switching functionalities of Layer 2 with advanced routing capabilities found on Layer 3. This mixed approach differentiates MPLS over the common IP routing framework:

- No more necessity of the longest match algorithms of IP routing.
- IP routing table reduction. Now the gateways switch IP packets.

Basically MPLS defines the following concepts which form its specification:

- *Forward Equivalent Class (FEC):* group of packets forwarded through the same path and with the same treatment on the routers.
- *Label Switching Router (LSP):* an MPLS node capable of forwarding packets with labels.
- *Label Edge Router (LER):* entry/exit point for an MPLS domain to/from the IP world.

Label switching permits that the IP header is only checked once at the entrance of the MPLS domain, LER is in charge of process the IP header like a common router and then attaches the Label to the packet. From that moment, all the routers on the domain handle the packet using that label.

2.1.3.1 Basic description of MPLS

As said before, instead of the longest match for routing, MPLS places a label to do the work. Each FEC has the same label inside the domain. The label is used for forwarding. The LER at the exit of the domain extracts the label.

Such label belongs to the MPLS header which has 32 bits and is formed by:

- Label (20 bits): the actual label. A stack of labels can be used for hierarchical routing.

- Exp (3 bits): Experimental.
- S (1 bit): indicates whether this is the last label on the stack.
- TTL (8bits): this field gets copied by the LER from the TTL in the IP header. And is written back on the exit point (also the LER).

Any intermediate router can change the label depending on a table which indicates Next-Hop and new label (label swapping), similar to ATM networks, where a Virtual Circuit in a switch has an entrance VPI/VCI which correspond to an exit VPI/VCI.

2.1.3.2 Label Distribution

For the protocol to work properly, the entities of the MPLS domain have to agree on the used labels, its meaning and the association with the actual FEC. That's why a Label Distribution Protocol (LDP) is needed.

For commodity, the LDP protocol is built on top of existing protocols, i.e. RSVP, which was originally intended for IntServ. The protocol determines the action which has to be taken when a packet with certain label arrives at the router. Common actions involve: pop a label, push a label, swap labels, etc. More information can be found at [68].

2.1.3.3 Traffic Engineering

Another important subject to have in mind, are the QoS and route optimisations permitted by the protocol. In fact, the Reservation Protocol used for the label distribution, with minor modifications (RSVP-TE) can be used for resource reservation, but in this environment refererring to a FEC, and not to a single flow. With the obvious burst in performance and the lowering of needed resources for doing so.

2.2 Interdomain QoS provisioning extensions

The interdomain scenario forced the research groups to change typically working QoS protocols. The reason was to adapt them to a new point of view given that different administrative domains can have different policies or different bandwidth availability.

On one hand there is a protocol (COPS) which tries to join the IntServ accuracy with the DiffServ scalability, that results in a protocol with low signalling overhead, with classes of service and admission control policies.

On the other hand there are extensions to the classical BGP routing protocol, which insert QoS capabilities to interdomain environments. Here we discuss QBGP and EQ-BGP [81].

2.2.1 Common Open Policy Service (COPS)

As stated on the Assured Forwarding description for the network to guarantee some QoS, it has to deploy some kind of Admission Control (as in IntServ). The reason is straight-forward, if there isn't any AC and there are one thousand flows of a gold class, the silver, which has, say 100 flows, will perform better because its queue will be unloaded, but the gold's will be dropping packets, despite of their theoretical better traffic guaranties.

For issuing admission control there are several options, one of the most used is COPS, defined at [39].

The Common Open Policy Service (COPS) protocol is a simple query and response protocol. It can be used to exchange policy information between a policy server (Policy Decision Point or PDP) and its clients (Policy Enforcement Points or PEPs). One example of a policy client is an RSVP router that must exercise policy-based admission control. At least one policy server exists in each controlled administrative domain. COPS protocol has a simple but extensible design. The main characteristics of the COPS protocol include:

- COPS employs a client/server model where the PEP sends requests, updates, and deletes to the remote PDP and the PDP returns decisions back to the PEP.
- COPS uses TCP as its transport protocol for reliable exchange of messages between policy clients and a server.
- COPS is extensible in the sense that it is designed to leverage self-identifying objects. It can support diverse client specific information without requiring modifications to the COPS protocol itself.
- COPS was created for the general administration, configuration, and enforcement of policies.
- COPS provides message level security for authentication, replay protection, and message integrity. COPS can also reuse existing protocols for security such as IPSEC or TLS to authenticate and secure the channel between the PEP and the PDP.
- COPS is stateful in two main aspects:

  - Request/Decision state is shared between client and server.
  - State from various events (Request/Decision pairs) may be inter-associated.

- Additionally, COPS is stateful since it allows the server to push configuration information to the client, and then allows the server to remove such state from the client when it is no longer applicable.

All the above functionality is accomplished by a very efficient signalling protocol which mixes the advantages of both DiffServ and IntServ. The entity in charge of managing all this is called Bandwidth Broker (which in fact is the PDP).

2.2.2 Quality Border Gateway Protocol (QBGP)

The QBGP is an extension of BGP4 protocol that allows to include into exchanged BGP update messages the information about available network services (traffic classes) and the QoS level offered along the path. The first extension of BGP was originally proposed at [3]. This approach assumes that BGP update messages include additional parameter, called TE (Traffic Engineering) weight. This parameter can express available bandwidth, the number of hops, maximum delay, etc. The value of TE weight may improve the routing decisions performed by particular domains.

Other proposed extensions to BGP were done, by adding attributes such as QoS_NLRI (Quality of Service Network Layer Reachability Information) that allows to exchange between ASs QoS related information, like:

- Packet rate (reserved, available).
- One-way delay metric (minimum, maximum, average).
- Inter-packet delay variation.
- Loss rate.
- PHB Identifier.

This information is used for performing routing decision and then it is advertised to the neighboring QBGP routers.

Several work groups have considered different approaches to QBGP. The most important being:

- *CoS capability:* used at the TEQUILA project [122] and renamed at MESCAL project [86] as *MC identifier*. It uses QBGP to exchange only the information about available Classes of Service (Meta Classes). In this approach the created path includes information about supported classes of services, but there is no detailed information on QoS experienced on paths.

- *QoS parameter:* used at TEQUILA and again renamed at MESCAL as *QoS information* based that uses QBGP to exchange detail values of QoS parameters offered by a given traffic class in particular domain. As a result QBGP provides information about the value of QoS parameters experienced on particular path. Moreover, in MESCAL project, the *QoS based approach* was further enhanced by measurements of the actual offered QoS level.

2.2.3 EQ-BGP

The EQ-BGP [81] protocol was developed within the EuQoS project [45] with the aim of performing interdomain QoS Routing (QoSR). The objectives of EQ-BGP are to advertise and select the routing paths for the different CoSs. The considered CoS are same as we introduced in Table 2.3. EQ-BGP extends the BGP-4 routing protocol in the following way.

First, EQ-BGP includes an optional path attribute that considers information about the QoS capabilities of a path. Second, it includes a QoS assembling function for computing aggregated values of the QoS parameters for the whole routing path. This assembling function supplies the sum of the delays for each segment of a path or the minimum available bandwidth along a path. Third, EQ-BGP has a QoS-aware decision process for selecting the best end-to-end path for the different CoSs. And fourth, EQ-BGP handles multiple routing tables in order to store the available paths for each end-to-end CoS.

EQ-BGP performs QoS routing in multidomain networks by taking into account both intra and interdomain QoS information. For that purpose, each EQ-BGP router advertises to its neighbours the reachable destination addresses including information about the available end-to-end CoSs. On that basis, each EQ-BGP router selects the best QoS path for each end-to-end CoS and informs its neighbours about its choice. Thus, EQ-BGP sets the roadmap for the available QoS paths between each pair of source and destination networks. These paths are called end-to-end QoS paths and they are computed and advertised by EQ-BGP routers for each CoS separately.

2.3 Quality of Experience

QoS has been used as a mechanism to provide network level guaranties. But given the increase of different services on the network this concept has evolved into Quality of Experience (QoE). QoE is a subjective measure of the user's satisfaction of a given service. In ITU Recommendation G.1080 and G.1081 (still in draft mode) QoE is defined as: *The overall acceptability of an application or service, as perceived subjectively by the end-user*. They define the requirements in a network independent fashion. The restrictions are specified in terms of video, audio, graphics or end-to-end communication and how these services affect the network transport behaviour.

In our work we will consider the *service* only as a *network* service, usually meaning a real-time communication such as a VoIP conversation.

QoE is related to QoS but it differs in some points: where QoS refers to the network *point of view* of quality by objectively measuring its behaviour. QoE represents the higher layer, human perception, and thus subjective measures.

These differences between QoS and QoE have a number of implications:

1. New point of view about *what*, *why* and *how* to measure.

2. Totally different set of metrics:

   - Objective.
   - Subjective.

3. Different targets in order to define the different metrics and uses of the results.

The new point of view implies that now we will need to monitor different set of properties: audio/video codecs, quality of the speakers, mood of the users, etc.

As it can be noted the amount and diversity of the parameters makes of QoE a very complex issue.

As pointed out before, QoE is subjective by nature, because each user can have radically different opinions about the quality of a communication. Eventhough there are several efforts invested on *"objectivising"* the QoE, in terms such as *Mean Opinion Score*, which we detail in the next chapter.

Finally QoE is related to QoS in the sense that depending on the desired QoE for an event, this will be converted to specific QoS policies and network behaviour. Nevertheless, this mapping is still under development by the competent organisations (e.g. ITU-T).

3

Network Measurements

We have seen that Quality of Service is necessary, but the deployment of a QoS enabled network with custom policies needs an extra component. It is the actual assessment that the quality is properly delivered.

From the network point of view, this implies that specific network metrics (e.g. One-Way Delays, Packet Loss Ratio, etc.) are delivered within bounded thresholds. The network has to guarantee those parameters and, take the necessary actions when they are not being provided. Actions to enforce QoS contracts are, for example, early dropping *best-effort* traffic or forcing routing changes using traffic engineering [47].

Hence, there is a tight relation between QoS provisioning and network status. Classically, such network status is measured in terms of *Metrics*.

Metrics can be computed using different techniques, all wrapped within the umbrella of network measurements. In this area lots of research efforts have been invested, basically in efficient ways of traffic collection, effective methods for active traffic generation with various goals, such as network metric acquisition or topology discovery.

In the framework of network measurements, there are two main tendencies, active and passive traffic analysis; the first one promotes the controlled injection of packets to the network, and analysing its behaviour at the destination point. The focus relies then in the differences between what has actually happened and what should have happened on an ideal situation.

Passive measurements, on the other hand, perform the analysis of already existent traffic, capturing it on the fly in any intermediate point between the source and the destination. This second approach takes advantage of analysing real traffic and not synthetic flows created by some application.

As stated on [127], having a good network measurement methodology is good for:

1. *Network Troubleshooting:* by analysing the network it is possible to pin point the sources and the reasons of many kinds of anomalies.

2. *Protocol Debugging:* controlled traffic generation can help to display protocol deficiencies or misbehaviours under heavy load, which are difficult to debug in a production network.

3. *Workload Characterisation:* by statistical traffic generation is possible to develop more reliable network topologies or transport protocols.

4. *Performance evaluation:* by injecting controlled traffic on the network or by analysing already existent flows, it is possible to assess if the network can comply with the contracted SLAs.

Before discussing about Active and Passive traffic measurements, it is important to gather some knowledge about the *Metrics* we can compute with these mechanisms. In the next section we present the most relevant metrics, focusing on the ones related with SLA assessment.

3.1 Network Metrics

On a networking environment there is a huge amount of parameters to be measured. In the effort of standardising such parameters has been carried by two major organisations: International Telecommunication Union (ITU) [58] and IP Performance Metrics (IPPM) [1]. This section describes the definition given by both organisations about the more common metrics.

3.1.1 IP Performance Metrics (IPPM)

IP Performance Metrics (IPPM) is a Working Group of the Internet Engineering Task Force (IETF) [56], their task is to develop a solid criteria for defining all the concepts related to performance metrics, hence the name of the working group.

This definition work has the aim of developing a set of standard metrics which could be applied to measure the quality, the performance and the reliability of Internet communications. The metrics are not focused on providing a subjective result but objective analysis, which will highlight an unbiased quantitative measure of performance.

IPPM in the specification of its metrics defines some basic concepts required to understand them:

- *Wire time:* Wire-time is the time that takes a packet from the delivery of its first bit on the physical wire (out of the Network Interface Card (NIC)) until the reception of the last bit on the other side.

- *Type-P packet:* Given a set of packets, a Type-P packet is a packet that complies with certain specified condition, the set of Type-P packets are the chosen for computing the desired performance metrics, Type-P packet selection can be left as general as desired or, on the other hand, it is possible to narrow the condition to match only a few set of packets in the whole measurement as stated on [96], where a framework for generic metrics notation is introduced.

Following sections will describe the set of important measurements stated by the IPPM WG. All this section supposes that the packet sent is a Type-P packet unless noted differently.

3.1.1.1 Connectivity

Connectivity measurement is defined on [79], and its aim is to define whether there is connectivity between a source and a destination or not, either in one way, from source to destination, or both ways.

Connectivity is a property that depends on an instant `T`, where such connectivity has to be tested, so the Unidirectional connectivity is accomplished when at a time `T` a packet send from the source (A), can reach its destination (B).

With this definition, defining Bidirectional connectivity is almost trivial: When there is unidirectional connectivity from a point *A* to a point *B* and from the point B to the first one, then the bidirectional connectivity is achieved.

This metrics are called *Instantaneous One-Way/Two-Way connectivity* respectively.

Mathematical remarks: Connectivity is considered as a boolean variable on a given instant (`T`) for a given connection.

This arises the need to determine connectivity during an interval, so a new metric is defined from the previous one: *One-Way/Two-Way Connectivity* (note the absence of the word Instantaneous). Where the connectivity is checked on an instant `T` during a `dT` interval.

3.1.1.2 One Way Delay

One Way Delay, as stated on [5] is:

> For a real number dT, *"the *Type-P-One-way-Delay* from Src to Dst at T is dT"*. This means that Src sent the first bit of a Type-P packet to Dst at wire-time T and that Dst received the last bit of that packet at *wire-time* $T + dT$.

Which means that the OWD is the time passed from the first bit of the packet sent to the network to the last bit reached its destination.

An assumption usually taken on this metric is to compute the End to End delay, which includes the Wire-time as stated before, but also the overhead imposed by the application's packet generation time and its processing time at destination.

Instantaneous One Way Delay (OWD): If there is a packet flow from source to destination, the definition of OWD is not complete, given that it only assumes one packet. Thus, the Instantaneous One Way Delay is the OWD of each packet which belongs to a set of Type-P packets taken individually. The result shows the time evolution of the delay each individual packet has to travel towards its destination.

By default when speaking about OWD calculation, instantaneous one way delay is supposed.

Mathematical remarks: Mathematically, the One-Way Delay is a strict positive real value. It is computed by subtracting the sender's timestamp to the reception time as shown on equation 3.1 where N is the number of sent packets for the tests.

$$OWD_i = T_{reception_i} - T_{sender_i} \qquad 1 \leq i \leq N \tag{3.1}$$

In order to have an accurate OWD it is mandatory that both endpoints, Source and Destination, must be synchronised (i.e. have the system clocks marking the same hour at some instant), as will be described on section 4.1.

3.1.1.3 One-Way Packet Loss

One-Way packet loss over a Type-P packet, is the condition whether a packet was received on a time T from a source to a destination.

It is computed in a one way fashion for different reasons as defined on [6]:

1. *Asymmetry over a path:* Not always the response to a packet is sent over the same path back to the origin. So there is no relation of packet losses on the flow.

2. *One-Way traffic:* Real time applications or UDP traffic are unidirectional.

It is important to remark that a corrupted packet is also considered a lost packet even if it reaches its destination. Key difference with the ITU's recommendation, where a packet loss and an errored packet are considered different.

Mathematical remarks: Under the point of view of IPPM, the One-Way Packet Loss is a boolean metric, where 0 means that the packet has been received properly, and 1 when the packet has been lost.

If considered over a dT time interval, there is the packet loss ratio, computed as:

$$PLR_{dT} = \frac{\#Loss}{\#Total} \tag{3.2}$$

Usually this is the used metric regarding packet losses to assess the network quality.

3.1.1.4 Round-trip delay and Loss

Despite the inconveniences presented before, *Instantaneous Round-Trip delay* usually applies to the great majority of Internet flows because of its bidirectionality.

As defined on [7]:

> *Instantaneous Round Trip Delay* of duration `dT` is the time passed since a source emits a packet to a destination, until this destination answers back by emitting a response packet as soon as possible.

The definition and use of this metric is much easier over OWD, where there is no need to synchronise the system clocks of the involved hosts, the Source (Emitting entity) and the Final Destination (Receiver of the response) are actually the same machine.

By default when referring to *Round-trip Delay* it has to be understood as *Instantaneous Round-trip Delay*.

Mathematical Remarks: Just as the *One-Way Delay*, the value obtained from the previous definition is a strictly positive real number pointing to an *Instantaneous Round-trip Delay*.

If there is a sample of packets where the Round-trip has to be computed the applied formula is the one shown on 3.3, which is the same as for the *One-Way Delay* with the difference that the timestamps are taken from the source machine only.

$$RTD_i = T_{received_i} - T_{sent_i} \qquad 1 \leq i \leq N \tag{3.3}$$

Packet Losses: A packet is considered Lost on the *Round-Trip Delay* either when the packet emitted by the source does not reach its destination, or when the response is lost.

Another possible situation is when the delay obtained is greater than a predefined threshold (usually 255 because of the TTL).

3.1.1.5 IP Delay Variation

IP Delay Variation as stated on [34] is:

> The variation in packet delay is sometimes called ”jitter”. This term, however, causes confusion because it is used in different ways by different groups of people.
>
> ”Jitter” commonly has two meanings: The first meaning is the variation of a signal with respect to some clock signal, where the arrival time of the signal is expected to coincide with the arrival of the clock signal. This meaning is used with reference to synchronous signals and might be used to measure the quality of circuit emulation. There is also a metric called ”wander” used in this context.
>
> The second meaning has to do with the variation of a metric (e.g. delay) with respect to some reference metric (e.g. average delay or minimum delay). This meaning is frequently used by computer scientists and frequently (but not always) refers to variation in delay.
>
> **Definition 1.** *IP Packet Delay Variation (IPDV) defined for a selected pair of packets in the stream going from measurement point MP_1 to measurement point MP_2.*
>
> *The IPDV is the difference between the one-way-delay of the selected packets.*

Instantaneous IP Delay Variation (IPDV): When dealing with Instantaneous IPDV, instead of arbitrarily selecting a pair of packets, we will consider two consecutive packets of the flow. In the case of any packet losses the computation of IPDV will be avoided.

As on the OWD computation, when speaking of IPDV in general, the computation of Instantaneous IPDV is supposed.

Mathematical remarks: Once the selection algorithm is chosen, the formula used for the Instantaneous IPDV can be seen on the equation 3.4.

$$IPDV_i = OWD_{i-1} - OWD_i \qquad 1 \leq i \leq N \tag{3.4}$$

IPDV as stated on the above equation can be any real number, either positive or negative. In this case the clocks synchronisation is not necessary per se. The only requirement is that the rate of both clocks must be the same, this is known as skew as will be discussed throughoutly in Section 4.1.

3.1.1.6 Bulk Transport Capacity

Bulk Transport Capacity (BTC) as defined on [82] is the maximum net transport capacity of a link using a single congestion aware protocol (i.e. TCP) connection. This highlights the maximum theoretical capacity of the link which connects two different end points.

Is important to notice that the result (in bits per second - bps) refers to the net throughput, where all headers, retransmissions or losses are discounted of the overall performance.

This metric gives an idea of the maximum performance from an user point of view where big data transfers are involved (FTP, big HTTP downloads...).

Mathematical remarks: As stated above the unit for measuring the BTC are the bps, as in Kbps (Kilobit per second $10^3 bps$), Mbps (Megabit per second $10^6 bps$), Gbps (Gigabit per second $10^9 bps$), etc. The final bandwidth is computed with equation 3.5 which refers to the whole transmission. In the equation $data\_sent$ refers to the net number of bits transmitted, and $time\_taken$ is the time in seconds since the start of the transfer until the end.

$$BTC = \frac{data\_sent}{time\_taken} \tag{3.5}$$

3.1.1.7 Loss patterns and Loss Bursts

Packet losses reflect directly the quality of the network, but more importantly different loss patterns could bring different effects to the user, even with the same Packet Loss Ratio (PLR).

The definition of this metric can be found at [71]. This RFC defines two different metrics:

1. *Loss Distance:* given a set on packets which are numbered in a sequence order, the loss distance is the amount of consecutive lost packets.

2. *Loss Period:* indicates the starting point and the duration of a lossy period. The duration can be specified both in terms of number of packets or time interval.

Bursty losses degrade the quality to a higher degree than sporadic packet losses, since application's correcting algorithms do not work in such situations [59]. In [17] and in [71] the authors define a loss burst as the sequence of consecutive lost packets. With this definition it is possible to detect lossy periods efficiently, but in a congested environment many of those loss periods can be chained together, with few successfully transmitted packets, forming a longer time period with poor network conditions, this behaviour is not considered on the metric, but due to its importance and practical applicability, we improve this burst definition by inserting density concepts to the classic definition in Section 8.1, as part of contribution of this work.

3.1.1.8 Link Bandwidth Capacity

Link Bandwidth Capacity (LBC) is also known as Available Bandwidth (AB). There is no RFC defining this metric yet. However, there is much related work that studies it, for example [94], [120].

As opposite to BTC, Link Bandwidth Capacity (LBC) estimates the available bandwidth between two end points, in this case, the metric obtained only measures the maximum theoretical bandwidth, without congestion control and without any additional information.

The main problem with this metric is that it can be highly variable given that it depends on the actual use of the network. Nevertheless, not always is possible to overload the network for computing its BTC. Therefore LBC is the most common alternative when estimating the Available Bandwidth on a network. More discussion about this topic will be done in section 3.2.2.

3.1.2 ITU

On the other side of IPPM's practical approach towards standardising QoS metrics, ITU with a broader scope on their recommendations specify similar parameters on a slightly different way, taking into account a more general description of the metrics.

By default all the ITU metrics refer only to one way traffic, or at least, to the unidirectional part of a bidirectional flow.

This section is devoted to the explanation of such different views of the same concepts. As before, some previous definitions are necessary for understanding ITU's specifications (defined in [64]):

- *Measurement point (MP):* point between source and destination where any performance events can be observed and measured.
- *Section:* part bounded by MP. The most important sections are:
  - **Basic sections:** any group of sections between source (SRC) and destination (DST).
  - **End-to-end IP network:** MP located either on SRC and DST.
- *Transfer Event:* IP Packet transfer events occur when an IP packet crosses an MP, is verified that it is valid and the source and destination addresses are correct.

ITU states that for measuring the performance metrics, it has to be done necessary using the IP packet transfers, this describes the status of the transfer. ITU defines four IP packet transfer outcomes for packet delivery. They can be:

- *successfully transferred:* packet reaches its destination successfully.
- *errored:* corrupted header or errored payload arrived at DST.
- *lost:* the IP packet never reaches its destination.
- *spurious:* an IP packet arrived successfully at its destination but SRC didn't send it. This effect is also know as spoofing.

Of all the IP packets on a link, the **population of interest** is the set of IP packets which will be under study (analogous to the IPPM's Type-P packet). In the end-to-end case, usually this will mean, all the packets going from SRC to DST.

3.1.2.1 IP packet transfer delay (IPTD)

Is defined for all successful and errored packet outcomes across a basic section. IPTD is $(t_2 - t_1)$ between the egress event at t_2 and the ingress event at time t_1. In this environment $(t_2 > t_1)$. This definition is analogous to One Way Delay defined previously.

Mean IP packet transfer delay: is the arithmetic average of the delays of the population of interest.

Outliers on IP packet transfer delay: ITU's outlier definition considers packets out of the $99.9th$ percentile as outliers which cannot be considered for the analysis. In some specific situations one can consider $95th$ percentiles.

In the case of outliers for the minimum case they are often not considered since the hard minimum is determined by physical constraints. Nevertheless sometimes the $0.1th$ percentile or the $5th$ percentile can be used.

This specific definition for outliers is one of the main differences with IPPM, where the metrics are considered per Type-P packet, leaving open the outlier or the statistical operations over the metrics to the user.

3.1.2.2 IP Packet Delay Variation (IPDV)

Analogous to IPDV defined by IPPM, the main difference is that it is composed by a single value per time interval. It is computed using:

$$IPDV = IPTD_{upper} - IPTD_{min} \tag{3.6}$$

Where $IPTD_{upper}$ is the $1 - 10^{-3}$ quantile of the OWD, and $IPTD_{min}$ is the minimum OWD value of the measurement. This permits the detection of congestion in the network, analysis of the TCP window behaviour, and effects on IPDV of routing updates.

In QoS environments IPDV limits the lower bounds of the reception buffers. This metric is important both for the interactivity (small buffers) and the quality (late packets are considered as lost) of the communication. ITU-T defines an upper bound of $50ms$ for the real-time classes.

End-to-end 2-point IP packet delay variation: Given two MPs the IP packet delay variation is the delay of a given packet (x_k) and a defined reference IP packet delay between the same MPs, which is the first packet from SRC to DST of the population of interest.

Variations on the above description can be considered, either taking the reference IP packet as the minimum or the maximum absolute value of the set of IP packets.

3.1.2.3 IP packet error ratio (IPER)

IP packet error ratio is the ratio of total errored IP packet transfer outcomes to the total of successful IP packet transfer in a population of interest. There is no equivalent for this definition in IPPM's definitions.

During this thesis we will consider the IPPM approach, that is to assume that errored packets are lost.

3.1.2.4 IP packet loss ratio (IPRL)

IP packet loss ratio is the ratio of total lost IP packet transfer outcomes to the total of successful IP packet transfer outcomes in a population of interest. To compute this metric we need a time interval (t_1, t_2) and knowledge about the transmitted and received packets. Then it is computed by using equation 3.2, where dT in ITU's nomenclature is the time interval (t_1, t_2).

3.1.2.5 IP packet severe loss block ratio (IPSLBR)

Convenience metric, extracted from IPRL, it defines a period of time with unusually high packet losses. Tentatively the threshold to consider IPSLBR is when a network has more than 20% IPRL in a one minute interval. Rendering any service carried by the network unusable.

Not directly, but with the Burst definition we issue in Section 8.1 we use similar concepts as IPSLBR.

3.1.2.6 Spurious IP packet rate

Spurious IP packet rate is the number of spurious IP packets found on a MP divided by the time interval of the observation.

3.1.3 Subjective Metrics

In opposition to the objective metrics studied so far, the user level metrics are related to QoS level experienced by the user (QoE). It is also called Perceived QoS (PQoS). These metrics refer to the overall quality experienced by users. This quality is influenced by QoS assured at the network level, the quality of voice/video codecs, the effectiveness of supporting mechanisms in applications such as playback buffer, codec rate adapters, etc., as well as the quality provided at the call level.

In this work we focus on the Mean Opinion Score (MOS) which is the metric used to assess the quality of voice transmission. According to ITU-T [61] MOS is defined in the following way:

> The mean of opinion scores, that is the values on predefined scale, that subjects assign to their opinion on the performance of the telephone transmission system used either for conversation, or for listening to spoken material.

Originally, the MOS metric was only evaluated in a subjective way by the group of users, who mark the quality of a voice transmission in the scale from 5 to 1, where the following scores are used: 5=Excellent, 4=Good, 3=Fair, 2=Poor, 1=Bad quality. However, apart from subjective approach, the MOS value can also be evaluated based on objective models or estimated models, see [61], [121]. As a consequence, we distinguish different types of MOS metrics as presented in Table 3.1, where LQ refers to Listening Quality, CQ to Conversational Quality, S to Subjective, O to Objective and E to Estimated.

	Listening-only	Conversational
Subjective	MOS-LQS	MOS-CQS
Objective	MOS-LQO	MOS-CQO
Estimated	MOS-LQE	MOS-CQE

Table 3.1: Types of MOS metric

In subjective methods, we calculate the MOS value as an arithmetic mean of subjective judgements of a group of users. The objective methods are performed, based on an objective model that allows assessing the perceived quality. For example by comparison of sent and received test signals. The estimated method allows calculating the MOS value based on a model of a communication system (e.g. E-model as proposed in [59]). Since this approach does not require transmission of voice signals, it can be used for the network planning.

In Table 3.2 we present the exemplary MOS values for typical voice codecs. The MOS was measured in the reference network, where no packet transfer delay, delay variation nor packet losses were introduced. Note that even under such ideal network conditions, non of the codes offer MOS equal to 5. This is caused by quality degradation introduced by the compression algorithms.

Standard	Codec Type	Rate[Kbps]	Frame[ms]	Lookahead[ms]	MOS value
G.711	PCM	64	-	0	4.43
G.729	CS-ACELP	8	10	5	4.18
G.723.1	ACELP	5.3	30	7.5	3.83
G.723.1	MP-MLQ	6.3	30	7.5	4.00

Table 3.2: MOS values for typical voice codecs

The MOS metric can also be used for evaluation of video quality. Its definition is similar to voice assessment. More details can be found in [63].

3.1.3.1 Subjective Assessment Method

The ITU-T in [61] recommends two methods for the subjective assessment of MOS that correspond to conversation-opinion tests and listening-opinion tests.

Conversation-opinion tests require participation of a number of user pairs, who take part in a normal conversation using voice or videoconference application. Every conversation should be purposeful and should have a natural beginning and a natural ending. The conversation must never be terminated in the middle of the test. After the test, each participant scores the perceived quality in the MOS scale. Conversation tests are intended to reproduce in the laboratory environment the conditions as close as possible to real-life.

Listening-opinion tests require participation of a speaker and a number of listeners. The speaker reads a list of previously prepared sentences or phrases, while the listeners only listen and give opinions about the perceived voice quality. The speech material should consist of simple, meaningful and short sentences, chosen at random as being easy to understand. These sentences should be made up into lists in random order in such a way that there is no obvious connection of meaning between one sentence and the next. Very short and very long sentences should be avoided. The aim is that each sentence when spoken should fit into a time-slot of $2-3$ seconds. The sentences are organised in lists and after each list the listeners scores the perceived quality in the MOS scale.

Another approach for MOS evaluation is based on the evaluation of logatom articulation that reflects the intelligibility of transmitted voice signals. The logatoms are meaningless words that are composed of a group of phonemes that are characteristic for a given language. In this test a speaker reads a list of logatoms, while listeners write them down exactly as they hear them. After the test, the lists written by the listeners are compared with the original one and then MOS value is evaluated based on the percentage of correctly written down logatoms. Note that logatoms that have similar pronunciation, but were written in different way should be treated as correct. The logatom articulation test allows for eliminating the influence of human capabilities for guessing the meaning of words from the sentence context.

3.1.3.2 Objective Assessment Method

The objective methods are aimed at assessing the MOS value without participation of users. For this purpose specialised models are defined that allow for the estimation of the perceived quality. This is based on the comparison of originally transmitted voice or video signal with the received one, but it can also be based on detailed characteristics of terminals and the network. Below we briefly present two objective methods that are standardised by ITU-T.

Perceptual evaluation of speech quality (PESQ) method [62] was designed to estimate the MOS value in objective way. It requires transmission of the reference test voice signals that should be collected at the receiver side. Then, the original and the received voice signals are compared with the special PESQ algorithm. The key to this process is transformation of both the original and received signals into an internal representation that is analogous to the psychophysical representation of audio signals in the human auditory system, taking account of perceptual frequency and loudness. The internal representation is processed to consider such effects as local gain variations and linear filtering that may – if they are not too severe – have little perceptual significance. Finally, the PESQ algorithm compares both signals and provides the MOS value on that basis. It should be noted that the PESQ does not provide a comprehensive evaluation of transmission quality. It only measures the effects of one-way speech distortion, such as degradation introduced by voice codes, channel errors, packet losses, delay variation, time wrapping of audio signals and noise on speech quality. On the other hand, the effects of loudness loss, delay, sidetone, echo, and other impairments related to two ways interaction are not reflected in the PESQ method. As a consequence, it is possible to have high PESQ scores, yet poor quality of the connection overall. The PESQ method can be applied for

waveform codecs such as G.711; G.726; G.727 and for CELP and hybrid codecs e.g. G.728, G.729, G.723.1.

E-model method [59] was designed to estimate the value of MOS based on the set of parameters that represent the terminal, network and environmental quality factors. More precisely, it allows the computation of the MOS value based on 20 input parameters such as: voice codec distortion, quantisation, echo, room noise, SNR, loudness, sidetone, delay, losses factors, as well as the advantage factor reflecting human psychological aspects. Compared to other methods, the E-model does not require transmission of voice signals through the system under test. As a consequence, the E-model can be used as a network planning tool, that allows network designers to estimate the QoS level perceived by the users. It should be noted that E-model can only be used in case of telephony handsets that carry narrow band voice signals (300-3400Hz).

The important issue is that the E-model has not been fully verified by field surveys or laboratory tests for the very large number of possible combinations of input parameters. For many combinations of high importance to transmission planners, the E-model can be used with confidence, but for other parameter combinations, E-model predictions have been questioned and are currently under study.

However, since the E-model is the "de facto" standard, it is the one we use in Chapter 9 for the QoE evaluation.

3.2 Active Measurement

Active measurement, is a network testing technique that is based on the synthetic insertion of controlled traffic on the network and its reception at the destination. One of the goals of such testing is to measure End-to-End network parameters, which range from QoS performance evaluation to routing algorithm analysis.

Active measurements permit a huge amount of possible actions to perform over the network. This section will be mostly focused on describing the controlled traffic injection for QoS analysis, with some overview of other possible uses of the methodology for completeness.

An important issue to consider on Active Measurement, in Passive Measurement as well, is the accuracy of the measurement. It depends, in active testing, on the precise generation of the probe packets and the accurate reception at destination. All the important accuracy concerns will be explained along with the methods depicted here.

3.2.1 One-Way testing (UDP)

In the environment of Active Measurement is possible to do lots of different classifications depending on the characteristics of the generated traffic. This section overviews several of the most important techniques and uses of unidirectional flow generation.

In general when one-way flows are generated it is done through UDP packets, which tend to emulate real time applications or to compute one way metrics from the network in a predetermined way.

First issue to consider, when dealing with active measuring, is the definition how the packets will be generated and injected on the network. Depending on the generated pattern of the flows the results may vary significantly.

Most common one-way patters are:

Periodic flows: periodic flows define a packet rate and a packet size, both parameters are fixed during the tests. Usually this flows are used for one way bandwidth calculation [100] or QoS metric computation [22].

The usefulness of such flows relies on the fact that the generation is deterministic and known beforehand. Thus the variation of the metrics shown at section 3.1 at the reception point are due to the underlying network with small interference at the end points.

Poissonian flows: they define a poissonian function for packet generation. Such flows are inherited from the telephony networks where the inter-call arrival time can be modeled with that function.

Due to its unpredictable nature, Poissonian flows are often used to test application buffers or application behaviour under network stress.

Synthetic Real Traffic: There are some efforts to generate *"Internet like"* traffic. This is a complex issue because it implies modelling the Internet behaviour. Some efforts have been done in defining a model which can reproduce most internet traffic, usually these models are based on the Gamma-Farima distribution.

Specially crafted patterns: there are times when a simple pattern is not enough for the needed measurements. Thus, the generated traffic sometimes does not comply with any predefined pattern, but to a general form which helps the researcher to reach his/her goal. Examples

where this technology is applied is at [90], where special packets are sent for detecting broken links on the path.

Independently of the used patterns the most important part is the usage of the obtained results. There are too much work on this area to explain it here. That's why the discussion will be focused on the most important usages for this thesis:

- *QoS Parameters:* as shown in [22], with the One-Way Delay, the IPDV and the packet losses, is possible to analyse whether the network can provide the desired QoS or not.
- *Application Bottlenecks:* as a different research topic. Traffic generation can help profiling applications in order to detect weak points. In this direction, our work in [112] we studied the behaviour of the linux kernel's IP stack under various degree of stress. This is done by generating synthetic flows in increasing intensity until packet losses occur due to bottlenecks on the kernel's code.

3.2.2 Bidirectional testing

Opposite to unidirectional testing, there is always the need to reproduce traffic which resembles actual Internet traffic, in this trend, there is mandatory to test the network with bidirectional flows, generally using TCP.

The goal of bidirectional testing is much broader that the one accomplished by unidirectional flows. Usually TCP traffic has much more possible parameters to tests than a simple UDP flow.

The major problems found are related to the protocol itself, when there is need of deterministic results, TCP introduces a lot of variations to the traffic (mainly because of adaptation algorithms).

In bidirectional flows, the real control over the traffic is handled by the protocol itself. This excludes most of the traffic patterns seen on the previous section.

The use of traffic patterns, usually for computing one way delay metrics, along with bidirectional flows is not useful. In this environment, though, other computations can take place, like Round Trip Time, link bandwidth, response time, etc. Other studies are also possible, for example about real traffic behaviour on the Internet for data transfers or protocol analysis.

The paradigm defined here is more close to set up an environment and generate a bidirectional flow with the aim of:

Connectivity Verification: in this case usually the generated traffic is ICMP, and the goal is to send an *Echo Request* [31] to be responded by an *Echo Reply* in the case of the packet reaching the destination.

Hop count: another possible test is to know the number of hops that separate two endpoints, in this case the procedure is to generate packets with a TTL starting at 1 and incrementing it each round. This approach permits to receive the *TTL Expired* ICMP packet from the router who discarded the packet.

Geographical Localisation: on a more complex approach, there is work towards localising hosts geographically from the RTT (Round Trip Time) delays of specially crafted packets, for example in [129] the authors present a methodology to infer the host geographical location from well-known landmarks and delay models.

Protocol Analysis: this study can be done also in unidirectional flows, but if a new protocol or application is developed (say, a new HTTP server) and the goal is to compute its performance, then the important part is to see the final user's impact. Examples of that can be found at [72], where the authors perform an analysis of the HTTP protocol over real servers on the Internet.

Available Bandwidth estimation: The Available Bandwidth (AB) of an end-to-end path is its remaining capacity, that is, the amount of traffic that can be sent along the path without congesting it. Recently, the area of end-to-end AB estimation has attracted considerable interest. Mainly because the AB is an important metric for several applications such as overlay networks, dynamic server selection, or inter-domain path monitoring. As a result, several estimation techniques and tools based on active measurements have been developed.

Most of the proposed tools designed to estimate the AB fall into two categories: the *Probe Rate Model* (PRM) and the *Probe Gap Model* (PGM). The first model uses packet trains and it is based on the concept of self-induced congestion. Informally, if one sends a packet train at a rate lower than the AB along the path, then the arrival rate of the packet train at the receiver will match the rate at the sender. However if the sending rate is greater or equal than the AB then the packet train will congest the queues along the path and the receiving rate will be lower than the sending rate. Tools such as Delphy [102], TOPP [85], PathLoad [36], IGI/PTR [55], pathChirp

[103], BART [41] and AB [25] use this model. The second model (PGM) uses packet pairs and bases its estimation on the differences of input and output time gaps of the packet pairs [120].

Routing Analysis: another important research done with bidirectional traffic is to analyse the impact on the overall network performance of the routing algorithms. There are lots of groups working on this subject, for example at [78] controlled packets are sent for detecting changes on the routes and their impact on the available bandwidth from an end-to-end perspective.

3.2.3 One Way Active Measurement Protocol (OWAMP)

Despite that this protocol belongs to the IPPM defined metrics, it is important enough in active measurements for separating it on a section of its own. This protocol is defined at [116]. The goal of this RFC is to provide good and reliable one way metrics framework. The way of accomplishing that is by defining a protocol that controls the progress of the tests, manages the tests sessions, and determines if a test has been successful.

All this control is needed because, in current tests environment, there is no guarantee that the testing methodology is correct. Typically, there are synchronisation issues (explained at 4.1), or problems related to tests repetition and management, etc. This protocol has the aim of being a generic framework for active network testing.

OWAMP Protocol is used for active performance measuring of IP-networks by inserting UDP streams of test-packets to the IP-network. The goal is to obtain the travel time from source to destination for each test packet (one way delay).

A part from the UDP flows, there are in parallel TCP flows (transmitted through a different interface) which will be in charge to control the actual tests.

The protocol: The One-Way Active Measurement Protocol (OWAMP) consists of two inter related protocols: OWAMP-Control and OWAMP-test. Where OWAMP-Control is used to initiate, start and stop test sessions and fetch their results, while OWAMP-Test is used to exchange test packets between two measurement nodes.

For controlling and running the tests OWAMP defines several roles:

1. *Session-Sender:* the sending endpoint of an OWAMP-Test session.

2. *Session-Receiver:* the receiving endpoint of an OWAMP-Test session.

3. *Server:* an end system that manages OWAMP-Test sessions, is capable of configuring per-session state in session endpoints, and is capable of returning the results of a test session.

4. *Control-Client:* an end system that initiates requests for OWAMP-Test sessions, triggers the start of a set of sessions, and may trigger their termination.

5. *Fetch-Client:* an end system that initiates requests to fetch the results of completed OWAMP-Test sessions.

The tests will be done between the Session-Sender and the Session-Receiver, the protocol, handles all the storage of the test's parameters for future use. The possible defined parameters are defined in OWAMP-Control. It is designed to support the negotiation of: Sender and receiver addresses, Port numbers, Session start time, Session length, Test packet size, The mean Poisson sampling interval for the test stream, Per-hop behaviour (PHB), Encryption and Authentication for both test and control traffic.

This strict definition of which parameters are used and why permits deterministic results if the tests are to be repeated (single management unit for all the parameters), or for issuing good comparisons among different tests.

In this thesis we do not use directly OWAMP, but thanks to this structured testing methodology we can assure that our testing environment is correct, in terms such as synchronisation or management.

3.2.4 Active measurement platforms

When dealing with active measurement platforms, usually the testbed is a non realistic closed environment, where the tests are carried. There are several research projects that work to provide a broad analysis of the Internet from an active point of view. This discussion will focus on the most important at this moment.

Etomic: In order to visualise and to understand the dynamics of Internet its topology should be continuously monitored and the traffic of data packets should be measured with high temporal precision and good spatial resolution. This gives a nanosecond active measurement precision for inferring Internet's topology. For further information look up at [43].

DIMES: DIMES is a distributed scientific research project, aimed to study the structure and topology of the Internet, with the help of a volunteer community (similar in spirit to projects such as SETI@Home). For more information refer to [35]. Both DIMES and Etomic belong to a greater consortium called EVERGROW [46].

Test Traffic Measurement Service (TTM): The Test Traffic Measurement Service (TTM) measures key parameters of the connectivity between a site and other points on the Internet. TTM allows to comprehensively and continuously monitor the connectivity of a network running an application to other parts of the Internet. This technology was first presented by RIPE at [50].

Active Measurement Project (AMP): was initially promoted by NLANR, but since June 2006 it belongs to CAIDA. It focuses on site-to-site active measurements conducted between campuses connected by high performance networks. The data collected by AMP is proved to be a valuable resource for network analysis to study the network and derive performance models for various aspects of Internet traffic [9].

CAIDA: Cooperative Association for Internet Data Analysis is one of the most active research branches on both active and passive measurement platforms. The use of innovative technologies for extracting diverse information of the Internet are developed. Projects like Skitter (Internet Atlas Project), AS Mapping and much more fall on the Active measurement part of this consortium. For more information refer to [23].

PERFormance Service-Oriented Network monitoring ARchitecture: (PerfSONAR) is an infrastructure for network performance monitoring, making it easy to solve end-to-end performance problems on paths crossing several networks. It focuses on studying the capacity and availability of the links, specifically its perceived quality. PerfSONAR delivers a service oriented interface to monitor the network status [97].

3.3 Passive Measurement

The other main trend on traffic analysis is the Passive Measurement environment. As seen before, Active Measurements use an intrusive method for inferring network characteristics. But we cannot use it when analysing real traffic. With synthetic flows is very difficult to

simulate the network's Internet traffic. But, when the administrators are able to capture real traffic existing in an Internet link, the analysis could prove much more significant.

Traffic capturing has two main trends, software and hardware capturing. Both solutions are described below.

3.3.1 Software Capture

When the capture requirement is not very hard, in terms of amount of packets per second to be captured, there are several methods for actually capturing packets. Such methods, often, are related on the layer where the packet is captured:

Kernel Level packet captures: This low level captures are the most useful for strict network analysis. That's because the sooner the packet is captured (closer to the wire) the more accurate will be the information extracted, at least for network analysis.

At this category fall several technologies. The most important are:

1. *Direct kernel support:* the operating system has internal functionalities for filtering packets at kernel level. On example on this category is Netfilter [89] which is a Linux kernel functionality.

2. *BSD Packet Filter (BPF):* was first presented on [84]. In infrastructure BPF is a technology similar to the Direct kernel support, given that its support has to be built-in the actual kernel. The difference is that BPF is a broadly deployed specification, which is available on almost all current OS implementations.

Network Layer capture: this technique is not broadly used. It does not capture when the packets as close to the interface as possible. Instead, are captured when they arrive at upper layers. At this point, only packets destined to the capture host or, if the host is a gateway, packets which the next hop is the actual router are treated. Usually on this layer, the capture is only used for firewalling purposes, because the packets can be modified.

Application Level capturing: application level capture, almost always, is the same as Active Measurement, because the capture point is the actual application used for generating the results. This high layer analysis is often used to extract application dependent statistics, such as user perceived quality, response times, etc.

3.3.2 Hardware Capture

Software capture has the advantage of versatility, and a broad scope of possibilities. But, in nowadays networks the problem resides when there is a necessity of capturing the huge amount of traffic on high speed links (e.g. Gigabit Ethernet link) with thousands of simultaneous flows traversing the capture point. At this stage, is very important to have a really real-time packet capture and processing. For accomplishing this there are the hadware capture facilities. Which, do not have as much flexibility as the software approach, but enable wire speed capture for the network.

DAG: one of the most famous hardware capture platforms is DAG [42]. This technology is distributed by Endace and it has a hardware infrastructure which permits full link capture in real-time. This solution is only needed when collecting large amounts of data.

SCAMPI: more than a product SCAMPI [106] being an IST European project, is a research effort to build a scalable network adapter for high speed traffic collection. The device is specially designed to perform up to 10Gbps wire speed collection and tailored for monitoring applications.

3.3.3 Passive measurement platforms

As Passive measurements is concerned, there are several research groups who work to provide a broad analysis of the Internet from a passive point of view. This discussion will focus on the most important at this moment.

The problem with such infrastructures is the complexity to have access to the service provider information, therefore, the passive platforms are more difficult to deploy.

InterMON project focused on the data gathering, its automated access and database design for inter-domain QoS analysis [88]. This infrastructure aims at delivering human understandable information about traffic traces.

Passive Measurement Analysis (PMA): The goal of the *Passive Measurement Analysis* (PMA) project is to deliver new insights into the operation, behaviour, and health of the Internet, for the benefit of network users and operations. Passive header trace data provides the

means to study workload profiles for a number of strategically located measurement points in high-speed environments [99].

Lobster: As a successor of the SCAMPI project, *Lobster* is a pilot European infrastructure for accurate Internet traffic monitoring. Based on passive monitoring, their goal is to deploy an infrastructure that can efficiently monitor up to 10Gbps of traffic in order to detect security attacks, test the performance of network services [77].

CoralReef is the monitoring infrastructure proposed by CAIDA. It delivers a solid passive monitoring platform. Monitoring of optical networks is done with an optical splitter, which diverts a small fraction of the light from the optical fibre to the monitor device. Real-time monitoring support includes system network interfaces), FreeBSD drivers for Apptel POINT (OC12 and OC3 ATM) and FORE ATM (OC3 ATM) cards, and support for Linux drivers for Endace DAG (OC3 and OC12, POS and ATM) cards. The package also includes programming APIs for C and Perl, as well as applications for capture, analysis, and web report generation. The CoralReef software suite includes tools for analysis of traces collected by these type of monitors.

SMARTxAC: SMARTxAC aims to develop and deploy a passive measurement infrastructure and a real-time analysis system for high-speed links. Currently, SMARTxAC is being used for capturing and analysing the traffic of the Anella Científica (Scientific Ring). The Anella Científica is the name of the Catalan R&D Network, which is managed by CESCA and connects about 50 Universities and Research Centres in Catalonia. Further information available at [117].

3.4 SLA Assessment

SLA assessment is the action of verifying that the network is honouring the SLA as contracted in advance. Prior to describing the different SLA Assessment methods, it is in order to introduce the concept of *Service Level Agreement* (SLA).

SLA is a formally negotiated agreement existing between two different customers. In our context the customers are the *Internet Service Provider* (ISP) and the big corporate customers.

This formal agreement determines the permitted bounds on the delivered service to the customer, often specified by the *Service Level Specification* (SLS) [51]. Violation of such specifications may incurr in penalties for the violating part.

The most used metrics, coming from the old telecom days, for SLA are:

- *ABA (Abandon Rate)*: Percentage of calls abandoned while waiting to be answered.
- *ASA (Average Speed to Answer)*: Average time (usually in seconds) it takes for a call to be answered by the service desk.
- *TSF (Time Service Factor)*: Percentage of calls answered within a definite timeframe, e.g. 80% in 20 seconds.
- *FCR (First Call Resolution)*: Percentage of incoming calls that can be resolved without the use of a callback, or without having the caller call back the helpdesk to finish resolving the case.
- *TAT (Turn Around Time)*: Time taken to complete a certain task.

Due to the lack of standardisation for the packet switched networks, we assume during the whole thesis that the relevant SLA metrics used to assess the network quality are: *i)* One-Way Delay, *ii)* IP Delay Variation, and *iii)* Packet Loss Ratio. Specifically under the constraints specified by ITU-T in [65] as discussed previously.

SLA assessment has been a study of research previously on the literature, the efforts focus on efficient end-to-end estimation of the QoS metrics. All the proposed solutions use a different methodology with varying degree of success, some very accurate (e.g. [107; 130]), by using passive traffic collection, others more efficient (e.g. [52; 119]) but too coarse, by using an active traffic measurements approach. There is a third approach, namely InterMON [88], that has an hybrid approach, on one hand it uses proprietary active probing mechanisms and passive analysis of MIB found in the edge nodes of the network.

In this thesis we aim at designing a efficient and scalable distributed SLA assessment infrastructure. We accomplish this by following the passive traffic analysis approach. Since there are not other systems using the same approach in this section we discuss isolated efforts at assessing the SLA of the network even if they do not design a full-fledged solution but address specific problems.

3.4.1 Assessment with active measurements

The main issue of active traffic techniques is its intrusive nature and its questionable accuracy, usually subject to the type and rate of the generated probes.

Most of the network analysis mechanisms for SLA assessment fall in this category, we present here a brief overview of the most relevant work in this area.

In the field of distributed platforms for QoS validation, approaches solving the SLA assessment problem have been already studied before in work such as perfSONAR [97].

perfSONAR [16; 52] is an infrastructure for network performance monitoring, focusing on the easiness to solve end-to-end performance problems on paths crossing several networks. It focuses on studying the capacity and availability of the links. This is a main limitation of the system since its coarseness does not permit to set the study of SLA compliance to finer detailed analysis, for example CoS, or even per flow.

Moreover, perfSONAR is focused on service oriented performance, this gives a very good framework to have automated queries and responses to the system.

In the case of specific solutions for some aspects of SLA assessment, there are more contributions in this area. The most relevant work has been carried by Barford et al. in [12] where the authors highlight the limitations of packet loss estimation using active probes, respect to the ones found via SNMP in commercial routers. This work is continued by Sommers et al. in [118] where the authors improve the loss estimation of classical Poisson-modulated probing mechanisms by presenting *Badabing*, a dynamic active tool that improves accuracy depending on the resources used for the estimation. More recently, in [119] Sommers et al. gather together all the above work, and present *SLAm*, another active probing tool that implements innovative packet loss, delay and delay variation estimation techniques. In all this research, the authors stress the need of proper metric estimation in order to lead to correct SLA assessment.

3.4.2 Assessment with passive measurements

As discussed before, fully deployable infrastructures for SLA assessment using passive traffic analysis are hard to implement. Mainly they need a powerful, often distributed, framework to collect traffic, manage and analyse the results, with the associated cost and complexity. Classically on the literature [38; 133], the passive analysis of existing traffic is performed very often in the edges of the network, this has a very important scalability problem over the globally

needed resources by the different collection points. Both in terms of metric reporting and traffic collection on high speed links.

Zseby et al. in [130] propose a method for efficient distributed flow and packet identification, this knowledge is very useful in order to design distributed QoS assessment systems.

When passively analysing traffic for SLA assessment in various locations we first must decide a common policy in order to capture the same set of packets on each collection point. The next step is to gather packet information relevant for the metric computation: flow where the packet belongs, the reception timestamp, etc. Then, to compute the network metrics this data must be shared by the other measurement points in a central unit, which will extract the actual metrics. As an example with reception and sending timestamps we can compute One-Way Delays. The drawback of this solution, which we address in this thesis, is the big amount of information that has to be sent on these various collection points. This can be alleviated by selecting carefully which information can identify a packet in order to reduce the information sent to the other collection points.

We said that we need to identify packets in various collection points, the issue is that packets get modified on each hop (TTL). Nevertheless, a packet can be identified by the flow it belongs to (Flow Identifier – *FlowID*) and with some fixed fields of the packet (the Packet Identifier – *PacketID*). The *FlowID* is straight-forward since usually it is identified by *IP Source*, *IP Destination*, *Port Source*, *Port Destination* and *Protocol*. To gather a *PacketID*, Table 3.3[1] shows the details about the variability of each IP header field.

In this regard the most representative fields we chose to identify a packet are: *Total Length* and *Datagram ID*. Despite that Table 3.3 also considers Protocol, Source and Destination addresses, we claim that they are not required since we already used them for the *FlowID*. Nevertheless these fields are not enough to distinguish packets belonging to the same flow, this can be achieved by considering 27 bytes of the packet's payload [38]. This is enough to guarantee low collision probability among the packets of the same flow.

Once all the above information is gathered by using a CRC-32 function we can compact the *PacketID* to a 32bit field that identifies our packet, together with the *FlowID*, which is also computed by the same CRC-32 as before.

The main problem of this methodology is the required resourced in terms of bandwidth for the control traffic. Consequently, in network monitoring, the reduction of the required resources for monitoring high speed networks is important [4]. Such reduction is often accomplished by

[1] This table has been extracted from [133].

Header Field	*Immutability on the Path*	*Variability Between Packets*	*Considered*
Version	Yes	Extremely small	No
Header Length	Yes	Small (only if options are present)	No
Type of Service (TOS)	No (some routers change this field)	Can be high (but usually not used)	No
Total Length	Yes	Can be high	Yes
Datagram ID	Yes	High	Yes
Flags	No	(intermediate Moderate routers may set the “don’t fragment” flag)	No
Fragment Offset	No	Can be high (depends amount of fragmentation and packet size distribution)	No
Time to Live (TTL)	No (decrements at each router)	Can be high	No
Protocol	Yes	Small	Yes
Header Checksum	No (changes always if other header-fields changed)	Can be high	No
Source Address	Yes	Can be high	Yes
Destination Address	Yes	Can be high	Yes

Table 3.3: Variability of IP header fields

using different techniques such as traffic sampling which permits to infer with high accuracy the traffic characteristics [37]. We leave the full description of Sampling techniques to Section 4.3.3.

In this thesis we use both the proposal about distributed packet matching and distributed traffic sampling to build our Network Parameter Acquisition System.

4

Initial challenges

In general network measurement is not an easy task, and QoS measurement in particular is not an exception. In any measurement environment, before measuring any metric some issues have to be considered. This section highlights some typical problems found on network measurements such as clock synchronisation or analysis of the results, and points to some advise to avoid them.

Moreover, in the final section we present the lessons learned from our experience in measurement infrastructures, which gave us the required knowledge about network measurements to develop our SLA Assessment system.

4.1 Synchronisation

Measuring often uses One-Way Delays (OWD) as a performance metric (see Section 3.1.1). This process involves timestamping of the received packets at different measure points (OWD) where data is gathered. In order to have accurate one-way delays, all the involved clocks on the measurement have to be synchronised (e.g., indicate the same time value at the same instant). Besides the complexity of having accurate time sources, the main issue in this context is to verify that the computed metrics are correct. Hence, while measuring delays in a distributed scenario, it is important to keep track of the clock accuracy [116].

Sometimes it is very easy to learn that two clocks are unsynchronised, these cases typically are: when we obtain $OWD < 0$, an impossible situation due to the always increasingly nature of time, and the opposite, when we obtain unrealistic huge OWDs. In these cases it is very easy to automatically discard the measurements as invalid. Nevertheless, in many other cases,

even with wrongly synchronised clocks it is not possible to spot the errors because they are *"believable"* even if incorrect.

Synchronising clocks may seem straight-forward by using well-known protocols such as NTP. But when working with microsecond resolutions (not rare in current link speeds), the clocks must be very accurately synchronised in all the hosts involved in the timestamping.

Related to this, another critical consideration is the used operating system's (OS) (or in its case hardware) accuracy, even with reliable times sources, our OS updates the clocks using software interrupts, which are invoked several times per second (typically between 100 and 1000). This usually bounds the clock's accuracy. Luckily, in current kernels of most UNIX systems there are techniques that permit to compute the time between interrupts, therefore inferring the time with much higher accuracy.

Due to physical limitations, no clock is perfect; each clock has slightly different rates (*skew*) that are noticeable at small time scales. Moreover, the skew is not constant as it is affected by various external conditions (e.g. changes in temperature). These variations on the skew are known as *drift*. Hence, if the measurements do not include clock precision information, it is not possible to know whether the results are accurate enough.

Synchronisation must be maintained, and estimates about the error have to be known during the timestamping process. This can be achieved in different ways; the most relevant in network measurements are:

1. *Software Synchronisation* is the less precise form of clock synchronisation. This approach requires a daemon running on the host and an external time source that is used as a reference clock. The daemon instructs the system clock to converge towards the reference clock. The most used protocol is the Network Time Protocol (NTP) as defined in [87].

2. *Hardware Synchronisation* is another approach using specific hardware connected directly to a reliable time source. The host using an external time source (e.g., a GPS antenna or PPS - Pulse per second source), uses the precise time information delivered and performs the timestamp directly on the Network Interface Card (NIC), which forwards such information to the operating system.

   This mechanism guarantees precise timestamping, since no software is involved in the process. The drawback is the availability of NICs with such hardware capabilities.

Another methodology, standardised under IEEE-1588 [40], gives LANs very high accuracy mechanism for synchronisation by using specific hardware and separate synchronisation channels that deliver up to nanosecond accuracy in the clocks.

3. *Mixed Software and Hardware Synchronisation:* In some environments having hardware, timestamping is not possible or too expensive. Hence, a common solution for synchronisation uses both software tools (NTP) with hardware time sources (GPS antennas). This can be used in a LAN to broadcast time information to all the hosts in a more scalable way than using a pure hardware solution, achieving estimated errors within few microseconds precision.

Lately in [125], another trend in synchronisation is being proposed. Instead of using remote clock sources, their solution is based on the use of the TSC (Time Stamp Counter) register found in modern microprocessors, which gives a higher accuracy on the local clock. The issue that remains to be solved is how to exploit this technique in a distributed environment with multiple clocks.

4.2 Data Collection, Storage and Analysis

Network measurement is typically divided into three different phases:

1. *Data gathering*, collecting of all the relevant data for the experiment.

2. *Data storage*, selecting which data to store and determining which analysis is possible to perform later on.

3. *Data analysis*, study and delivery of the results about the data acquired on previous phases.

Often, measuring network metrics requires the collection of existing traffic on the links. Such a collection is different depending on the layer where it is performed. Usually, the collection point is a station located on the path of the traffic under study. Before performing any collection, one must answer to these questions: *i)* where to collect data (e.g., in the border router, in the end point, etc.). *ii)* what to collect (e.g., all the packets, just one flow, one Class of Service, etc.). And *iii)* how to collect it: using active or passive traffic measurements. Which available tools exist useful for the task, etc.

Depending on what is collected and the traffic load, the data collection task can be very resource consuming. Moreover, it can require using specific hardware equipment.

In order to decrease the required resources for such collection, different techniques have been proposed, *i)* traffic aggregation (i.e., by using protocols such as SNMP), or *ii)* traffic sampling (i.e. as proposed in [37]). These techniques are detailed below in Section 4.3.3.

After collecting all the required traffic, it needs to be stored (for later processing) or analysed (in case of real-time analysis). Storing the data is not straight-forward, as the collection process can generate huge amounts of data per time unit. When measuring QoS parameters, the payload of the packets is often not relevant for the study, thus it is only necessary to store the packets headers.

Sometimes, simply storing the traffic is not sufficient. Providing meta-data along with the results permits one to understand how the measurements were taken to reduce the analysis complexity, in case the measurements are performed long before extracting the results, or to give other researchers better insights about the data. Such meta-data is important as it describes how the measurements were taken, what the characteristics of the traffic are, what the reason for testing was, explains the considerations taken to do the testing, etc. Work in this direction by the IETF's IPFIX working group in [101] defines a generic format designed specifically for storing networking information in a structured way. This format is based on XML in order to be extensible, and allows representing any kind of traffic along with its descriptive meta-data.

The last phase on network measurement is the analysis of the results. Such results may be processed on-line (i.e., for real-time QoS analysis), or it can be performed off-line (i.e., whether the QoS contract has been provided for a given test).

On-line analysis has the limitation that not all the information is available, since it arrives during the study. Thus, concepts such as average one-way delay, or maximum jitter are limited to the already received set of data. Therefore, on-line analysis has to be efficient due to its real-time nature, limiting the amount of computations executed per time unit.

On the other hand, off-line analysis does not have the time constraints and the whole data set is available. In this context, in order to analyse the data, statistics are used for summarising the obtained results. Not all the statistics are fit to describe all the measurements. More information about the most common statistics is follows in next section.

4.3 Base Techniques

Once the measurements have been performed, the next issue to solve is analysing and presenting the results rigorously. This can be achieved with the proper mathematical theory. Usually statistical tools give the necessary mechanisms, eventhough, sometimes the volume of data to manage is too big, or hardware limitations do not permit obtaining all the desired data. In this case it is required to use volume data reduction techniques, such as traffic aggregation or sampling techniques, which help in easing the information management overhead of the system.

This section discusses some statistical principles that can help to represent the data meaningfully, along with the most basic aggregation and sampling techniques, which help to discard data of the measurement set without affecting too negatively the final result.

4.3.1 Statistical Tools

QoS metric measurements deliver objective results about network performance, but depending on the objectives of the measurement, just having the metric values is not enough.

Statistics are a mathematical tool that permits formally summarising a set of results in order to look for some desirable properties and to obtain significant results. A full discussion about statistics is out of the scope of this section. Only basic statistics that help the QoS measurement are presented. Part of this section is based on [65] where a clear description about the useful statistics for the metrics is provided. Some hints are taken from [30], where the authors provide a good overview on Internet Measurements in general.

Among all the estimators, the most used in network measurements are the simple first moment estimators: the average and the standard deviation, which are used to summarise the results from the experiments. It has to be noted that such estimators can often be misleading, depending on the size or the diversity of the samples.

Other broadly used values are the minimum and the maximum values. When using them one must care about the existence of outliers. For example, as we detailed in Table 2.3 in Section 2.1.2.1, some upper bounds for the metrics are defined. Such upper bounds refer to average values (as an estimator of the mean) in case of IPTD (or OWD) and *maximum* values in case of IPDV and IPLR. In these QoS measurements, *maximum* values assume that no outliers are present on the set of results. In order to remove such outliers, the most common technique is to use the 99.9th percentile ($1-10^{-3}$ quantile) of the set. This implies that for obtaining this percentile we need a set of at least 1000 samples.

Differently to upper bounds, lower bounds do not tend to have the outliers problem, since they usually are bound by physical constraints (distances, bandwidth, processing time, etc.). In the case of some metrics such as OWD, such lower bounds are rarely studied. Nevertheless, if needed, the $0.1th$ percentile can be used [65].

Besides the pure statistical analysis of the data, it is often useful to have a general overview on the results (i.e. graphically representing the data). One broadly used graphical representation is a histogram showing the percentage of values on the result set with a given property. In QoS measurements, this often means OWD or IPDV. These kinds of histograms are known as One-Way Delay Distribution (OWDD) and IP Delay Variation Distribution (IPDVD). This shows how the spectrum of OWD (or IPDV) is distributed, and usually help to visually understand the frequency spreading of the samples.

Even though broadly used, histograms tend to aggregate the data into bins, with a consequent loss of information. In order to overcome this limitation, using Cumulative Distribution Functions (CDF) tend to be more understandable in general. Examples of these graphical representations can be found in later chapters of this thesis.

4.3.2 Aggregation

Aggregation is a technique that groups together a set of data with similar properties with the goal of reducing the data set, and speeding up the processing. In the networking area common examples of these properties are: aggregating statistics per flow, per source address, or destination port, etc. In QoS, the most common example is the equal treatment of packets within the same Class of Service (CoS) in a DiffServ environment.

Aggregation is always tied to loss of information, if there is no loss, the process is called classification and its goal is to ease the further location of the data. However, the aggregation is irreversible and the aggregate always contains less information than the original set. The reduction of the data to process gives the required speeding up of the analysis, so that even hardware not designed for statistical analysis can deliver meaningful results without excessive computational needs.

Networking equipment (e.g., routers, switches, etc.) contain statistical information about the traffic on the network. In order to reduce the computational burden of those statistics, the reported is aggregated in: packet counts per interface, number of flows, etc. All these statistics can be queried using *SNMP* (Simple Network Management Protocol). SNMP is the standard protocol for querying networking equipment. It is currently at version 3 [53] and

defines a structured query language for statistics retrieval, among other functionalities. Instead of keeping detailed information about the traffic profile, the networking equipment periodically updates the databases with the new data about the traffic, which permits to reduce the memory requirements of the operation, at the cost of losing information.

Another protocol that uses aggregation in order to deliver network statistics is Cisco Netflow [27]. It provides statistics for network traffic accounting, usage-based network billing, network planning and security. Netflow, instead of the generic approach of SNMP, is designed specifically for flow based statistics, and delivers more detailed information about the traffic. Netflow has per flow statistics which include number of packets, protocol, amount of traffic, etc., with a versatile design to select which flows have to be monitored. Netflow exports the information using the new proposal from the IETF IPFIX presented previously. Lately, in the framework of the Lobster IST project [10], IPFIX has been extended with QoS reporting facilities. Such improvements include inter-packet distance, distribution of packet sizes, maximum rate, etc.

4.3.3 Sampling

Sampling is a technique broadly used for reducing the set of information. Opposed to aggregation, which clusters data with similar characteristics, traffic sampling chooses a subset of packets for analysis and ignores the rest. The selected packets are later analysed as if they were the full set. No statistical operations are performed before the analysis.

Selecting which packets to sample depends on the goal of the analysis. As presented by [30], sampling can be performed in three different ways:

- *Random sampling:* a packet is sampled with some fixed probability p.
- *Deterministic sampling:* using a deterministic function, for example sample one out of N packets.
- *Stratified sampling:* the set of packets is divided into subsets and the sampling is performed within the subsets. This technique is useful in the case that it is required to have at least one sample of each subset. The drawback is that the packet must be analysed before being selected.

All the above mechanisms are useful on generic networking measurements, but in QoS measurements it is not possible to use them, because computing the metrics requires a distributed infrastructure. Thus, it requires a deterministic selection of packets on all the measurement points. However, the deterministic sampling as presented before is not an option, because the packet sets of each measurement point might be different (packet losses, route changes, etc.). In order to overcome this, [38] proposes an enhancement to deterministic sampling to work in a distributed environment, namely *Trajectory Sampling* (also known as *Hash Sampling*), based on a hash function which guaranties that the same packet is sampled on all the measurement points. As detailed later in 3.3 this, together with [133], permits SLA assessment of arbitrary flows.

4.4 Understanding network testing

Before starting the description of the core of this thesis, and as a complement to the measurement background required to develop a robust distributed SLA assessment system, we present here some good practices anyone should consider to have a solid methodology to obtain accurate results when testing.

These best practices are remarked with the different contributions when necessary. We discuss some specific issues that appear usually in network testing, our focus resides on the negative effects some environments or methodology can inflict to the results.

About System Resources and Equipment

Before starting any traffic analysis work, the first question we must answer is: *Are we using the proper hardware?*.

This question is determinant for the proper outcome of any traffic measurement we want to perform. The fact is that system resources limit the maximum processed information rate, that can be: the amount of packets we can generate per time unit, or in the case of using passive measurements, the highest rate of collected and processed packets, etc.

As an example in [112] we performed a series of CPU intensive traffic captures with different traffic patterns using commodity hardware to see where the limits are in terms of packet collection. The results show that for high traffic loads the system starts dropping random packets, rendering unusable any analysis we want to perform with the data. The key point here is

that we do not know the amount or which information we lost, hence it makes impossible to correct the results.

About Testbeds

Another important issue to consider when performing network testing is about the suitability of the testbed under analysis. There are different types of testbeds:

- *Local controlled testbeds*: those where all the involved machines and networking equipment are controlled by the tester. These kind of testbeds are very useful to develop and analyse protocols or to study different stress conditions, since the tester can introduce any kind of network disruptions, in a controlled manner. A very useful set of tools to do this can be found at [75]. Where a detailed description about `NetEM` and Linux Traffic Control capabilities is performed.

  The drawback of such testbeds is the lack of *real world* traffic or unexpected situations found in other environments.

- *Local non-controlled testbeds*: In this category fall Corporate, middle sized testbeds, used for production but where the tester can perform some tests. This permits to have more realistic cross-traffic and more *real world* situations even with some degree of control since all the topology and internal details of the testbed are known.

- *Distributed testbeds*: Wide area testbeds with links and stations on other networks distributed among several countries and controlled by different administrative units, but with partial knowledge about the network topology. These testbeds are very useful for several real tests with unknown cross-traffic and sufficient hops in order to have very accurate estimation on how the system under evaluation works. A fair example in this category is the Gêant Network with each countries' NRENs.

- *Distributed* real *testbeds*: The same as the above but involving home users with a broad range of different access technologies and with different operators. Therefore with absolutely unpredictable situations and unknown topology. In this category fall all the tests performed in commercial networks. However, good and extensive quality tests in this environment are very hard to obtain given the lack of support from the corporations.

As a good analysis of *Distributed Testbeds* described above in [111] we presented the structure and performance of the EuQoS [45] testbed, which was composed by 12 different testbeds with several different access technologies (Ethernet, UMTS, WiFi and xDSL). This diversity and control over the end-points of the testbeds gave a perfect environment in order to produce high quality tests, as we will prove during this whole thesis.

About Access Technologies

Sometimes, the access technology limits the possible set of tests to perform. For example, when testing protocol efficiency, response time or any time related study at protocol level, the researcher has to be aware that the results might vary greatly depending on the underlying technology. It is normal to have higher packet loss ratios or bigger one-way delays on shared medium networks, namely WiFi or UMTS, than in more controlled and reliable, such as Ethernet.

Therefore the tester has to know with great detail the effects of the underlying network in order to remove their effects, if necessary, when studying the results.

Another face of the impact of the access technology is when we want to study a specific property of a given access technology. In this situation we must have tight control and knowledge about its environment and properly isolate the characteristic under analysis.

In [22] we performed a series of studies about handover time in a wireless environment using Mobile IPv4 and Mobile IPv6 in order to compare the effects of both protocols. In that case for providing accurate results we forced each handover at a given time by lowering the visibility of the mobile node respect to the access point, therefore forcing the handover to the more reachable access point.

Using this (or other) structured methodology gives to the testing, one critical property, the repeatability of the tests, which in later stages of testing gives more insights about the obtained results.

About Synchronisation

Again we consider synchronisation as a critical requirement when testing. Besides the obvious case when we need to compute OWDs or any similar metric, in a first approach a researcher might think that when using single points of analysis on a network, when not needing one-way

delays or when working with a single timestamping equipment, there is no need of synchronisation.

The above affirmation is common misleading reason for having incorrect results. Always when our testing involves timestamping we will need of a synchronisation device, either hardware, for very fine accuracy or software for a coarser one. The reason is, as said before, accuracy. If our system's clock does not have any reference source, the time passed between two different samples might differ in a random value caused by our clock's skew and drift, therefore presenting incorrect results, which are very hard to detect.

That is the main reason why, most of the measurement hardware equipment (e.g. DAG Cards [42]) include facilities for direct GPS timestamping on the card, even if these network cards do not include any end-to-end or distributed infrastructure for OWD or any other metric computation.

About Metrics

Finally computing metrics is a very important matter to consider in our environment. Besides its meaning, some metrics give veiled information, or behave in a specific way which can give very valuable information.

In [114] and in [110] we studied in detail the behaviour of packet losses both in a controlled testbed and in a distributed testbed. Such studies permitted us, as we will show in Chapter 7, to greatly improve the accuracy of our distributed system by using this information.

Another metric that can unveil particular network behaviour is the packet reordering, this is a very interesting metric which impacts on the protocol efficiency, and puts to test the application buffers as it can disrupt greatly any real-time communication. Moreover, out-of-order packets can give information about load-balancing links found in the Internet today, as we proved in [113].

5

Network Parameter Acquisition System

The Network Parameter Acquisition System (NPAS) explained in this chapter is the starting framework where we will construct incrementally our SLA assessment infrastructure. We present here the basic building blocks that form the system, which will be optimised on subsequent chapters. Our focus for the whole thesis relies on such intra-domain reporting optimisation. Nevertheless, in this chapter we introduce some techniques in order to use the system in inter-domain environments.

5.1 Building Blocks

NPAS is a distributed infrastructure for real-time QoS parameter measurement and on-line reporting. It has three main features, *i)* extensible intra-domain traffic analysis, *ii)* end-to-end information reporting, and *iii)* hardware and network independence. Although the platform has been designed to operate on both intra and inter-domain environments, this thesis mainly focuses in intra-domain scenarios. Outlining the requirements and presenting a first approach to a full inter-domain system.

We follow the passive traffic analysis methodology in order to perform the network quality assessment. To do this we propose the direct acquisition of network metrics. Therefore, the network parameter acquisition is performed by different collection points spread over the edges of the network. These collection points report to a higher level entity within the network

domain that extracts information about the traffic under SLA constraints. This entity and its associated collection points form a Measurement Domain (MD).

We propose this distributed infrastructure because each collection point can only access to local packet information, while usually most metrics such as One-Way Delay (OWD) require distributed information (i.e. timestamps from the collection points). All the collected information is aggregated and published to the neighbour domains in order to have end-to-end information.

The above entities form a two layer infrastructure. The first layer is in charge of intra-domain reporting, where detailed traffic information is gathered and reported to the higher level unit (See Figure 5.1). While the second layer covers inter-domain reporting, which consists of publishing aggregated information about the traffic. Such traffic is aggregated depending on the underlying network, for example using classes of service. With this approach it is possible to reduce the overhead caused by the gathering and publishing of the network status, specially on inter-domain links where the control traffic might be subject to some constraints.

The most relevant packet information extracted for the assessment is: One-Way Delay (OWD), IP Delay Variation (IPDV) and Packet Loss Ratio (PLR). In this stage NPAS does not pretend to define the actions, neither the values nor thresholds necessary to detect SLA violation periods. All these actions and QoS enforcement mechanisms are left for higher layer entities and out of the scope of NPAS.

NPAS is divided into three main entities as shown in Figure 5.1: *i) Monitoring Entities*, *ii) Processing Entities* and *iii) Inter-Domain Subscriber Entities*. Full discussion of each entity follows.

5.1.1 Monitoring Entity

The Monitoring Entity (ME) collects the QoS constrained traffic and extracts the required parameters (e.g., reception timestamp, packet size, etc.) for later processing. Since MEs act autonomously over a single network point, they only get local packet information. Thus, for computing network metrics such as OWD or Packet Losses, several ME are required.

ME is divided into two different parts, the Hardware Dependent Monitoring Part and the Hardware Independent Monitoring Part.

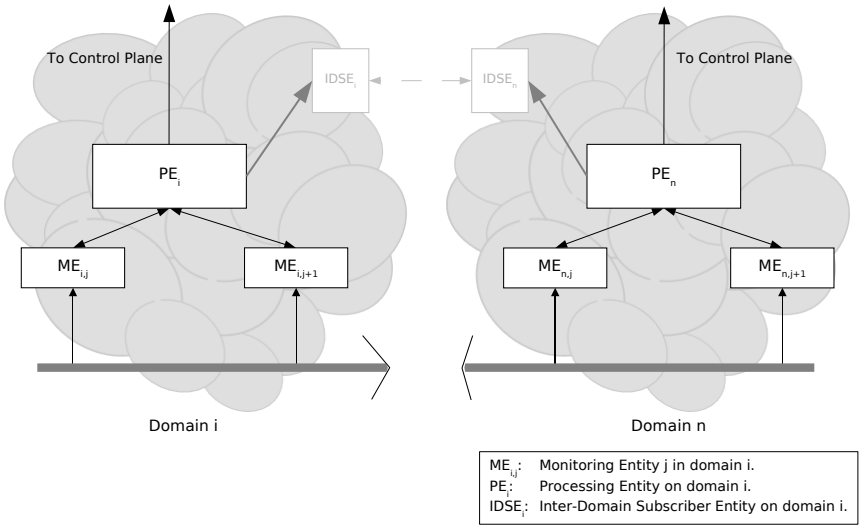


Figure 5.1: NPAS structure

5.1.1.1 Hardware Dependent Monitoring Part (HDMP)

It is in charge of performing the traffic collection depending on the underlying hardware and network technology.

HDMPs need to have direct access to the traffic. Usually, this involves the use of optical splitters or monitoring ports on the switches to replicate the traffic towards the ME, where it can be collected without interfering with the normal operation of the network.

Current link speeds make traffic collection a difficult task. For this purpose there are specific hardware products (i.e, Endace [42] DAG cards) that permit line speed collection in high-speed networks, but its deployment costs in most environments make it an expensive alternative.

Another possibility is to use commodity equipment. In this context, this solution is feasible because the number of flows under QoS restrictions is relatively small compared to the total network traffic. Thus significantly reducing the required resources for the collection tasks. However, our solution is flexible enough to be used with specialised hardware as well.

5.1.1.2 Hardware Independent Monitoring Part (HIMP)

It abstracts all the hardware details managed by the HDMP, sets up collection policies and provides a generic interface for collecting the QoS traffic. It also reports such information to

the higher level entities in charge of the processing and QoS parameter extraction.

Collection policies are based on a selection function that determines the level of aggregation in the reporting. The packet filtering policy is set up by the administrator or the QoS Control Plane. In this work we will consider as a proof of concept Per Flow and Per CoS traffic reporting.

The ME has to uniquely identify every collected packet to report its network information to the higher level entity, which will gather the information of all the ME to extract the QoS metrics. We achieve this by using two different identifiers, a *packet identifier* and an *aggregate identifier* (e.g. Flow, CoS, etc.). The packet identifier (P_{ID}) (32 bit) is generated by a fast CRC computation. It is obtained from: IP Source and Destination, Datagram Identifier, Protocol Identifier and 27 bytes of the packet's payload as we described in Section 3.4.2. And optionally, the TCP Window which permits to distinguish among retransmissions when TCP traffic is being collected. Using 27 bytes of the packet's payload overcomes the identifier's collision caused by some operating systems leaving a blank Datagram Identifier. This approach only requires to process a small portion of the packet. Further discussion about the selected fields can be found at [133].

We consider during most of this thesis the Aggregate Identifier (A_{ID}) as a Flow Identifier (F_{ID}) without loss of generality.

The Flow Identifier (F_{ID}) (32 bit) must be unique, as it identifies a given flow on all the ME. It is obtained similarly to the P_{ID} by computing a fast CRC, as described in [133], using the following header fields: Source and Destination Addresses (32 or 128 bits each), Source and Destination Ports (16 bits each) and Protocol (8 bits).

The Type of Service field used sometimes as part of the flow identifier [133] is not considered, since in a QoS environment this field might change along the packet's path depending on the DiffServ policies [76]. This Flow Descriptor is $|F_D| = 13$ bytes long, and it is sent independently to the Processing Entity for the first packet of each flow.

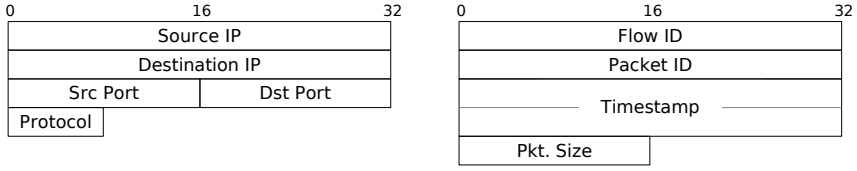


Figure 5.2: Frame format. Flow Descriptor (F_D) (left) and Packet Descriptor (P_D) (right)

The per packet information sent to the Processing Entity (each ME) is the minimum indispensable for extracting end-to-end QoS metrics. The descriptor is $|P_D| = 18$ bytes long and it is shown in Figure 5.2 together with (F_D) for later reference.

5.1.2 Processing Entity

The Processing Entity (PE) is the MD entity in charge of gathering the traffic parameters extracted by the MEs. It uses the P_{ID} and the F_{ID} fields to classify the packets and to identify them on each ME. Once a packet has been collected on all the MEs of its path it is possible to:

- Identify the ingress and egress points within the MD along the packet's path.
- Compute QoS parameters needed by higher layers within the MD:
  - *One-Way Delays* from each reported timestamp on the MEs, as described in RFC-2679 [5].
  - *IP Delay Variation* as defined in RFC-3393 [34].
  - *Packet Losses* as specified in RFC-2680 [6].

Packet losses can be computed by packets appearing on the ingress ME without showing up on the egress ME during a τ timeout interval. In following chapters we study packet loss and one-way delay behaviour and present values which reliably compute the PLR and give a good trade-off between reporting speed and accuracy of the estimation.

Each ME reports per packet information to its PE. This control traffic introduces a significant overhead within the MD. In later sections we evaluate the amount and effects of such control traffic.

All this information is forwarded to the QoS control plane and to the inter-domain subscriber entity. The QoS control plane will take the required actions to assure that the QoS is provided.

The protocol controlling all this intra-domain assessment is called *Intra-Doman Reporting Protocol* (IDRP).

5.1.3 Inter-Domain Subscriber Entity

End-to-end reporting involves often inter-domain links. Usually such scenarios belong to different administrative domains, which might limit the interchanged control traffic. This renders

the mechanism explained above not suitable for inter-domain reporting, since it assumes no constraints over the control traffic. Moreover deploying broadly such resource intensive mechanism is clearly not scalable.

IDSE is deployed as a service offered under subscription by each MD. It publishes aggregated QoS details from each PE to allow legitimate subscribers to extract them. The information is aggregated on a per CoS basis. This aggregation level is feasible given that in DiffServ domains all the CoS receive similar treatment on the MD. This overcomes the scalability problems because the number of CoS is low (typically 5 as defined in [65]).

The subscribers of this service are authorised entities (e.g. a peer MD or a network administrator) who queries all the MD on the interesting flow's path to compute whether the end-to-end QoS is provided or not.

5.2 Intra-Domain reporting

As stated above in the intra-domain environment, we propose the use of a protocol called Intra-Domain Reporting Protocol (IDRP), the design of IDRP is based on simplicity and during this thesis we will keep adding new functionalities and extensions in order to support the proposed reporting mechanisms.

In this initial stage, IDRP is formed by the most basic set of functionalities, namely support for per packet reporting, as detailed on Figure 5.3.

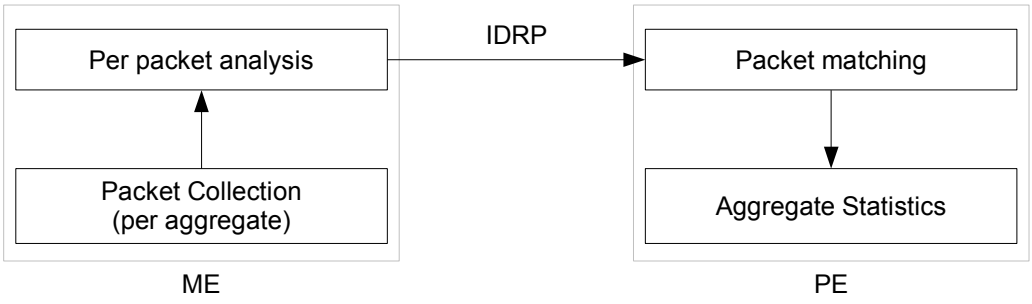


Figure 5.3: Initial Intra-Domain Reporting Protocol building blocks

In the figure, as described before, we have the ME, which captures and reports per packet information towards the PE. The PE in its turn, gathers the statistics which will be aggregated for further reporting by the IDSE or by the control plane.

The frame format for this per packet reporting is the Packet Descriptor detailed previously in Figure 5.2.

5.3 Inter-Domain reporting

Since we will not consider inter-domain reporting in the following chapters. This section is devoted to overviewing several possible techniques that extend the presented framework in the more general inter-domain reporting.

To reliably perform inter-domain SLA Assessment it is necessary to design a specific methodology, we do this via the *Inter-Domain Subscriber Protocol* (IDSP), which goal is to collect that information about the QoS level offered on end-to-end paths. For that purpose in a given domain, the PE periodically advertises its aggregated measured values about QoS parameter information to its associated IDSE, this information is structured on the IDSE in a way that any domain with exchanging traffic to the same particular destinations can query its status. For the destinations that are located inside the domain, each PE advertises the locally measured QoS values.

The aggregated values are assembled using the QoS parameters received from the neighbouring PE as well as the local QoS contributions introduced between the corresponding pairs of border routers when available.

Figure 5.4 shows an example illustrating how the IDSP collects information about the QoS. The QoS is offered in end-to-end paths across three domain scenario, from domain AS_1 towards domain AS_3. We assume that each domain allocates one deployed IDSE that periodically acquires measurements representing the offered QoS inside the domain (Q_1, Q_2 and Q_3) as well as on adequate inter-domain links ($Q_{1\to2}$, $Q_{2\to1}$, $Q_{2\to3}$, $Q_{3\to2}$) in case it is needed.

In a first generic approach, PE_3 propagates information about its computed QoS parameters, denoted as Q_3, to PE_2. The PE_2 checks whether domain AS_3 is a peer (which means that it is available in its BGP routing table) and, in such case advertises aggregated information to PE_1 about the offered QoS towards the destinations located in domain AS_3. This value considers QoS contributions of domain AS_3 and AS_2 (Q_3, Q_2) as well as the inter-domain link between domain AS_2 and AS_3 ($Q_{2\to3}$) so it equals to $Q_3 \oplus Q_{2\to3} \oplus Q_2$. The operator $\oplus$ denotes an aggregation function which is specific for each particular QoS parameter (e.g. a sum for additive QoS constraints such as the OWD, a product for multiplicative QoS constraints such as PLR,

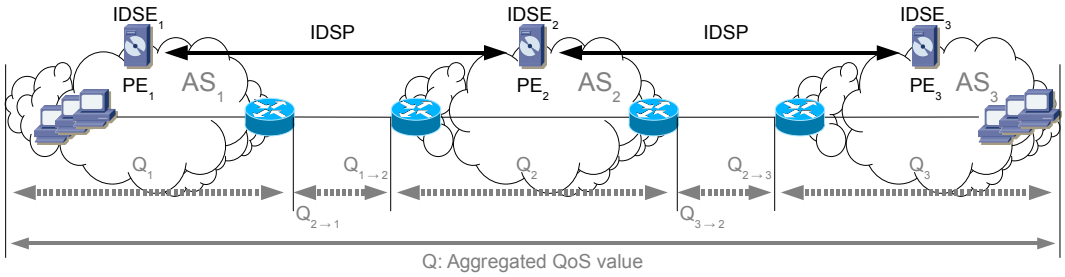


Figure 5.4: Example of IDSP

or the minimum for QoS constraints such as bandwidth). Finally, following the same scenario in domain AS_2, PE_1 receives information of the QoS offered on the path towards domain AS_3.

Assuming that each IDSE periodically advertises information about the offered QoS towards available destinations, each IDSE will create a *QoS map* that shows updated QoS values offered towards any reachable destination. Therefore, each IDSE has to maintain a table (similar to a BGP routing table) that contains available destinations jointly with the corresponding QoS parameters and associated PE.

The accuracy of the gathered end-to-end QoS values depends on the precision of QoS aggregation functions as well as on how frequently PEs send updates to their neighbours. Updates might be triggered for reporting special network conditions when needed.

5.3.1 Deployment Model

This section defines the deployment options along with some important decisions related with the setup of an inter-domain NPAS. After the generic model and protocol presented above, we provide here different alternatives, each one presenting a set of advantages in respect to the other.

The reporting granularity and its different options is an important decision when deploying NPAS, and hence several concerns have to be addressed. One refers to administrative issues, that is, sometimes it is not possible to deploy a ME on all the hops over the traffic path. In this regard, NPAS is designed with such flexibility in mind. The MEs can be spread in different points with one or more PE to report to. Depending on this deploying density more detailed information can be extracted.

Depending on the location of the ME, the reporting can be done from the intra-domain scenario up to end-to-end with similar infrastructure. IDSE will be in charge of reporting the desired aggregated metrics computing finally the end-to-end values. When referring to end-to-end we mean as network level (from ingress to egress nodes of the whole path). The users experience as well as applications specific metrics are considered in the Chapter 9.

As a particular deployment of the QoS monitoring system, we propose three different approaches in a inter-domain network scenario. The main objective of on-line QoS monitoring is to provide the updated network information about the offered QoS level. The information is about the established paths between its own domains and other destination domains. The information provided by the on-line QoS monitoring function constitutes a base for *i)* providing "certificates" for users about assured QoS level and *ii)* OAM (Operation, Administration and Maintenance) process that triggers fault management actions in the case when QoS level falls below the expectations.

Designing an "on-line" monitoring system for inter-domain network is a complex task as shown in [57; 83]. The main problem comes from the fact that domains are usually under autonomous administration and as a consequence their operators have their own policies for performing measurements. On the other hand an on-line monitoring system also requires some cooperation between domains. The main requirements for the measurement system deployed in a given domain are the following:

- The measurements performed by a particular domain should correspond to the same metric and have to be measured at the same technology layer, usually IP level.
- The measurements should cover the whole path so they have to be performed between clearly defined hops of the path (e.g. border routers where the limits of network service of a particular domain are).
- The measurement system has to interoperate with other systems.
- The measurement system should allow for scalability when networks with large number of domains are investigated.

The on-line QoS monitoring system may be implemented in different ways. Among them we distinguish three main categories:

- *Centralised* solution that assumes a single entity controlling all measurements performed inside entire network, including several domains.
- *Fully distributed* solution where each domain has its own control entity responsible for performing measurements inside a domain, while measurements corresponding to end-to-end paths are collected with the aid of the control and reporting protocol. The base information for this protocol comes from the results of local measurements performed independently by each domain, as we already described.
- *Semi-centralised* solution where each domain has a control entity responsible for performing measurements on the path between its domain and any destination domain.

5.3.1.1 Centralised on-line monitoring system

This approach assumes that centralised measurement controllers (PE) are responsible for managing of all measurements preformed inside each domain. This solution can be extended to any arbitrary number of domains, see Figure 5.5. The PE controls all MEs located in the domain borders as well as on the borders of access and core networks.

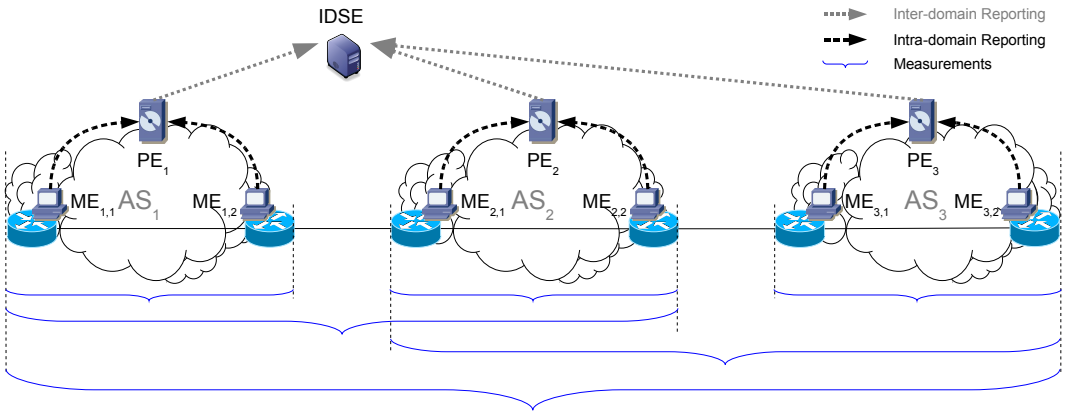


Figure 5.5: Architecture of centralised on-line monitoring system

The main advantage of this approach is its simplicity of implementation. However, because of assuming a centralised point of control, this solution is difficult to deploy in a network consisting of many autonomous domains, because it is not scalable. As a consequence this approach may be applied for small networks under a common administration, like in the case

of trial or dedicated networks. As an example, such approach was implemented in the ETOMIC project [83] where measurement probes were deployed in different sites that were connected to the Géant academic network [49].

However, this approach can keep a proper accuracy of the reporting system, since we have fresh information sent by all the PE.

5.3.1.2 Fully distributed on-line monitoring system

To reduce the strain of the centralised solution, a different approach would be to have, as initially proposed, an IDSE per administrative domain. But in that case, to consider the effects of the inter-domain links, the first ME of each domain reports upstream (or downstream) information about that link, giving more accuracy mechanism for end-to-end reporting, specially because it reports inter-domain link status.

This solution assumes that monitoring of end-to-end paths is performed in a distributed way as presented in Figure 5.6.

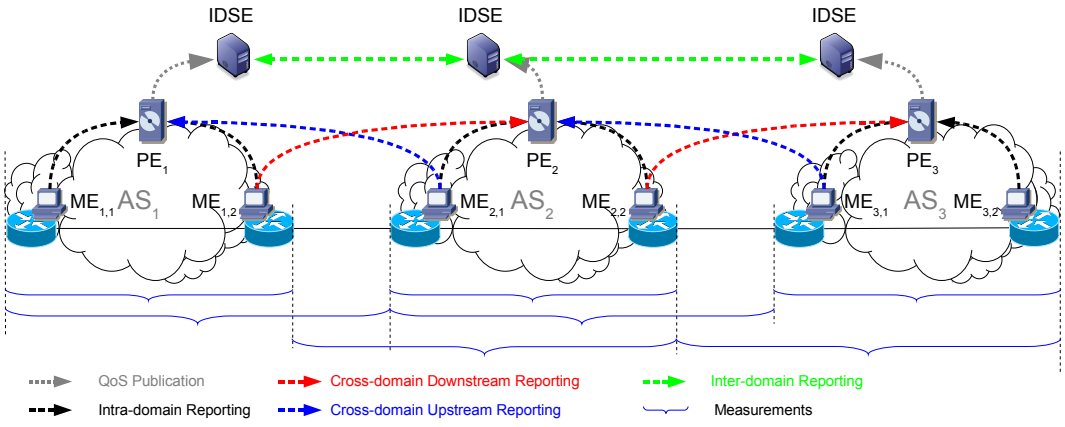


Figure 5.6: Architecture of distributed on-line monitoring system

For that purpose each domain implements its own domain measurement controller (PE) that is responsible for performing on-line measurements inside the domain. Those measurements are performed in a continuous way between any pair of points where the service is offered. In a typical domain, measurements should be performed between all pairs of border routers. With the goal to measure QoS offered for traffic crossing the domain, between all pairs of access networks to measure QoS offered for local traffic, as well as all pairs of access network

and border router to measure QoS offered for traffic originating or terminating in this domain. For obtaining end-to-end measures, the intra-domain results are periodically advertised to the neighbouring IDSEs. The used protocol for the task is the IDSP presented before.

The main advantage of a distributed on-line monitoring approach corresponds to the independency of measurements performed on the different domains. The only requirements are that the measurement systems deployed in a particular domain should either measure the same QoS metrics, implement the same measurement and control protocol; or present mappings among the different policies. Due to the distributed operation, this approach can be easily applied in a inter-domain network. On the other hand, the main drawback corresponds to the accuracy of end-to-end results. In fact the results are not directly measured as they are based on the aggregated results measured by local systems. However, by properly designing of the assembling functions used for calculations of cumulative values of QoS parameters (as proposed in [13; 21]), we may approximate the upper bound for a particular QoS metric.

5.3.1.3 Semi-centralised on-line monitoring system

This approach assumes that each domain has its own IDSE that is responsible for centralising measurements on all paths from its own domain towards any required destination domain. For instance on Figure 5.7, the $IDSE_1$ can control the PE from all domains, hence we can measure paths between domains AS_1 and AS_3, as well as between AS_1 and AS_2. Unfortunately, in this approach it is not possible to collect measurements between AS_2 and AS_3.

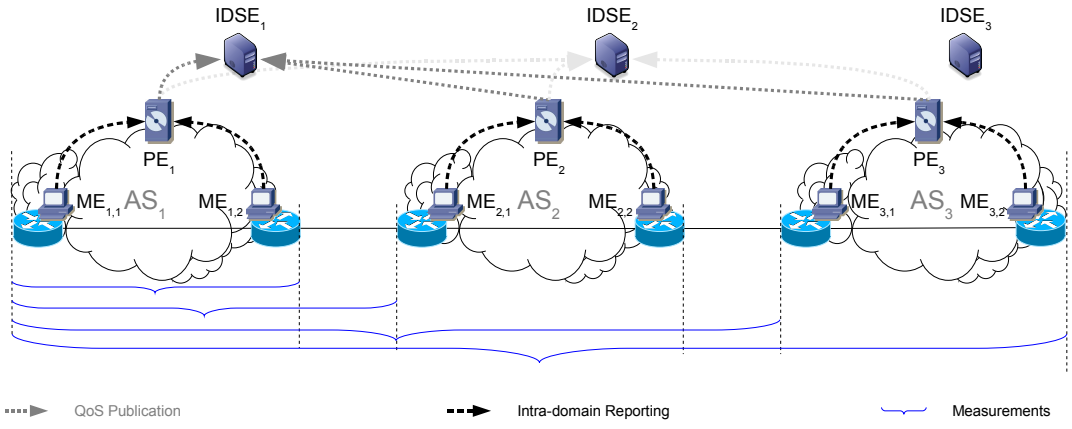


Figure 5.7: Architecture of semi-centralised on-line monitoring system for performing measurements from AS_1.

This approach allows us to get measurements on all end-to-end paths, however it requires that MEs are controlled remotely by PEs belonging to different domains. This constitutes the main limitation of semi-centralised approach, as it may be only applied between domains that trust each other. Nevertheless, it is possible to restrict such communication using trust relations. Such approach was proposed by the IST-INTERMON project [57] where the function of IDSE was performed by a *Global Controller* entity that communicates using the Specification of Monitoring Service interface.

5.4 Architecture Deployment

As a proof of concept, we deployed the infrastructure described in Section 5.1. Such infrastructure has been developed in the framework of the EuQoS project, which poses a very good testing environment as described below.

5.4.1 The EuQoS System

The term EuQoS stands for *"End-to-end Quality of Service support over heterogeneous networks"* [45]. This European research project built a complete QoS system aimed at supporting stringent QoS services among heterogeneous access technologies, both within and between different routing domains. EuQoS customers will be able to subscribe to this system, for which we developed a series of novel QoS mechanisms and protocols that build upon the state of the art technologies. Among these we have security, admission control, signaling, monitoring measurements and reporting, QoS Routing (QoSR), fault management, and Traffic Engineering (TE).

The focus of our work resides on the Monitoring and Measurement System (MMS) and the Monitoring Measurements and Fault Management (MMFM), which are the realisations of the NPAS within the project.

MMS The EuQoS MMS functionality is twofold: to validate of the QoS delivered by the EuQoS system (in performance trials) and to support other EuQoS functionalities. Such functionalities include topology acquisition, network resource usage and delivered QoS levels in order to support traffic engineering. The intra-domain portion of NPAS falls within the specification of the MMS.

MMFM The MMFM is the EuQoS function in charge of network resource monitoring and network topology discovering, fault management and QoS monitoring. The Network Monitor inside the MMFM manages different threads in charge of interfacing with the different MMS parts in charge of supporting the EuQoS system.

The interfaces between MMS (NPAS) and MMFM (which holds the role of our IDSE) are defined in order to support periodic monitoring, where MMFM periodically asks the system for measurements and event notifications in a per domain basis.

The MMFM gets the measurements from the different MMS parts processes them and commit the changes to a database for persistent storage. This database is used by different EuQoS modules to get information about the network status, the QoS performance information and the usage of the link bandwidth. This permits later queries and historical data repositories for off-line processing.

5.4.2 Resource consumption

Previous sections dealt with the internals of a generic NPAS system, where we highlighted the scalability issues of the solution. In order to further investigate the required resources we performed a series of tests in order to experimentally study the system requirements.

5.4.2.1 Testbed

Monitoring high-speed backbone links using generic purpose equipment poses a high demand on optimising memory usage, because one needs to track parameters of tens if not hundreds of thousands of simultaneous flows. To have better insights about the used bandwidth when deploying our solution, typical traffic characteristics must be known. We collected a packet level trace at the core link of the Spanish NREN[1]. This Gigabit ethernet link has an average load of 360$Mbps$ which under our opinion constitutes a representative sample of backbone traffic. This trace was collected during 30 minutes on November 2005 with a peak of 483$Mbps$. With a total amount of 103.7 million packets.

The trace showed around 10000 different flows per second and an average of 44000 packets in the same period. Moreover we experienced a new flow rate of around 2000 new flows per second.

[1]Provided by the SMARTxAC project [117].

The results given in this section assume the unrealistic scenario where all the traffic is under QoS restrictions, so the obtained results are strict upper bounds of the real cost of the solution in such a network.

5.4.2.2 Memory resources

PEs will store information for each flow detected by each ME, even if the flow is not detected on all of them. The flows will be stored in a hash table that facilitates quick lookups but easy updates as well. A flow is considered *active* if packets with its descriptor have arrived within the last 30 seconds, after that the flow will be *expired* and removed from the data structures.

The information kept about each flow includes the F_D, that includes the F_{ID}, the source and destination addresses and ports as described in Figure 5.2. Other fields stored are the counters needed to monitor the QoS parameters, and to perform internal computations of detecting and synchronising the order of packets between the distributed MEs. These are responsible for the bulk of the memory requirements but they are needed for detecting the direction of traffic and for seeing if a F_{ID} represents a flow that passes through all monitored points.

The general memory usage can be summarised in a total of 423 bytes per active flow plus the control structures of the hash table, which is negligible and fixed for the whole system per PE. This amounts to a total of $\sim 4Mbytes$ of flow structures, considering the above traffic conditions.

Related to the memory consumption per packet, the system only consumes 18 bytes, supposing the aforementioned 44000 packets per second this result around $\sim 780Kbytes$ of memory per second per ME, which is flushed and refilled constantly as new packets arrive. Assuming that we keep a window of 1 second of packets before considering the packet as lost we need $\sim 1.5Mbytes$ of memory per ME. Asuming we have 10 ME the total amount of memory, including the flows described before we need: $\sim 20Mbytes$ of memory, which is perfectly manageable with state-of-the-art hardware.

5.4.2.3 Bandwidth

The other important bottleneck to consider is the amount of control traffic generated by the system. This is important in order to evaluate the scalability of the platform.

Using the same test as before with the same conditions, we obtain a cost in terms of bandwidth composed by two different parts:

1. New flow rate ($\mathcal{N}$): $F_D * \mathcal{N} = 13 * 2000 \sim 208Kbps$

2. New packet rate ($\mathcal{P}$): $P_D * \mathcal{P} = 18 * 44000 \sim 6.3Mbps$

Amounting to a total of $\sim 6.5Mbps$ per ME, again, with 10 ME the total amount of traffic generated would be $\sim 65Mbps$ which compared with the average of $360Mbps$ of real traffic corresponds to a 18% of control traffic ratio.

5.5 Conclusions

This initial chapter presents the grounds and basic building blocks of a Network Parameter Acquisition System. The proposed system constitutes the main entities, namely Monitoring Entities, Processing Entities and Inter-Domain Subscriber Entities. Each one complying with the different requirements of the system: traffic collection, packet matching and metric extraction; and Inter-domain result publishing. All these make of NPAS a suitable proposal for a on-line SLA assessment infrastructure.

However, deploying a NPAS system as described here can consume a fair amount of resources, we proved this considering specific but representative network conditions. This poses the main scalability problem of the system.

The main contribution related to this chapter is [107], it already presents some ways of improving this scalability issues, we detail such improvements in chapter 6. In this trend, the major goal of the rest of this thesis is devoted at looking for ways of improving such poor performance, with the goal of making of NPAS a really scalable and broadly deployable system.

The different optimisations we will present range from time classification of the packets, to advanced traffic sampling techniques to accomplish the desired reduction in control traffic requirements.

6

Time based reporting

So far NPAS has defined a structured framework for on-line SLA assessment. But it requires a fairly large amount of resources in terms of control traffic on the ME. As a first approach, to alleviate this burden, we propose in this chapter a simple mechanism, based on time windows, that can drastically reduce the required amount of resources without any loss in the accuracy.

6.1 Traffic classification

Deciding the information that must be available on the PE or IDSE is not enough for having a scalable and reliable reporting system. We present here the initial set of extensions in the Intra-Domain Reporting Protocol for reliable and efficient reporting, we also study the cost of the solution in terms of bandwidth.

The Intra-Domain Reporting Protocol (IDRP) specifies the information exchanged between ME(s) and PE. In this extension, which we name *Extended Intra-Domain Reporting Protocol* ($E-IDRP_1$), we aim at the reduction of the used bandwidth in the ME and PE communication while keeping accurate information about QoS metrics.

The basic approach presented so far reports the QoS metrics in a per packet basis. Besides the obvious overhead caused by the IP headers, the whole solution is very expensive in terms of bandwidth. As described before, each new flow generates a 13 byte flow descriptor (F_D) that is sent separately from the packet's information. Together with F_D a P_D is generated for each packet. It requires a 18 byte data structure as already shown in Figure 5.2.

In summary, reporting per packet information depends on the new aggregate rate and the

packet rate for each monitored flow, which follows expression 6.1.

$$BW = |F_D| \cdot \mathcal{N} + |P_D| \sum_{i=1}^{n} \mathcal{P}_i \tag{6.1}$$

where $\mathcal{N}$ represents the rate at which *new* aggregates (flows) per second arrive. $\mathcal{P}_i$ holds the packet rate of flow *i* for all flows under analysis. As it can be noted the bandwidth usage of this mechanism grows linearly with $\mathcal{N}$ and $\mathcal{P}$.

In order to overcome this high demands the rest of this section presents an optimisation to the system by using traffic classification.

6.1.1 Extended Intra-Domain Reporting Protocol

The proposed mechanism, instead of performing per packet reporting, uses a Time Window (TW) for packet collection. Such time window is defined by a time interval *t* that sets the gathering periodicity and the reporting rate. This solution permits to reduce the reporting overhead, by removing some redundant information present on the per packet case, at expenses of delaying the packet's reporting. The frame format is shown in Figure 6.1. The fields contain:

- *Window ID:* It is an identifier which indicates the base time window on the ME for the aggregation group. It contains the timestamp of the window start time.
- A_{ID}: It contains the aggregate identifier (Flow, CoS, etc.).
- *Packet's CRC:* Identifier for matching packets among ME.
- *Offset:* Offset of the current packet since the beginning of the time window.

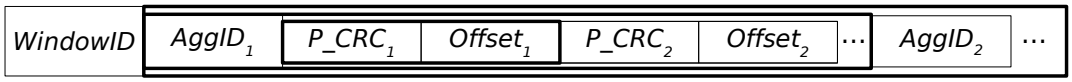


Figure 6.1: Frame format

This protocol extension fits in the overall system as a new optional layer as depicted in Figure 6.2. This new layer sets the Time Window, informs the ME how to encode the per packet information and instructs the PE on how to decode it. It also decides which is the aggregate to use.

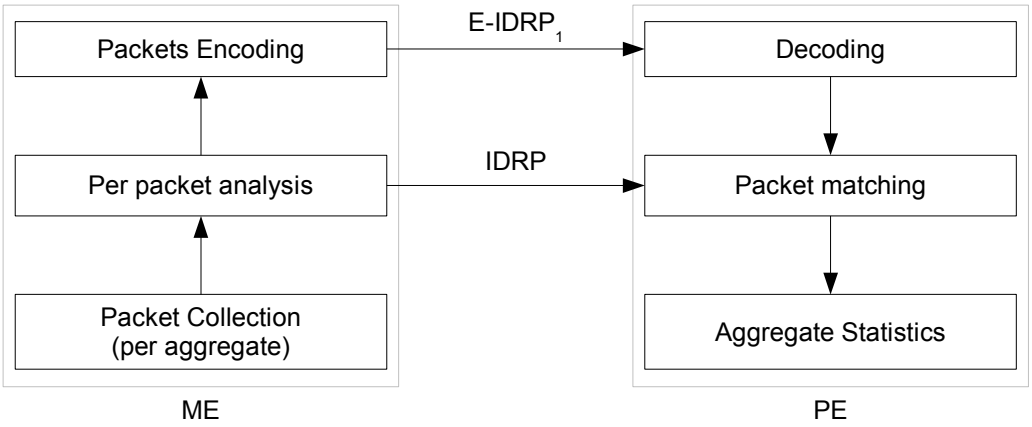


Figure 6.2: Block view of (E)IDRP

The advantage of this design is that the system, in case of need can fall back to the per packet reporting whenever it is necessary.

Another issue to consider is that this mechanism requires alignment in the TW among all the ME on the Measurement Domain, since PE matches each packet on the fly, it requires that the information for the same packet arrives as close as possible from all the ME. Even having this TW alignment among ME, in the window edges some packets can fall on different time windows as shown in Figure 6.3. This is caused by non-constant One-Way Delays. For example P_3 in ME_1, which is in Time Window $TW-1$ ends up in $TW-2$ for ME_2.

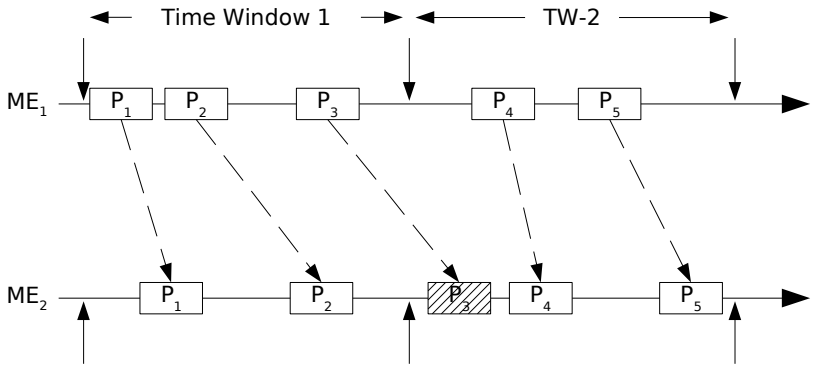


Figure 6.3: Time Window alignment problem

The TW misalignment can be avoided by having a buffer in the PE with a history of n time windows. Therefore $H = n \times t$ determines the time threshold when a packet falls out of the buffer, and is considered as lost by the system. Provided that the analysed traffic in QoS is constrained, it means that such traffic is sensible to high one way delays. Therefore considering these packets as lost is not a big issue from the practical point of view, assuming we honour the limits set by the SLA contracts.

In this extension n and t are critical parameters. On the one hand, t determines the reporting interval latency, on the other hand, n limits the maximum packet's OWD.

It is important to consider the following issues when choosing the values for these parameters:

- The traffic under analysis has QoS restrictions.
- Big t leads to high lags in the reporting.
- Both t and n determine the required memory, into the ME and into the PE.

The system requires fast response when the desired QoS is not provided, which implies small values for t. At the same time, H must guarantee that all the traffic within delay and loss limits is properly identified. This can be accomplished by using high values of t or n.

As reflected in [65] depending on the classes of service, the upper OWD bounds *end-to-end* are around $400ms$ with a IPDV of $\pm 50ms$ (notice that this values might differ depending on QoS policies).

Choosing t and n complying with the above restrictions has the trade-off between fast response and traffic parameters limits. Regardless of the selection $H \geq 450ms$ must hold in order to guarantee that the parameters are within the limits. Sensible values for t range from 50 to $225ms$. In the case of n in normal conditions it ranges from 2 ($n = 1$ is not a valid option if we want to avoid the alignment problem) to 9 (as $n \times t \approx 450ms$ for $t = 50ms$). Although depending on the QoS policies this values can differ. An experimental analysis and proper selection of t and n is presented on section 6.2.2 below.

6.1.2 Bandwidth and memory requirement analysis

Expression 6.2 models the bandwidth requirements when using traffic classification. Given a list of present aggregates $\mathcal{A}_t$, a list of New Aggregates $\mathcal{N}_t$ in t where $\mathcal{N} \subset \mathcal{A}$, $\mathcal{P}_{t_i}$ the new packet

rate in t of aggregate i, and $|o|$ the size of the offset. Then,

$$BW = |A_D| \cdot \mathcal{N}_t + |A_{ID}| * \mathcal{A}_t + (|P_{ID}| + |o|) * \sum_{i=1}^{n} \mathcal{P}_{t_i} \tag{6.2}$$

All the values in the expression depend on the time window t.

Depending on the aggregation type, the required bandwidth might vary largely, as it relies on $\mathcal{A}_t$ and A_D. Although, several types of aggregation might be used, as a proof of concept, this work uses Flow and CoS aggregation. The choice is compelled by the different overhead present on each alternative.

When using flow aggregation the overhead tends to be high, as there can be many new flows per time window, with the corresponding F_D (13 bytes) and the flow identifier (32 bit). While for CoS aggregation the overhead is much lower since CoSs are limited in real scenarios. For example in ITU's recommendation [65] just 5 Classes of Service plus *best effort* are defined. This represents a negligible CoS descriptor.

The drawback of high aggregation techniques, such as CoS aggregation, is the loss of information tied to it, as different flows from different sources and to different destinations are aggregated and considered as similar traffic. In this context this is not an issue, since Differentiated Services guarantee that each class is treated fairly on the whole DiffServ domain [33].

Comparing expressions 6.1 and 6.2, we can infer that the overhead produced by the A_D is the same in both solutions, but in the case of the packet reporting itself with this new proposal the obtained reduction comes from the fact that $|P_D| \gg (|P_{ID}| + |o|)$ in particular, $18 \gg 6$, expecting in a theoretical best case, three times less resource consumption with the packet reporting. Notice that the overall reduction will be lower than that since the A_D is the same as before, unless no new flows arrive during the period, or we use highly aggregated traffic collection (e.g. CoS).

In terms of memory depending on the implementation on the PE it does not change its memory fingerprint. But in the case of the ME, with this new approach we will require memory at least for one Time Window, plus for the new flows descriptors during the same time window.

This amounts to:

$$\mathcal{M} = |A_D| \cdot \mathcal{N}_t + |A_{ID}| * \mathcal{A}_t + (|P_{ID}| + |o|) * \sum_{i=1}^{n} \mathcal{P}_{t_i} \tag{6.3}$$

The expression is similar to the case of bandwidth requirements but expressed in bytes.

6.2 Experimental Evaluation

With the goal of validating the feasibility and resource consumption of NPAS, we performed a series of tests in different real scenarios. Such tests aim to prove that the system can be deployed with bearable resource consumption. The tests also state experimentally the effects of the different parameters in IDRP (t and n).

6.2.1 Testing environment

Two different sets of tests have been performed with the goal of, *i)* Experimentally evaluating the values t and n presented above. *ii)* Estimating the real bandwidth used by NPAS on a trace obtained in a real backbone.

This evaluation is performed by using two different testbeds, each one achieving one of the above objectives:

EuQoS testbed In order to experimentally obtain suitable t and n values we used the testbed provided by the IST EuQoS project [45].

The EuQoS project, besides developing a solid infrastructure for end-to-end QoS provision, provides a European wide testbed which allows us to test different technologies, resulting in a representative scenario where to deploy NPAS.

There are currently eleven different local testbeds which are interconnected via the Géant network and *National Research and Education Networks* (NRENs) through private tunnels whose topology form a configurable mesh. Each partner's testbed uses different network technologies. Such as: UMTS, xDSL, Ethernet, Gigabit Ethernet and WiFi (802.11).

For the purposes of this work, the whole testbed will be considered as a single MD.

Backbone traffic collection Although the EuQoS testbed is suitable for testing under controlled traffic loads, to experimentally evaluate the scalability of the proposal, more knowledge of operational traffic from a real network is required.

Hence, to estimate the used bandwidth of the system we performed a full link collection in different hours on a collection point located in a vantage Gigabit Ethernet link on the backbone of the Spanish and Catalan NREN. This point permits to compute and analyse traffic characteristics such as number of packets, number of flows and amount of new flows per time unit.

We do not detail here the traffic characteristics since this trace is the same we already used in Section 5.4.2.3.

6.2.2 Experimental parameter selection

We used some real network information to have a clearer idea of typical end-to-end OWD characteristics of the network. Hence, we performed a set of 520 tests from January until December 2006 using the EuQoS testbed. The tests were composed by a set of combinations of different packet rates, packet sizes and daytime and nighttime tests. A broad range of packet loss and delay variation conditions were encountered given the different cross-traffic found on the network at different days, hours and physical locations all over Europe. That gives good range of one way delays to give proper insights for choosing t and n in the network.

The tests were performed by actively generating controlled traffic into the EuQoS network and computing all the end-to-end OWD. Later these traces were processed off-line to model the OWD characteristics of the network depending on different t and n values.

As we already discussed, t and n set the thresholds for packet losses. Figure 6.4 shows the percentage of packets out of the window due to late arrivals for each t and n. For ease of exposition, the figure only shows the results for $n = 1,3,5$.

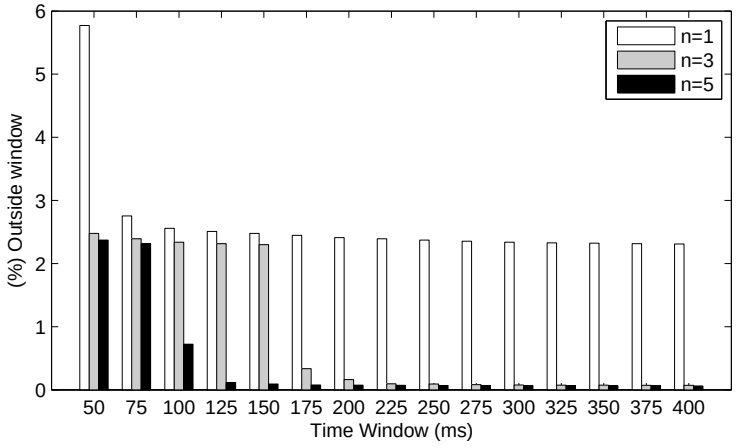


Figure 6.4: Error in packet losses count for $n = 1,3,5$

The results show a non-negligible amount of packets with very high delays (higher than 400ms). This is caused by congested xDSL and UMTS links found in some testbeds.

In order to guarantee that most of the packets are within the time window we computed the 99.5 percentile of delay for all the tests, obtaining a value of 509ms. As discussed before $H = t \times n$ and for assuring a smaller error than 0.5% $H \geq 509ms$, we also need $n \geq 2$ because with $n = 1$ the system suffers of the TW alignment problem exposed before. With these constraints the optimal values for our tests are $n = 3$ and $t = 175$. Giving a $H = 525ms$.

Moreover, observing the figure, it also shows the big jump on the delay distribution between $t = 150ms$ and $t = 175ms$ for $n = 3$ which passes from 2.2% of packets out of the window to less than 0.5%. Another important conclusion is that for $n = 5$ the small improvement is not worth maintaining the window respect to $n = 3$ in this particular case.

6.2.3 Bandwidth usage

Apart from the percentage of losses due to TW's size, the lower is t the higher is the overhead caused by the reporting. This overhead is caused by sending A_{ID} for each reporting block. The bandwidth used by this control traffic depends on the packet rate, new flow rate and number of flows per t.

To have better insights about the used bandwidth when deploying our solution, typical traffic characteristics must be known. We perform this evaluation with the trace we got from the Spanish NREN as described previously.

The results given in this section assume the unrealistic scenario where all the traffic is under QoS restrictions, so the obtained results are strict upper bounds of the real cost of the solution in such a network.

Table 6.1 summarises the bandwidth required for each t both for the Flow and CoS based aggregation. Values represent absolute bandwidth per ME on the MD. The values on the table show the average cost when the system reached the stationary state without including the additional cost of registering all the flows by sending F_D to PE during NPAS startup. In our experiments during such startup, the maximum new flow rate grows linearly from 1315 to 5756 for 50 and 400ms respectively.

Using the original per packet reporting as discussed in Chapter 5 with this data, the control traffic generated is 6.5$Mbps$. Using the aggregation technique presented here, it represents a reduction in the worst case of 27% of control traffic overhead over the per packet reporting.

Using CoS aggregation delivers a control traffic reduction higher than 1$Mbps$. This reduction is caused by two different factors:

t (ms)	N. Flows	N. Pkts	New Flow Rate	BW per flow	BW per CoS
50	1336	2870	101	4.74	3.68
75	1856	4305	152	4.68	3.68
100	2325	5740	202	4.63	3.68
125	2750	7174	253	4.59	3.67
150	3142	8609	304	4.55	3.67
175	3504	10044	354	4.52	3.67
200	3839	11478	405	4.50	3.67
225	4145	12913	455	4.47	3.67
250	4432	14348	506	4.45	3.67
275	4701	15783	556	4.43	3.67
300	4959	17217	607	4.41	3.67
325	5206	18652	658	4.40	3.67
350	5442	20087	708	4.38	3.67
375	5671	21522	759	4.37	3.67
400	5893	22956	809	4.35	3.67

Table 6.1: Used bandwidth per bin (Bandwidths in Mbps)

1. A_D is not generated as the CoS are decided in advance.

2. In a t period there are a non negligible amount of flows, which require a F_{ID}, as we discussed in figure 6.1, while CoS IDs are negligible.

It is worth to note that with the traffic reduction resulting from this optimisation and assuming the presence of 10 ME, the used bandwidth simbolise 9% of the present traffic on the links considering per flow reporting. This means a reduction of around 50% compared to the basic per packet reporting.

In the case of CoS reporting the reduction in required bandwidth lowers to 6.98% of the total existing traffic, a noticeable reduction as expected.

6.3 Conclusions

In this chapter we focused on the lossless optimisation of the required resources of the system. The most representative contribution of this part of the work can be found at [107]. We were

able to reduce drastically the required bandwidth to on-line reporting of the QoS metrics. This new solution impacts on the performance only in two different aspects: *i)* there is a lag upper bounde by t time units on the reporting to the PE, and *ii)* the ME requires more memory than the previous approach where no information was temporarily stored on the collection point.

Nevertheless, with this solution we have a very good alternative to the original approach as we experimentally demonstrated. In the next chapters we will try, using as a base this Time Reporting, to further reduce the required resources, this time with novel Traffic Sampling techniques.

7

Sampling based reporting

As we have seen, Time based reporting is a good optimisation in respect to the per packet reporting. Moreover it provides a lossless optimisation of the resources which is very desirable in this context. Nevertheless, this bandwidth requirements can be a problem in high speed links given the, still, large amount of bandwidth required. In this Chapter we propose a static sampling solution which, by trading-off some accuracy, permits to further reduce the required resources. For the sake of clarity, we do not introduce in this chapter the more advanced and tightly related Adaptive Sampling solutions, which we will detail in Chapter 8.

7.1 Background

A brief introduction about traffic sampling has been done in section 4.3.3, but given the specific constraints our system is tied to, it is in order to detail a bit more the relevant related work, and the required existing mechanisms, which we adjusted to our distributed reporting system.

Traffic sampling for network metric estimation is a topic covered many times in the past (e.g [37; 124; 130; 131; 132]). Sampling can be applied to a broad range of fields with the goal of reducing the needed resources to compute a given network characteristic.

Such efforts on sampling Internet traffic evolved on *Packet SAMPling* (PSAMP), an IETF Working Group in charge of defining and standardising the different sampling techniques applicable on network measurements.

Of all the different approaches proposed by PSAMP, the chosen solution must provide a deterministic packet selection function used to match the packets on each collection point. A

possible solution was first presented by Duffield et al. [38], where the authors present a methodology for inferring the packet's trajectory by using *hash sampling* that particular technique was called *trajectory sampling*. The solution reduces the analysed traffic by using sampling together with a hash function over some selected fields on the packet's header. The main difference between trajectory sampling and our approach is that, while trajectory sampling uses hash based sampling in order to decide which packets to select regardless of its origin, we must consider all the flows within SLA contracts, and hence we must extend the usage of hash sampling. Specifically we define a two level hash table to overcome this limitation.

In the distributed QoS metric analysis field another sampling approach is used in [29], where well-known traffic sampling methodologies are used for computing one-way delays and packet losses in ATM networks. The caveat of that approach is that it relies on the information stored in the ATM cells, not permitting its application in other technologies. Moreover, their solution only considers scenarios with two static monitoring points on the network, while our proposal goes one step further giving a generic solution for technology independent intra-domain QoS parameter acquisition.

7.2 Static Sampling

The complexity of applying sampling in this distributed scenario is that centralised techniques (e.g. the techniques presented in [124]) are not well suited for this task, since they do not guarantee that all MEs capture exactly the same set of packets. Thus, accurately determining packet losses or one way delays is not feasible. We solve this issue by defining a deterministic sampling technique to match exactly the same packets all over the different ME, and later computing the network metrics. This technique permits all the different ME to collect traffic independently, only the selection function at configuration time has to be shared by the MEs.

Hence, once again, we extend the IDRP ($E - IDRP_2$) as shown in Figure 7.1. As it can be observed, we insert a new abstraction layer on the ME, which is in charge of selecting the chosen packets to be reported depending on the sampling function. On the other side, the PE must infer the real statistical values out of the applied sampling.

As in the case of Time Based reporting, we can fallback to the simpler reporting mechanisms.

The distributed sampling method proposed in this chapter is based on the hash sampling technique proposed in [38]. Hash sampling computes a hash function over a set of fields on the

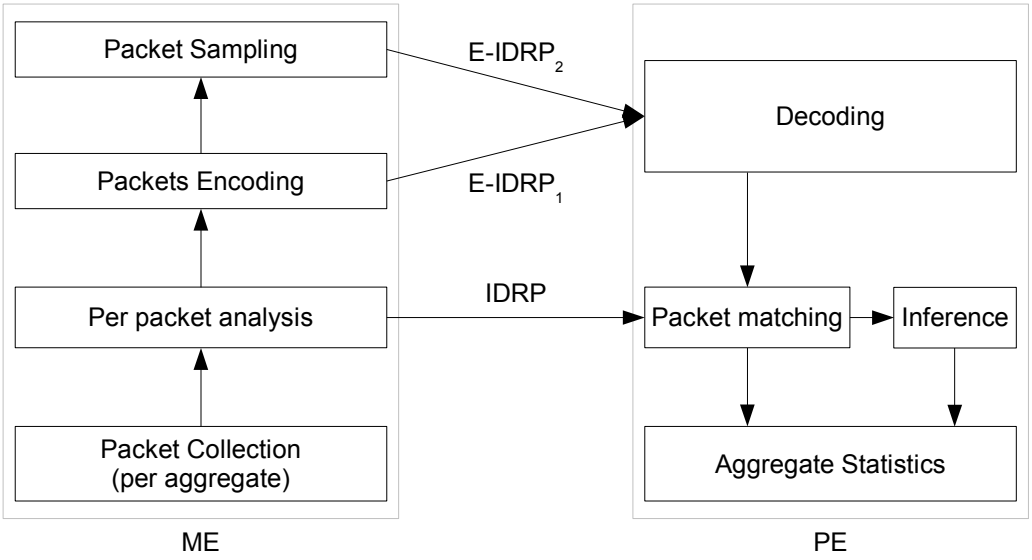


Figure 7.1: Block view of $(E)IDRP_2$

packet's header. Thus, a packet is only analysed if it falls within specified positions in the hash table. This permits to efficiently control the resources and the sampling rate of the solution. In order to adapt this technique to our environment, using only one hash table is not sufficient, since the QoS monitoring framework has to guarantee that all the flows under SLA contract are considered for the analysis. Thus, the sampling must be applied within the flows, not directly to all the collected traffic. Therefore we identify the packets using two different keys, the *flow identifier* and the *packet identifier*.

7.2.1 Hash functions analysis

The above requirements are implemented using a two level hash table as shown in Figure 7.2. The flows and the packets tables are indexed by two different hash functions, one that uses the *Flow identifier* and the other with the *Packet Identifier*.

7.2.1.1 Flow Identifier hash function (F_H)

The flow identifier (F_{ID}) has 32 bits and it is obtained using the typical 5-tuple for flow selection: Source and Destination Addresses, Source and Destination Ports and Protocol.

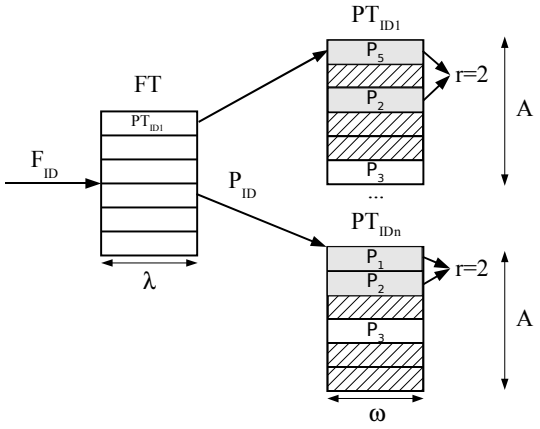


Figure 7.2: Structure of Flow (FT) and Packet (PT) Hash Tables

The flow identifier is hashed by a randomly generated H3 hash function [24], which distributes the flows uniformly and unpredictably. This way, we guarantee minimum collisions in the first hash table. If a collision is found, a linked list of the colliding flows is created. In our context we must resolve collisions on the flow tables, since the system must consider all the flows under SLA constraints.

7.2.1.2 Packet Identifier hash function (P_H)

The packet identifier (P_{ID}) has 32 bits, and it is generated by a CRC32 from 27 bytes of the packet's payload, as explained in Section 4.3.3.

Using both functions presented above allow us to quickly insert each desired QoS information. Each flow hash table has a size of A packets, from where only the first r are considered (see Figure 7.2 where we suppose $r = 2$ as an example). Possible collisions on the second hash table are ignored (the new packet silently overwrites the previous one). Hence, a parameter to consider is the size of the packet hash table. As we detail in the results, for fairly small hash tables collisions are rarely found.

When a new packet arrives to a ME it is classified to the flow where it belongs, the particular Flow hash Table (FT) entry indicates the specific table (PT) where the packet must be stored. Then P_H is computed and the packet information is stored into the second hash table.

7.2.2 Applying the Sampling

Once we have all the information stored into the hash table, the last step is to choose which packets will be sent to the PE. In this approach we consider that the sampling rate is applied uniformly within all the ME involved on the measurement. Let ρ be the upper bound of the sampling rate. In some situations ρ might not be applicable, since the number of packets received by ME is discrete, then the minimum applicable sampling rate to flow f is $\rho_{(f,1)} = \frac{1}{R_f}$, where R_f holds the number of packets received during the time interval in flow f.

Analogously, the maximum sampling rate is achieved by $\rho_{(f,R_f)} = \frac{R_f}{R_f} = 1$. Hence all the possible sampling rates for flow f are: $\mathcal{R}_f = \{\rho_{(f,1)}, \ldots, \rho_{(f,R_f)}\}$. Then we define the applied sampling rate to f as, $\rho_{(f,i)}$ which $\forall i : 1 \leq i \leq R_f$ is closest to the particular ρ.

Hence, the total effective applied sampling ρ_e is defined as

$$\rho_e = \frac{\sum_{i=1}^{|f|} r_i}{\sum_{j=1}^{|f|} R_j} \tag{7.1}$$

where $|f|$ is the number of flows among all the MEs and $r_i = \rho_i R_i$, the considered packets in flow i for the study. Therefore ρ_e defines the effective sampling rate on the system, and thus the real required bandwidth for the control traffic. We must highlight that this value depends on the traffic existing on the network, thus not known in advance.

The application of the sampling rate to the packet's hash table must be done in a deterministic manner in order to synchronise the different ME. Therefore, we use the same t time window defined in the previous chapter. Now t triggers the selection of the packets and flushes the hash tables. The sampling rate is applied by selecting the first r_f packets of each monitored flow on all the MEs. r_f is obtained from the applied sampling rate ρ_f and from R_f: $r_f = \rho_f R_f$.

Note that the hash tables on each ME will have exact copies for the common flows except when there are packet losses, that is when a packet appears on some ME and not on the others down the path. After t time units we transfer r_f packets of each table towards PE, which matches all the packets over all the MEs reporting the QoS metrics.

Moreover, using this solution we can speed up the packet matching performed on the PE, because we know the exact position of each packet in the hash table for all the ME, given that it is the same for all of them (or we have a packet loss).

Another point to consider is when the OWD between two ME is big, it can happen that some collected packets on the first ME have still not reached the second, to avoid this, the PE holds an historic of some time windows t. Analogously to the parameter n described before.

7.2.3 Memory and Bandwidth Requirements

Since we do not know in advance if the hash table will be full, or if there will be empty entries on it, we have to allocate memory for A entries for each P_{ID} to chose the first r_f. Assuming a flow hash table with f active flows, λ bits per F_{ID} and with ω bits per P_{ID}, the overall memory Θ required per ME by the system is upper bounded by expression 7.2:

$$\Theta \leq \lambda f + \omega f A \tag{7.2}$$

Where λf is the size in memory of the flow hash table, ωf is the size of one packet hash table. In our system $\lambda = \omega = 32bits$ which produces $\Theta \leq 32f(A+1)$. Where Θ depends on the active flows f and the size of the hash table A, and not on the actual packets traversing the ME.

Regarding the bandwidth, using small r_f implies less required bandwidth, lower sampling rate and less precision for the results, as we have less information to estimate the real values.

Moreover, t determines the flushing period of the hash table and thus bounding its reporting interval. In the next section we discuss the appropriate value for A and t in a real scenario along with the experimental memory and bandwidth requirements of its deployment.

7.3 Methodology and Experiments

For validating the proposed sampling mechanism we used the same set of tests provided by the EuQoS project and described in 6.2.1. For this study the testbed is configured to act as a single domain, with one ME deployed on each technology, amounting to 12 ME and 1 PE.

In order to ease the validation, instead of directly sampling the traffic, we collected the complete trace and applied the different sampling rates off-line. For the sake of the exposition we only show results for the following sampling rates: $\rho = 0.35$, 0.15 and 0.05, that correspond to effective rates (using expression 7.1) $\rho_e = 0.31$, 0.11 and 0.04, which in our opinion represent examples of low, medium and high sampling rates. With this methodology we were able to compute the metrics for each sampling rate and its accuracy by taking as reference the complete original trace. Therefore we can compute the accuracy of our solution.

In summary the methodology for validating the accuracy of the technique consists of the following actions:

1. Generate the test traffic and gather all the traces on MEs.

2. Apply off-line the different sampling rates ρ_f with the chosen bin size t. This is the information sent to the PE.

3. At the PE, for each bin and ρ_e the average OWD is computed.

4. Finally each sampled OWD result is compared with the original, as detailed later in this section.

As a second set of experiments, in order to assess the costs and the memory requirements of a deployment on a real network, we used again the same trace from the Spanish NREN described in Section 6.2.1.

7.4 Experimental Results

In this section we validate the proposed sampling mechanism by using the previously described testbeds. We detail the experimental tests, the selection of A and the accuracy obtained by applying several sampling rates to the traces. The analysis is completed by the study of the real memory requirements of the solution.

We use the same selected value of t in the previous chapter: $t = 175ms$.

7.4.1 Selecting A

One of the basis of our infrastructure is the use of a hash table. The size of the table impacts directly to the system's performance. On the one hand, the smaller is the table the less memory the system requires to operate. On the other hand, the bigger is A the lower is the collision probability. We have to point out that having collisions is not a big issue because we are using sampling. Therefore we will not consider all the packets for the analysis, as long as we have r_f packets to study per bin and per ME. So we advise using $A \gg max\{r_f\}$ for any flow on the system.

In order to evaluate the effects of such table size we used the Catalan Academic Network traces described before. There the results show that typically on a network where $t = 175ms$ we have an average of ~ 3500 flows with a total amount of ~ 10000 packets per bin. It results on an average of ~ 3 packets per bin on each flow, but with a maximum packets per bin of 1440 belonging to bulk transfer flows. This values are upper bounds *in our scenario*, since we consider all the traffic on the link, while for SLA assessment we would only consider part

of the traffic. Anyway we think that the analysed traces are quite representative of the normal behaviour on the Internet.

We used this information to study the effects of different hash table sizes, specifically we choose different prime numbers $A = 19, 73, 101, 647, 1297, 3187, 7919$ and 16979. Using prime numbers together with a robust hash function is advisable in order to minimise the collisions [38]. From the tests we computed several statistics as shown in Table 7.1. The table details the probability of having a bin with a collision, in our case it is $\sim 30\%$ for the minimum size and is rapidly reduced down to 0.01% in the case of the biggest considered A. From the bins with collision we computed its maximum and average length (columns 2 and 3 respectively), for example with size 1297 in the worst case we only have 143 collisions, while from all the bins with collision their average is only 1.34. As it can be noted the gain of further increasing A from 1297 is not worth the cost increase in terms of memory, since we have a very small amount of collisions.

A	*W/ Col.*	*Max. Col*	*Avg. Col*	*Max. Rate*	*Avg. Rate*
19	0.7	161	6.85	0.89	0.32
73	0.51	112	3.34	0.63	0.14
101	0.45	100	2.87	0.56	0.12
647	0.15	24	1.52	0.50	0.07
1297	0.08	15	1.34	0.33	0.06
3187	0.04	12	1.20	0.33	0.05
7919	0.02	5	1.12	0.25	0.04
16979	0.01	5	1.09	0.17	0.04

Table 7.1: Collisions summary for different sizes of A

In the table, the first column reports the collision probability for each tested size, it shows the fast decreasing of the collision probability as the table size increases. The rest of the columns only consider the case where there has been any collision, removing the cases where no collision occurred, this is the most common case for tables bigger than 100 packets. In the second column we show the maximum collisions in a bin, which again is greatly reduced by A. The third column indicates the average collisions per bin, which is very low as the table size grows. Columns 4 and 5 show the maximum and average ratio of collisions, that is, the number of collisions in respect to the whole amount of packets, only considering bins with at least one

collision.

In [38] the authors recommend to use $A = 16979$, with our strategy we can reduce this value because we classify the packets into flows, reducing this way the amount of packets entering each hash table. Therefore the rest of the chapter assumes $A = 1297$.

7.4.2 Validation

We focus the validation of the platform with OWD accuracy analysis. As a complementary study we analyse the drawbacks of the proposed technique with the PLR accuracy and the possible solutions.

7.4.2.1 One way delay analysis

The study of the OWD accuracy is performed by comparing the estimated values using sampling with the real results of the full trace. Therefore, all the analysis computes relative error values as detailed on equation 7.3:

$$\varepsilon_S = \left| 1 - \frac{\hat{x}}{X} \right| \tag{7.3}$$

Where X is the real value taken from the complete trace and $\hat{x}$ stands for the estimate obtained by applying sampling.

Expression 7.3 is applied to all the averages obtained from the actions described previously. Figure 7.3 presents the Cumulative Distribution Function (CDF) of the relative error of *One Way Delay*. In the figure the X-axis shows the relative OWD error and the Y-axis holds the cumulative probability. Table 7.2 complements the results with the most relevant percentiles.

ρ_e	99th prc.	95th prc.	50th prc.
0.04	0.16	0.04	0.001
0.11	0.08	0.02	~ 0
0.31	0.03	~ 0	~ 0

Table 7.2: Relative error Percentile for OWD

As expected, the more we increase the sampling rate the better is the accuracy. In the worst presented case with $\rho_e = 0.04$, the error in estimating the $95th$ percentile of OWD only shows 4% error, while for $90th$ percentile it is further reduced to 2%.

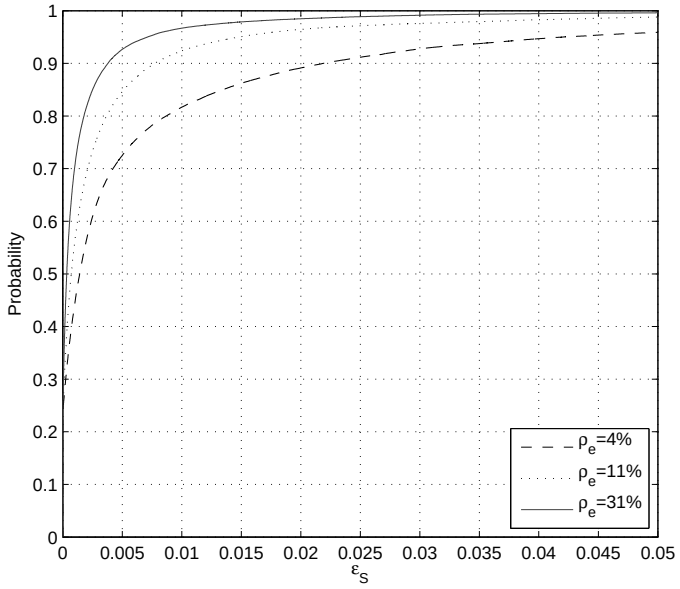


Figure 7.3: CDF: relative error of the sampling estimate

In a SLA controlled scenario admissible OWD for typical real-time applications range from 50 to 200ms (as in the case of VoIP communications [59]). Having relative errors lower than 2% give a usable estimate in order to decide whether the quality is within good thresholds.

7.4.2.2 Packet loss analysis

Analysing packet losses is a much more complex issue than OWD. The reason is the sporadic and discrete nature of a loss. To ease the analysis here we consider packet loss ratios instead of the canonical value. This permits to compare the results of packet losses for tests with different rates, we compute the loss ratio for each sampling rate using expression 7.4:

$$\widehat{PLR} = 1 - \frac{\hat{x}}{r} \tag{7.4}$$

Where r stands for the sampled packets of each p for the whole bin, and $\hat{x}$ is the loss estimate in number of packets obtained for p.

Then the absolute error is computed taking the full trace as reference using $\varepsilon = |PLR - \widehat{PLR}|$.

Another consideration is the case of zero packet losses on a test. This case clearly biases the results, since the packet loss ratio is always correctly estimated regardless of the applied

sampling rate. All the tests with no packet losses are withdrawn from the analysis, leaving a total of 194 tests with losses. Figure 7.4 displays the CDF of those 194 experiments. It can be noted that 95% of the tests have an error higher than 20% for 31% sampling rate, while 90% of the analysed cases fall around 15%.

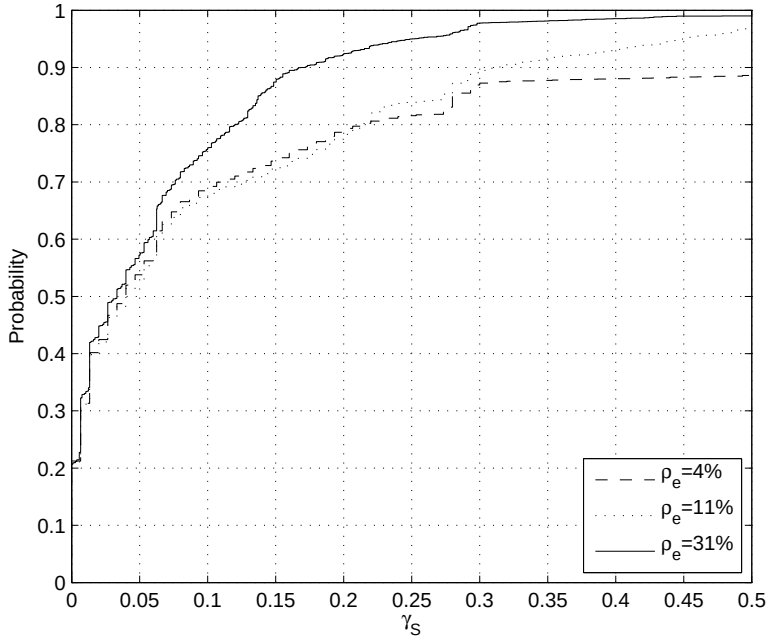


Figure 7.4: CDF: Absolute error in packet loss ratio respect to the real value

For lower ρ the results are much worse. Given the importance of accurate estimation of the PLR, we can assess that this mechanism requires some improvements. We address the issue in the next chapter.

7.4.3 Memory and Bandwidth analysis

With the proposed sampling technique, we reduce the required control traffic proportionally to ρ_e. Specifically we can reduce the cost of the packet reporting, but not the aggregate reporting, therefore:

$$BW = |A_D| \cdot \mathcal{N}_t + |A_{ID}| * \mathcal{A}_t + (\rho_e |P_{ID}| + |o|) * \sum_{i=1}^{n} \mathcal{P}_{t_i} \tag{7.5}$$

That is the same expression as 6.2 but weighted with the sampling rate on the cost of the packet reporting.

The associated cost of using this approach is the increase in memory requirements of the solution. Each ME must hold the hash structure presented before. We studied the memory requirements of deploying this solution on the Catalan Academic Network assuming the traffic conditions detailed previously.

With 3500 flows per bin, we need $\Theta \simeq 18 Mbytes$ on each ME for the hash table. This value has been computed using equation 7.2.

The amount of required memory on the PE depends on each r_f and on the amount of flows and ME. Assuming the above network with 12 ME with 20 packets per bin and $\rho_e = 0.11$, then $r_f = 2$ hence the PE needs $\sim 328 Kbytes$. Only considering the storage of the packets, the whole cost must also reckon the cost of the aggregate descriptors as we already discussed on the previous chapter.

7.5 Conclusions

In this chapter we discussed thoroughly the application of distributed sampling in order to estimate the OWD. The main contribution related to this chapter can be found at [108].

During the study we discovered that estimating PLR is a much more complex issue, the main problem comes from low rate flows, where our system collects few packets per bin. It is well known [26] that the achieved accuracy (within a 95% confidence interval) when classifying sampled traffic into categories is bounded by $\varepsilon \leq 1.96\sqrt{\frac{1}{r}}$, where r is the amount of sampled packets within the category (packet losses in our case).

The above equation assumes normality in the distribution of the samples, but it is feasible to use it if we have randomness in the sampling process, which we have with hash sampling. As a matter of fact, with the sampling methodology presented previously, the selected packets are unpredictable in advance since the selection depends on packet contents, hence the process complies with the restriction, as guarantied with hash sampling [37].

Nevertheless, estimating PLR with the above technique leads to inaccurate results. Hence this technique is not suitable for the estimation of this metric. In the next chapter we discuss some improvements over this misbehaviour.

8

Adaptive Sampling reporting

This chapter presents two different improvements respect to the static sampling solution introduced in the previous chapter. Both improvements use the same basic ideas about resource sharing. That is, to detect unexpected behaviour based on some metrics. We analyse IPDVs in one alternative and PLR on the other in order to get a novel adaptive sampling algorithm that gives two advantages over the static solution *i)* bounded and controlled use of resources, *ii)* much better estimation of packet losses.

The description of this novel approach is preceded by the deep study of packet loss behaviour in nowadays networks. This knowledge will be used later in the adaptive sampling solution. And it is presented here both as a contribution [110] and as a necessary background.

8.1 Packet Loss Analysis

Packet losses are an important metric when assessing network performance. Most of the previous work [17; 44; 119] proof that packet losses usually have a bursty nature, with a non negligible amount of sporadic packet losses.

Bursty losses degrade the quality to a higher degree than sporadic packet losses, since application's correcting algorithms do not work in such situations [59]. In [17] and in RFC-3357 [71] the authors define a loss burst as the sequence of consecutive lost packets. With this strict definition it is possible to detect lossy periods, but in a congested environment many of those loss periods can be chained together, with few successfully transmitted packets, forming a longer time period with poor network conditions. It is advisable to take into account such

periods for improving the PLR estimation in our algorithm. Hence we propose to generalise the concept of bursty losses by using density thresholds as explained in this section.

8.1.1 Definitions

In a congested link, when using a common drop tail queue, the last packets arriving a node are discarded. But while the queue is emptying there will be room for accepting new packets, at least until the queue is full again. In this situation there would be several packet loss bursts with few successfully sent packets in-between. This situation is difficult to detect with the classical burst definition because the loss bursts are much shorter than the congestion period. For NPAS this situation must be handled in order to react in time and improve the packet loss estimation in such cases.

Therefore, in this work we extend the classical burst definition: A packet loss burst is the interval of time with a density of packet losses higher than a threshold τ.

Formally, given a stream of numbered packets $\mathcal{P} = \{p_1, \ldots, p_n\}$ belonging to a flow f with sending timestamps $\mathcal{T}_{tx} = \{t_{1_{tx}}, \ldots, t_{n_{tx}}\}$, a packet p_i is considered as lost if it does not reach its destination with a predefined time T; that is when $T \leq t_{i_{rx}} - t_{i_{tx}}$. Then $\ell_i = 1$ if packet p_i is lost and 0 otherwise. Thus the flow stats regarding packet losses can be defined as $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$.

Using the above nomenclature we propose the following definitions:

Definition 2. *Distance $d_{(i,j)}$ between a pair of packets p_i, p_j, is the amount of transmitted packets between p_i and p_j. Where $i < j$, hence $d_{(i,j)} = j - i$.*

Definition 3. *The density $\Lambda_{(i,j)}$ of packet losses between packet p_i and packet p_j is $\Lambda_{(i,j)} = \frac{\sum_{k=i}^{j} \ell_k}{d_{(i,j)}}$.*

Definition 4. *A burst of packets β starting at packet i then is defined by the tuple:*

$\beta = \langle d_{(i,j)}, \Lambda_{(i,j)} \rangle$. *Where $\forall k \mid i < k \leq j, \Lambda_{(i,j)} \geq \tau$ and $\nexists\, z \mid d_{(i,z)} > d_{(i,j)}$ with $j < z \leq n$ and $\Lambda_{(i,z)} \geq \tau$.*

Hence, a burst is composed by the longest list of packets starting at i where the loss density between i and j is higher than τ.

Definition 5. *All the bursts $\mathcal{B}$ for a flow f are defined as $\mathcal{B}_f = \{\beta_0, \ldots, \beta_m\}$ as the list of bursts in a flow.*

With the constraint that any β_i cannot overlap with any β_j for any $j \neq i$.

Using the above definitions, we aim to study the behaviour of packet losses in a real scenario. Moreover we analyse the effects of packet losses in concurrent flows with the same source and destination. This study will help later in our adaptive sampling solution to estimate the packet loss ratio.

8.1.2 Testbed

The used testbed is, as we already have considered in Section 6.2.1, the EuQoS testbed, with the set of 520 tests of synthetic traffic. The experimental tests were performed during 2006 and 2007 using twelve different testbeds across Europe.

The packet losses were studied by actively generating UDP traffic on the network with different properties. Specifically we generated periodic flows, with varying packet rates, from 16 to 900 packets per second among all the involved nodes in the testbed. We used different packet sizes ranging from 80 to 1500 bytes per packet.

8.1.3 Burst study

Since many tests do not have any packet loss, we removed them from the analysis, keeping the 22% of tests with some loss. In some of the performed tests we intentionally generated more traffic than the available bandwidth, in order to obtain some congestion periods for our analysis.

In the tests we identified all the bursts with *Definition 5* and we used $\tau = 0.45$. In this section we do not aim to evaluate the effects of changing τ, we will focus on this in Section 8.3.1.

As expected from the tests most of the losses belong to a burst (around 99%), with an average distance of 182 packets, which indicates an average burst duration of $\sim 180ms$, but with a large standard deviation (around $1100ms$). The distance's 99^{th} percentile is around 6 seconds, this percentile is that high because of the induced congestion described above.

Considering the total amount of lossy periods[1] in all the tests, just $\sim 43\%$ were the beginning of a burst, while the other $\sim 57\%$ were only sporadic losses. This is detailed in Table 8.1. As it can be noted the periods with sporadic losses are higher than those for bursts. Later we will use this property in order to not overreact in the presence of sporadic losses.

[1] A lossy period starts at the first packet lost after a period of no losses and ends when the loss burst is finished (or immediately in the case of sporadic losses).

	Burst	Sporadic
Bursts in lossy periods	43.3%	56.7%
Packets in burst	99.2%	0.8%

Table 8.1: Details of Burst in the experimental analysis

But the most surprising outcome of this analysis is that counting the total amount of packet losses as a whole these $\sim 43\%$ in reality represent the $\sim 99\%$ of the total lost packets. Meaning that once in a burst, the probability of having lots of losses is very high.

8.1.4 Loss behaviour among flows

As we discussed before the accuracy of any sampling process is bounded by Equation 8.1:

$$\varepsilon \leq 1.96\sqrt{\frac{1}{c}} \tag{8.1}$$

Basically this means that the more packets we collect the better is our PLR estimation, but sometimes we can be analysing low rate flows where after applying the sampling rate maybe just one or two packets are considered per bin. In this situation we can fail to estimate the PLR very easily.

In the following study we aim at proving experimentally that packet losses spread regularly among all the flows in the same congested path.

It is well known from queueing theory that the probability of queueing of a packet depends only on the traffic load in the network [70] and it is independent of the size of the packet or cross-traffic packets. This means that the probability of losing a packet does not depend on the flow to which it belongs. Therefore, during congestion the packets will be lost among all the flows randomly. In this subsection we will analyse this theory empirically.

Then we can use this reasoning to infer that when we detect packet losses on one flow we should increase the sampling rate of all the flows of the path since most probably they are also experiencing losses.

We prove this somewhat regular spreading by selecting three different partners from the testbed presented before. They were located in Warsaw University of Technology (WuT), University of Bern (UoB) and Technical University of Catalonia (UPC), and generated traffic with several flows simultaneously, from UPC to UoB, from UoB to WuT and from WuT to

UPC. Some tests were performed with groups of 10 flows, for others we used 600 flows, each group of flows were sent from the same source to the same destination during a 5 minutes period. The groups of 10 flows had a packet size of 315 bytes with a rate of 200 packets per second. While the groups of 600 flows used a packet rate of 20 pkt/sec and 60 bytes each. The tests were performed at different hours with different cross traffic conditions produced by the Géant network.

The comparison is performed by computing the PLR of each flows and compare the minimum and maximum PLR as detailed in Table 8.2. As it can be noted the PLR bounds even if not exact on each flow within each group are close, with a bit more of variation for the 600 flows as expected due to the bigger amount of traffic.

(a) 10 flows

	Min.	Max.
Path 1	$8 \cdot 10^{-4}$	$9 \cdot 10^{-4}$
Path 2	$1.5 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$
Path 3	$1.7 \cdot 10^{-3}$	$4 \cdot 10^{-3}$

(b) 600 flows

	Min.	Max.
Path 1	$13 \cdot 10^{-2}$	$17 \cdot 10^{-2}$
Path 2	$12 \cdot 10^{-2}$	$18 \cdot 10^{-2}$
Path 3	$2 \cdot 10^{-4}$	$5 \cdot 10^{-4}$

Table 8.2: Min. Max. PLR among the flows

From the results obtained in this section we conclude that when we detect high density of packet losses in one or more flows within a path, the probability that all the flows in the path experience similar conditions is very high. Hence it is worth increasing the sampling rate of *all* the flows sharing that path. This is an important property to exploit in order to improve the PLR estimation accuracy.

8.2 Adaptive Sampling solution

Using traffic sampling in order to reduce the required resources is usually a good option. But choosing the optimal sampling rate with minimum loss of accuracy is not straight-forward.

Moreover, knowing that control traffic (the reporting information from MEs to PEs) must be limited on the network, it might force the per flow sampling rate to be further reduced. Hence, selecting the right flows to change the sampling rate leads to better results than just equally sharing the sampling rate among the flows.

The methodology presented here permits to efficiently estimate the One-Way Delay and Packet Loss Ratio using adaptive sampling and at the same time to have control over the used resources.

Before detailing the approach, in order to fit this new structure on our system, we must, once again, upgrade our IDRP to manage this new mechanism. In Figure 8.1 we show the new blocks over the previous version. The traffic sampling technique is not modified, since it worked properly. But now we insert the resource manager, which is the PE block that decides the amount of resources to give to each ME. The following section describe possible techniques to achieve this.

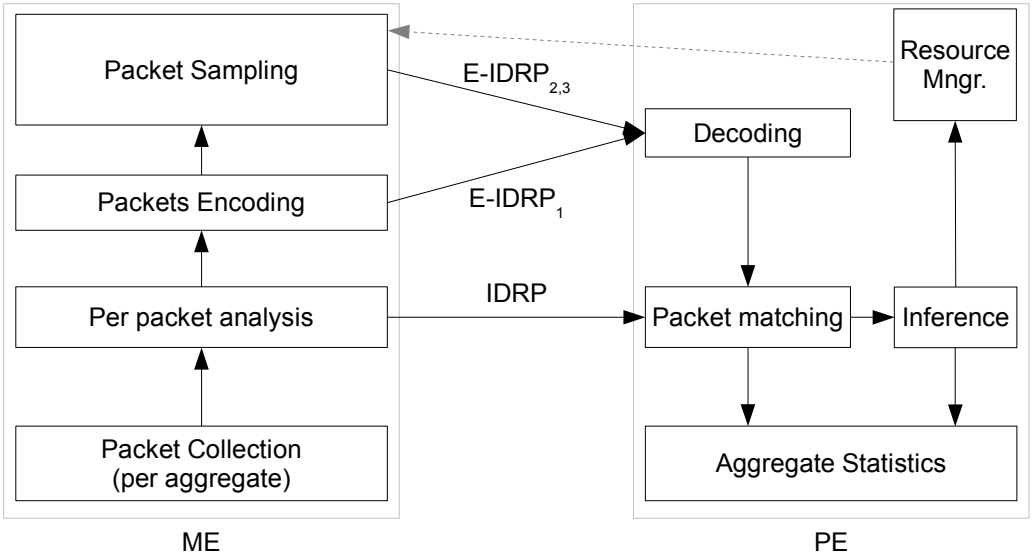


Figure 8.1: Block view of $(E)IDRP_3$

8.2.1 Problem Formulation

Given a domain $\mathcal{D}$ with $\mathcal{R} = \{r_1, \ldots, r_n\}$ ingress and egress points of the network, and $\mathcal{F}_\mathcal{D} = \{f_1, \ldots, f_m\}$ the list of active flows within the domain. Then $R_{f_i}(t)$ (R_i from now on) is the rate of the i^{th} flow at time t, being n the number of ME and m the number of flows.

For simplicity we consider throughout the chapter that the resources for a ME is the bandwidth required to send information to the PE.

The resources required to monitor all the existing flows from ME i to ME j are

$$x'_{ij} = \mathcal{S} \sum_{k=1}^{m} \delta_{ijk} R_k \tag{8.2}$$

where $\mathcal{S}$ is a constant defining the amount of resources needed to process (and send) a single control packet, for example the number of bytes. As it is a constant, for the ease of exposition we will assume $\mathcal{S} = 1$. And $\delta_{ijk} = 1$ if f_k goes from r_i to r_j and 0 otherwise. From this we derive that the resources required for each ME (X'_i) are $X'_i = \sum_{j=1}^{n} \left(x'_{ij} + x'_{ji} \right)$. Hence, reporting in a per packet basis implies that the total amount of resources (X) required for on-line monitoring in domain $\mathcal{D}$ is: $X_{\mathcal{D}} = \sum_{i=1}^{n} X'_i$.

In general per packet reporting requires too much resources to be feasible. We ease this requirement by applying adaptive traffic sampling to the solution. We consider that $X_{\mathcal{D}}$ are the global resources available to perform the collection and performance assessment. We need a fair share of the resources among all the ME, considering that ME with more load (e.g. with more number of monitored flows) deserve more of the resources. Hence, we need to adapt the per flow sampling rate (ρ) in order to comply with this restriction:

$$X_{\mathcal{D}} \geq X \geq \mathcal{S} \sum_{i=1}^{m} \rho_i R_i \tag{8.3}$$

where ρ_i is the sampling rate $(0 \leq \rho_i \leq 1)$ of flow i, R_i the rate in packets per second and X the maximum resources reserved for the monitoring. Note that R_i (in packets/sec) is known since the ME initially must collect all the traffic to decide whether it falls within the specified threshold set in the hash table as described in Chapter 7.

We need a lower bound of the sampling rate, namely $\rho_{i_{min}}$, which guarantees that it is adjusted depending on the rate in a per flow basis, since low rate flows at small time scales are very sensible to sampling rates. This lower bound determines the minimum required resources for the whole monitoring task. Thus,

$$X_{min} = \sum_{i=1}^{m} \rho_{i_{min}} R_i \tag{8.4}$$

This works well if the traffic has constant rates, but since traffic in general is variable these values need to be updated periodically. Moreover, new and expired flows are bound to appear over time. Both the new and expired flow management, and the update strategy will be discussed later.

The minimum applicable sampling rate has to take into account the rate of the flows, limited by expression $\rho_{i_{min}} = \frac{1}{R_i} \quad \forall i \setminus 1 \leq i \leq m$.

Where $\rho_{i_{min}}$ is the minimum applicable sampling rate for all f_i. This guarantees at least that one packet per flow is captured. If c_{min} is the number of samples per f_i then,

$$\begin{aligned} \rho_{i_{min}} &= 1 \quad \forall i \setminus c_{min} \geq R_i \\ \rho_{i_{min}} &= \frac{c_{min}}{R_i} \quad ,otherwise \end{aligned} \tag{8.5}$$

when more precision is required or more resources are available, assuring that at least c_{min} packets per flow are considered.

In the same way we can define the maximum sampling rate to

$$\begin{aligned} \rho_{i_{max}} &= 1 \quad \forall i \setminus c_{max} \geq R_i \\ \rho_{i_{max}} &= \frac{c_{max}}{R_i} \quad ,otherwise \end{aligned} \tag{8.6}$$

where c_{max} determines the maximum number of per flow packets to be collected. Then $X_{max} = \sum_{i=1}^{m} \rho_{i_{max}} R_i$ and $X_{max} \leq X \leq X_{\mathcal{D}}$.

In order to ease the exposition we define $x_{i_{min}}$ and $x_{i_{max}}$ which refer to the minimum and maximum resources for a particular flow respectively.

8.2.2 Resource Sharing

With equations 8.3 and 8.4 we know that after X_{min} are used, the remaining available resources (ΔX) to share among the flows are

$$\Delta X = X - X_{min} \tag{8.7}$$

The goal now is to share the ΔX resources. Let's define ω_i as the weight assigned to the flow i where $\sum_{i=1}^{m} \omega_i = 1$, then the resources assigned to the flow follow equation 8.8.

$$x_i = \min\{\lfloor \Delta X \omega_i \rfloor + x_{i_{min}}, x_{i_{max}}\} \tag{8.8}$$

It can be noted that the resources assigned to the flow are bounded by the X_{max} as assigned previously. Now the weights (ω) may be assigned in different ways, in this thesis we study two alternatives: IPDV driven and PLR driven resource sharing. We will detail both alternatives in section 8.3 and 8.4.

8.2.3 Update strategy

Flow rates vary over time, ideally in our approach we want to know the rate for all flows in advance. Since this is not possible in a distributed scenario, our proposal uses a periodic updating mechanism divided into these steps:

1. *Warm-up phase:* initially the PE assigns x_{min} resources to the system.

2. *Working phase:* in each step (bin of size t) the PE collects the rates and the estimated network metrics and computes the corresponding c_i for each flow f_i as explained before. This c_i will be effective on the next step.

Since the rate (r_i) might vary from one bin to the other, the estimated effective rate considered in the analysis is $\hat{R} = \frac{\sum_{i=1}^{h} R_i}{h}$, where h is the history of the system.

The impact of this on the system is not important, since the bin size usually ranges in the order of tens or hundreds of milliseconds, which implies that there is small variability from one bin to the next. In the experimental analysis we assume $h = 1$ for this reason.

8.2.4 New and expired flows

Since users connect to and disconnect from many services, continuously new flows arrive and others end in a domain. In our system the effects of such behaviour are that expired flows release resources for the reporting, while new flows force a rescheduling of the already shared ones.

In the case that a flow does not have any active packets on the bin, its resources are not used in the current step. Note that increasing the ρ for other flows would lead to incorrect estimation as the other ME receiving (or sending) the flow are not aware of the situation. In the next bin, the flow without active packets will be assigned c_{min} resources and will enter in the *Warm-up phase* again. If a flow stays in *Warm-up phase* during several consecutive bins it is considered as expired, hence it is removed from the resource scheduling.

Another condition to be considered is the apparition of new flows on the system. When this occurs, there is no time for the ME to ask the PE for a rescheduling of the resources, so the ME enters in *local rescheduling mode*.

We know that each ME has X_i resources dedicated to it. Therefore to accommodate the new flow (x_j) the ME has the following alternatives:

1. There are flows without active packets in the bin: in this situation c_{min} resources from that inactive flow are used for the new. Equation 8.8 guarantees that the resources are present.

2. All the flows have active packets: ME (r_j) searches for the $max(x_i)\ \forall i\ x_{1\ldots m}$ and shares its resources in the following way: $x_{i_{new}} = x_i - c_{min}$ and $x_{j_{new}} = c_{min}$. Forcing $x_{j_{new}}$ to enter in *Warm-up phase*.

In the next step the PE can fairly reallocate the resources to fit the new situation without further issues.

8.3 PLR Based

In the previous section we have shown that there are a ΔX (See equation 8.7) free resources to share among the ME. These ΔX resources need to be shared smartly. Obviously it is much more useful to increase ρ_i for flows with less accuracy on the metric estimation. But the estimated error is unknown in advance, so it is not possible to determine the sampling rate deterministically. The proposed mechanism uses the previously acquired knowledge about PLR as a measure of the accuracy of the estimation.

PLR driven estimation is based on the search for lossy paths and adjust the sampling rate of all flows within the path accordingly to its PLR.

The rationale behind this methodology becomes from the fact shown in Section 8.1 regarding the distribution of losses among all the flows on a path. Once a burst is detected on one flow between two ME, we must increase the resources used on estimating the PLR on that path. This way we increase the amount of samples, and thus the accuracy.

We define three different states for paths: *i)* Normal path, *ii)* Path with sporadic losses, and *iii)* Congested path.

A path is normal when $PLR \leq \gamma$[1]. Sporadic losses is when $PLR > \gamma$ and $\Lambda_{(i,j)} \leq \tau$ [2]. Where i and j are the first and the last packets on the bin. Finally if $\Lambda_{(i,j)} > \tau$ and $PLR > \gamma$ then the path is considered as congested.

This avoids overreacting if losses are sporadic (i.e. the sampling rate is increased for the whole path while having sporadic losses), and also underreacting in case of congestion. Here the most critical parameter to avoid under or overreacting is τ as we will show later.

[1] In QoS environments $\gamma \leq 10^{-3}$ usually (See Rec. ITU-T Y-1541 for more detail).

[2] The nomenclature used in this study is taken from section 8.1

The detailed process for weight assignment is shown in Algorithm 1. The algorithm's input is the list of ME pairs with the flow list. This list has the flow properties together with each flow's PLR.

Algorithm 1 Pseudo-code for weight assignment

Input: $ME_{pair}[1..n]$, F_{list} {ME Pairs and flow list}
for all me $\leftarrow ME_{pair}$ **do**
  $plr \leftarrow$ computeMEPLR(me, F_{list}) {$plr = n * max(me_{plr})$ in case of congestion}
  if $plr \neq 0$ **then**
    density = $computeDensity(me, F_{list})$
    $D \leftarrow D \cup \langle me, density, plr \rangle$
    $totalPLR += plr$
  end if
end for
for all $d \leftarrow D$ **do**
  if $d[density] > \tau$ *and* $d[plr] > \gamma$ **then** {We have a congested path}
    $\Omega \leftarrow \frac{d[me]_{PLR}}{totalPLR}$ {Ω is the total weight for the path}
    for all flow $\leftarrow$ allFlowsWithLosses($F_{list}, d[me]$) **do**
      $F[flow]_{\omega_i} \leftarrow \Omega \frac{1}{numberFlows(d[me])}$
    end for
  else if $d[plr] > \gamma$ **then** {We have sporadic losses}
    for all flow $\leftarrow$ allFlows($F_{list}, d[me]$) **do**
      $F[flow]_{\omega_i} \leftarrow \frac{F[flow]_{PLR}}{totalPLR}$
    end for
  end if
end for
Output: F_{list} with all the ω_i initialised

For each ME pair we compute the total PLR in the ME ($computeMEPLR$) as follows:

- If any flow has a $\Lambda_{(i,j)} > \tau$ then all the flows set its PLR to the maximum within the flow, hence the aggregated PLR will be $n * max(PLR)$ with n the number of flows between the ME pair. Here we use the property of uniform spreading of the PLR within a path.

- If the flows just have sporadic losses (i.e $\Lambda_{(i,j)} \leq \tau$) the returned PLR will be $\sum_{i=1}^{n} \ell_i * f_i[plr]$.

From line 10 to the end of the algorithm we fairly share the weights ω proportionally to the *PLR* among all the flows on the congested path. While we only give the proportional share to the flows with sporadic losses, not to the whole path, hence optimising the resource utilisation where is more needed.

8.3.1 Methodology and Validation

This section is devoted to the evaluation and validation of the proposal, we describe the used methodology used for validating the packet loss based distributed adaptive sampling solution. This validation is performed by using the testbed presented in Section 8.1 to show the accuracy of our solution in a real environment.

Moreover, we complete our validation showing the effects on the accuracy obtained by tweaking the density threshold (τ).

The tests were performed by actively generating synthetic periodic traffic as described before. In the validation we study the accuracy in the estimation of PLR and we also compare it with the one obtained by the distributed static sampling.

8.3.1.1 Methodology

To ease the test management, the tests were performed independently one from the others, at different hours, and we collected the full trace of the traffic for later processing. With the obtained traces we emulated a system with the flows entering randomly to the system. We also combined the tests with the 600 simultaneous flows used in Section 8.1.

We defined the maximum number of active flows to 100 plus the 600 parallel flows which were active during all the experiment. Besides, new flows were entering the system using an uniform random distribution with random duration. With this set up we applied off-line the distributed adaptive sampling. Hence we were able to evaluate the results for different resource reservations (X) by taking as reference the complete original trace. We chose for the experiments the following X: $20000, 15000, 10000$ and 5000 with $\mathbb{S} = 1$. These resources are treated as a global property of the tests, and given the total generated traffic among the MEs these X correspond to the following effective sampling rates (ρ): $26\%, 25\%, 22\%$ and 16% respectively.

Another important parameter to the system is the reporting interval (bin), where we assume $t = 175ms$ as we used before.

8.3.1.2 Accuracy of the solution

In order to quantify the accuracy obtained by our solution we performed two different sets of comparisons. First we analyse the effects of the different resources X detailed before, and second we compare the accuracy of our solution with the equivalent ρ of applying static sampling.

As expected, the more resources used for the reporting the more accuracy we obtain on the results. In Figure 8.2 we can see the empirical Cumulative Density Function (CDF) for the different resources and a $\tau = 0.45$. For clarification the figure shows the effective sampling rate since it states the reduction in resources more clearly.

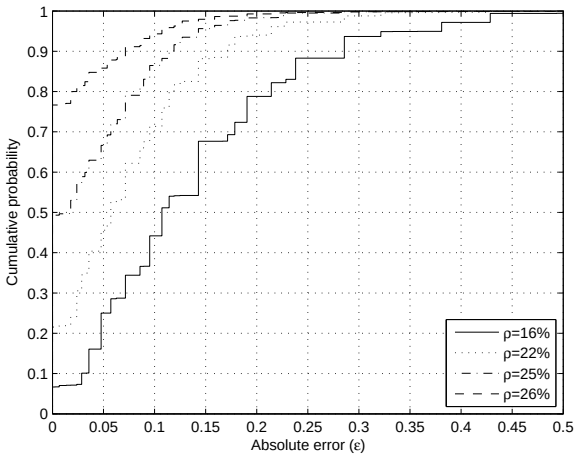


Figure 8.2: PLR estimation error respect to the full trace. Empirical CDF per sampling rate ($\tau = 0.45$)

In the figure it is possible to see that increasing the resources brings a big boost to the accuracy, even if the final effective sampling applied is similar. Just increasing a 1% the sampling rate delivers a much higher accuracy. Table 8.3 details the most important statistics for the study.

As the second study, we apply to the same traces the uniform static sampling and compare the results with our distributed adaptive solution. The differences can be observed in Figure 8.3 where the results highlight the improvement in accuracy of the absolute error, which is clearly noticeable for equivalent sampling rates. The figure details only mean values and in the case of our approach the 95% confidence interval is shown. We do not show the confidence interval

	Mean	StDev.	95^{th} Prc.
$\rho = 16\%$	0.13%	0.10	0.38
$\rho = 22\%$	0.07%	0.07	0.21
$\rho = 25\%$	0.04%	0.05	0.14
$\rho = 26\%$	0.01%	0.04	0.11

Table 8.3: Statistic values for the error with real tests

for the static sampling because it is too large, which confirms the better results obtained with our solution.

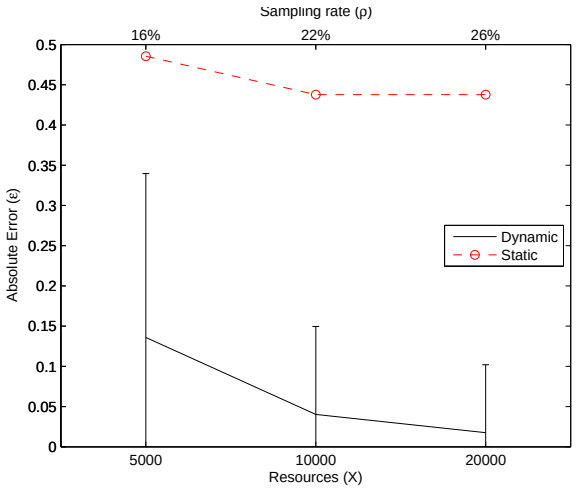


Figure 8.3: Static versus Dynamic adaptive sampling ($\tau = 0.45$)

8.3.1.3 Density threshold

The density threshold parameter τ is very important in this environment. It determines the sensibility of the reporting system when detecting packet losses. The lower we set τ the quicker we will reserve more resources to lossy flows or paths. But on the other hand this can force an over-reservation of the resources to flows with low loss ratio. The opposite is also true, with very high values of τ we take too long to react, hence we are underreacting to a potential congestion.

Figure 8.4 shows exactly this effect. In the figure the X-axis has the different τ values while

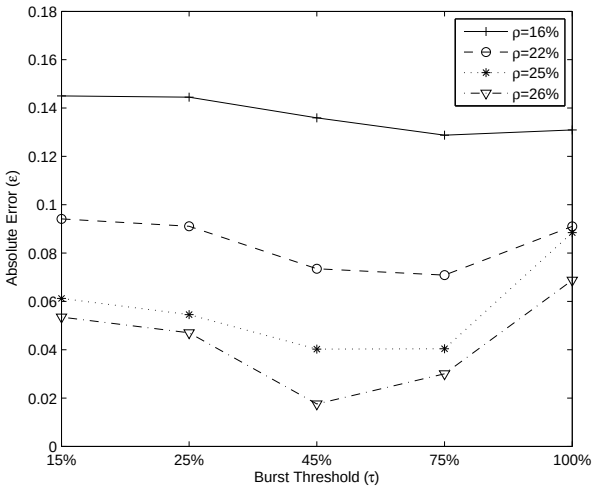


Figure 8.4: Density effect on PLR accuracy

in the Y-axis we show the absolute error, taking as reference the full trace. The figure shows that both for low and high density values the absolute error increases considerably. While the lowest values are accomplished for middle thresholds.

On the contrary, if we had an scenario without congestion and we wanted to detect shorter loss bursts or sporadic losses we should lower the density threshold. But, in the case of extreme congestion we should chose a higher value, just for guaranteeing that we do not over-react in case of sporadic losses. Hence τ is an input parameter which can be decided depending on specific needs.

With this approach we were able to reduce the absolute error in loss estimation, in this line of research, in the next section we further improve this estimation by using IPDV as a reference metric.

8.4 IPDV Based

After the throughout description of the PLR driven adaptive sampling approach, here we use a new metric in order to decide the resource sharing. In this case we consider the IPDV, which as we will show outperforms PLR as a good sharing policy for the sampling resources and the accuracy.

Before detailing the methodology, it is important to define *IPDV* in our context. *IPDV* usually is defined as $OWD_i - OWD_{i-1}$, excluding lost packets (See RFC-3393 [34]), or as $OWD_{99.9^{th}prc} - OWD_{min}$ in Rec. ITU-T 1541 [65]. In our work the first definition is not applicable since the packet sequence is lost during the sampling. Regarding the second option, obtaining the 99.9^{th} percentile for low rate flows is not possible because there are too few packets in the bin. As an alternative, RFC-3393 states that it is possible to specify a custom selection function to compute *IPDV*. Hence we use $IPDV_i = |\widehat{OWD} - OWD_i|$, for all the i packets within the bin. Where the estimator $\widehat{OWD} = E[OWD]$ for all the sampled packets.

Using the *IPDV* as an accuracy estimator is based on the following rationale:

1. Constant *OWD* leads to null *IPDV*, where both $\widehat{OWD}$ and $\widehat{IPDV}$ can be perfectly estimated using $\rho_{i_{min}}$.

2. High *IPDV* with low ρ_i might lead to inaccurate $\widehat{OWD}$ since the subset of packets selected by ρ_i may not be representative enough. Hence it is worth to increase ρ_i for high variable flows.

3. But, in some cases, with ρ_i we can incorrectly estimate small $\widehat{IPDV}$ due to the bias of the selected subset. This false negative will be corrected in later bins as $\widehat{IPDV}$ converges.

4. The other case is incorrectly estimating high $\widehat{IPDV}$ when in reality it is low. That being a false positive and forcing a needless increase of ρ_i for the flow. But not leading to incorrect estimation.

The probability of false positives or false negatives decreases as ρ_i increases. Then the system automatically tunes ρ_i reducing these mistakes in the estimation. Therefore, the different weights (ω_i) are distributed using:

$$\omega_i = \frac{\widehat{IPDV}_i}{\sum_{j=1}^{m} \widehat{IPDV}_j} \tag{8.9}$$

Fairly scheduling the ΔX resources proportionally to its delay variation.

8.4.1 Validation Methodology

This adaptive sampling methodology based on IPDV has been validated by using again the EuQoS testbed (Section 6.2.1). As before, we configured the testbed to act as a single domain, with one ME deployed on each testbed, amounting to 12 ME and 1 PE on this scenario. A

reduced portion of the testbed is depicted on Figure 8.5, where it can be seen the different paths for control and the normal traffic.

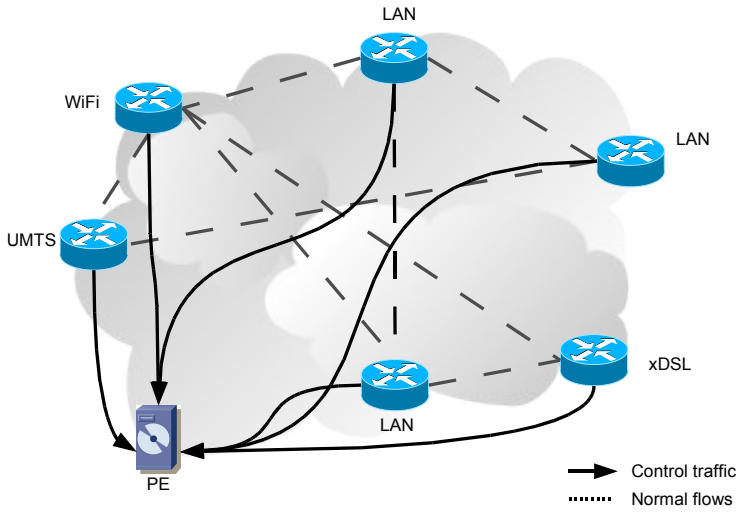


Figure 8.5: Portion of EuQoS Testbed with control traffic paths

The goal of the evaluation is twofold, in the one hand we compute the accuracy in the estimation of OWD and PLR. On the other hand, we analyse the used resources for the task.

For each test the full trace was collected. With the traces we simulated a network with the flows entering and leaving the system following an uniform randomly distribution up to a maximum of 100 simultaneous flows.

This way we were able to evaluate the results for different resource reservations (X) by taking as reference the complete original trace. In the experiments the used X were: $7000, 5000$, 3000, 2000, 1000, 500, 300 and 200 with the constant $\mathcal{S} = 1$ as detailed before. We also consider a bin size of $t = 175ms$ which provides a good tradeoff between the interval for live reporting and collection time to apply the proper sampling rate as we already proved in section 6.2.2.

Summarising the methodology used for obtaining a trace per different resources is:

1. Generate the test traffic and monitor all the traces in the MEs.
2. Simulate the scenario with multiple simultaneous flows using the real traces.
3. Split the results into bins of size t.

4. Apply off-line the adaptive sampling for the different resources (X).

5. For each bin and X the $\widehat{PLR}$ and the average $\widehat{OWD}$ are computed.

6. Finally the estimated OWD for the sampled bin is compared with the real value.

8.4.2 Evaluation Results

In order to evaluate the system we compare the obtained accuracy using adaptive sampling with the application of a fairly equivalent static sampling. Since the sampling rate (ρ) depends directly of c and R_i not all values of ρ within a bin lead to a feasible situation. For example, for low rate flows, (e.g. 16 packets per second), using the bin size $t = 175ms$ leads to $\simeq 3$ packets per bin, then $\rho_{i_{min}} = \frac{2}{3}$ since with IPDV we need at least 2 packets to compute it. But in the case of flows with 200 packets per bin then $\rho_{i_{min}} = \frac{2}{200}$.

8.4.2.1 Required resources:

The row labelled as (ρ_{eff}) of Table 8.4 shows the amount of resources effectively needed for each value of X. It details the global sampling rate applied taking into account all the flows and their durations. This value is obtained by aggregating all the considered samples out of the total amount of packets sent by all the tests. As it can be noted for low X values the sampling rate is very small, which means that only 3.8% (in the case of $X = 200$) of the resources needed to monitor $X_{\mathcal{D}}$ are required by our proposal.

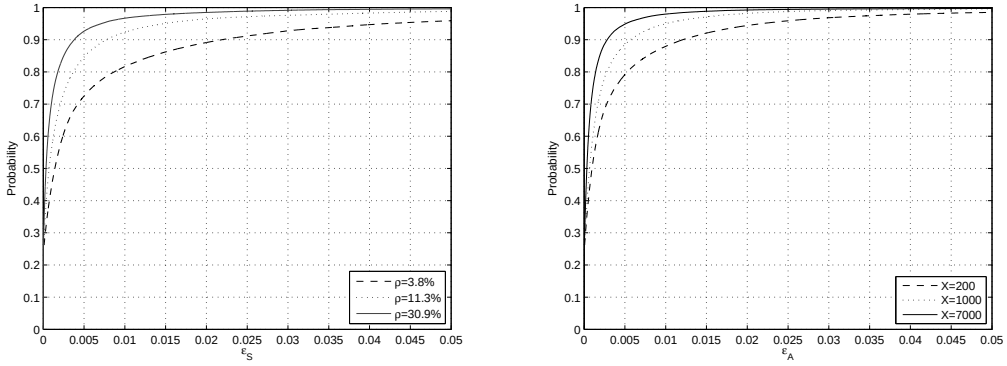
X	200	300	500	700	1000	2000	3000	5000	7000
ρ_{eff}	3.8%	4.5%	7.3%	9.0%	11.3%	17.6%	23.4%	27.3%	30.9%
ε_{delay}	0.025	0.018	0.013	0.012	0.010	0.008	0.007	0.005	0.005
γ_{loss}	0.22	0.17	0.15	0.11	0.09	0.07	0.06	0.05	0.05

Table 8.4: Effective global sampling and Delay and Loss errors for 95^{th} percentile per X

8.4.2.2 OWD Estimation

In order to evaluate the accuracy of the OWD estimation we compared the results obtained per X with the real traces by using relative delay error values: $\varepsilon = \left|1 - \frac{\hat{d}}{D}\right|$. Where D is the average OWD per bin taken from the complete trace and $\hat{d}$ stands for the estimate obtained by

our solution for different X. Figure 8.6 shows the comparison between static sampling (left) and adaptive sampling (right). The figures show the Cumulative Distribution Function (CDF) for various X for the adaptive and equivalent sampling rates for the static case. The X-axis of the figures holds the relative error (ε) while Y-axis is the error probability.



(a) Static Sampling

(b) Adaptive sampling

Figure 8.6: CDF of OWD accuracy comparison between adaptive and static sampling

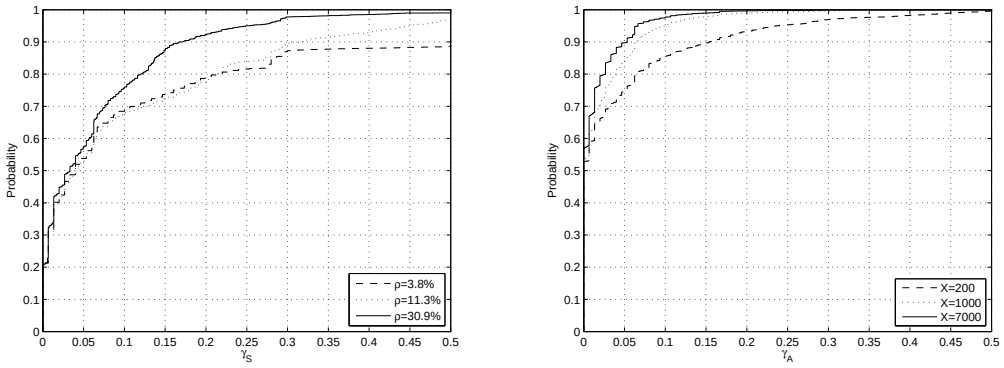
The results clearly show the gain obtained by our adaptive solution. Where adaptive sampling outperforms static sampling by around 50% for low values of X. For example in 95% of the tests, for $X = 200$ the $\varepsilon_A \leq 0.020$ while $\varepsilon_S \leq 0.040$. When the resources are increased the difference is reduced down to 5% and below for $X \geq 1000$. Moreover, adaptive sampling gives a more robust and reliable mechanism to control and schedule the used resources. This sensible gain in accuracy is further enhanced when dealing with packet losses as shown later.

From the accuracy point of view the figure shows that increasing the resources for the reporting improves greatly $\widehat{OWD}$, in fact 95% of the tests had an error smaller than 0.025 as summarised in the row labelled as ε_{delay} in Table 8.4.

8.4.2.3 PLR Estimation

Given that many flows did not have any packet loss and the system always estimates this case correctly, we decided to remove from the study such flows for not biasing the results. In this case PLR comparison is performed by using absolute errors. The results show that the accuracy is lower than for OWD as shown in Figure 8.7, but much higher than in the case of static sampling, whose error is up to $\sim 25\%$ in the best considered case $(30.9\%$ sampling$)$ for

95% of the tests. Compared to the worst obtained with adaptive sampling which is $\sim 20\%$ for the minimum resources.



(a) Static Sampling

(b) Adaptive sampling

Figure 8.7: CDF of PLR accuracy comparison between adaptive and static sampling

Eventhough for reasonable sampling rates the results show an error lower than 10% for 95% of the cases, the full details are in the row labelled as γ_{loss} in Table 8.4. A way to overcome this lower accuracy in the estimation of PLR, is to increase the bin size (t) in order to gather more information per bin. Hence improving the sampling estimation of the number of lost packets.

8.5 Bandwidth and Memory Cost Analysis

As we discussed before, our distributed adaptive sampling technique clearly outperforms static sampling. Although there is one open issue to be discussed, it is analysing of the cost of a real deployment of the architecture. All over the chapter we considered $\mathcal{S} = 1$ which in a real scenario it is not true.

The main goal of our proposal is that the final resource utilisation is bounded by X while when using static sampling, once we decide the sampling rate, the used resources will grow or decrease dynamically following this equation: $X_S = \rho_S \, \mathcal{S} \sum_{k=1}^{m} R_k$ where m is the total amount of flows within the domain, R_k as defined before is the rate of flow k. Which depends linearly of the sampling rate.

For the testbed used in this chapter, the total traffic generated in average is $\sim 200Mbps$ counting all the flows among all the 12 testbeds. Thus, considering the $\mathcal{S}$ values obtained in [107], the equivalent resources for the adaptive sampling solution are:

$X = 5000 \sim 470Kbps, X = 10000 \sim 600Kbps, X = 15000 \sim 700Kbps$ and $X = 20000 \sim 800Kbps$ in average.

While not using traffic sampling the amount of required resources is $2.6Mbps$.

In terms of memory, this solution uses the same amount of memory than static sampling, the only added overhead comes from the necessity of managing the resources among the different ME.

8.6 Conclusions

This chapter described the last piece of contribution related to the metric based distributed SLA Assessment. With the approach presented here, we developed a scalable, robust and controllable environment in order to assess the QoS of any network.

We proposed a fair resource charing methodology, which works by using estimated IPDVs and PLRs to focus the resources where are more needed. Moreover, with the used sampling solution it is possible to bound the used resources, therefore, the bandwidth issue found in the previous approaches has been solved. Evenmore, with the smart scheduling of the resources we can guarantee good accuracy on the reporting.

The main contributions related to this chapter can be found at [110] and [109], which are two of the most relevant publications of this work.

9

Quality of experience reporting

Previous chapters discussed about the SLA assessment at network level, this is the classical approach, using the network metrics, to map them into actual SLA contracts. But as more real-time applications appear, user level quality assessment mechanisms, namely Quality of Experience (QoE), became a must. In order to complete our pure network layer approach, in this chapter we, once again, enhance the IDRP to fit this new set of functionalities. Our focus on this regard relies on the pure VoIP user level objective assessment via Mean Opinion Score, briefly discussed in Section 3.1.3.

9.1 User Level Metric (MOS)

The main contribution of this chapter is to define a new metric and integrate it in our IDRP, which is based on existing ITU's definition of the *Mean Opinion Score* (MOS). This section introduces the current MOS specification, along with the underlying information which will help the development of such metric. It also discusses some important concepts that will limit or conditionate the results. The actual metric will be explained in the next section.

MOS is a value ranging between 1 and 4.5. It defines the overall subjective quality of any voice communication, 4.5 being the maximum and 1 the worst achievable degree of quality. MOS value can be obtained through the E-Model [59] which gives a deterministic computation of a subjective value. Its formula can be found in Equation 9.1.

$$\begin{aligned} MOS_{CQE} = &\ 1, && R = 0 \\ MOS_{CQE} = &\ 1 + 0.035R + R(R-60)\cdot \\ &\ \cdot(100-R)\cdot 7\cdot 10^{-6}, && 0 < R < 100 \\ MOS_{CQE} = &\ 4.5, && otherwise \end{aligned} \tag{9.1}$$

Where R is known as the transmission rating factor, computed by the equation 9.2. In this equation R_o stands for the signal-to-noise ratio, I_s is the simultaneous impairment factor. I_d refers to delay impairment factor, I_{e_eff} is the effective equipment impairment factor, and finally A holds the Advantage factor.

$$R = R_o - I_s - I_d - I_{e\_eff} + A \tag{9.2}$$

R ranges from 0 to 100 and its quality degrees are enumerated in table 9.1.

R (lower limit)	MOS (lower limit)	User Satisfaction
90	4.34	Everybody satisfied
80	4.03	Satisfied
70	3.60	Some users not satisfied
60	3.10	Many users dissatisfied
50	2.58	Nearly all users not satisfied
0	1	Impossible to understand the conversation

Table 9.1: possible R and MOS ranges

For further details on this topic, the reader is referred to ITU Recommendations [60], where the proposed implementation discusses several different environments and its effects on the final quality. For the ease of simplicity, we take the default values [28] on all the parameters, except the ones derived from the network behaviour, such as one-way delay and packet losses.

The most relevant parameters for this study are I_d and I_{e_eff}. I_d represents the mouth to ear delay (one way delay in our scenario) and I_{e_eff} that is codec dependent and is computed from the packet losses. They bound the quality limits of the communication in the network.

9.2 Extended MOS

The MOS metric was initially designed to describe the overall quality of a call on a subjective scale, based on the assumption that the call is routed through a circuit switched network. Today, as VoIP calls are routed over the Internet, we believe that it is not sufficient to describe the call

quality with a single value. The duration of a call can last from a few minutes to as much as few hours. During this period the underlying network properties may change significantly. This means that while during certain periods of the call we will experience good call quality, there may be intervals with poor quality or even complete lack of voice reception. This raises the legitimate question of how to decide the quality of these calls. While using the ITU-T recommendations formulae we get a quality estimate, this merely averages call parameters ignoring important information. In order to work around these shortcomings, we propose the definition of *Extended MOS (E-MOS)*. This definition is based both on the original MOS and on the IPPM's `Type-P` packet described previously.

The main improvement of Extended MOS is the division of the voice stream in smaller segments and to perform call's quality computation on these chunks. Each one of these chunks can represent from a predetermined number of packets to a variable size talkspurts (Nomenclature obtained from [18] referring to continuous talk from one person).

This proposal fits in the global NPAS infrastructure as a new layer to the system. Figure 9.1 shows the new blocks, it can be seen that the extensions to our IDRP are in both ME and PE. In the ME side we need some mechanisms in order to detect any VoIP flows and the codec used for the communication. Techniques for codec detection are out of the scope of this work, and during the rest of the chapter we assume knowledge about the codecs.

In the side of the PE we use this codec information, together with the metric statistics in order to give the higher layer QoE Assessement.

We define this new metric following the IPPM policy of metric definition, therefore we use the `Type-P` packets introduced in section 3.1.1.

9.2.1 Type-P-MOS

Let's define a singleton metric called `Type-P-MOS` with the following parameters:

- *Source*: Source IP address of a host.
- *Destination*: Destination IP address of a host.
- T_0: an initial time.
- T_f: a finish time.
- *C*: a voice codec from the list found in [59].

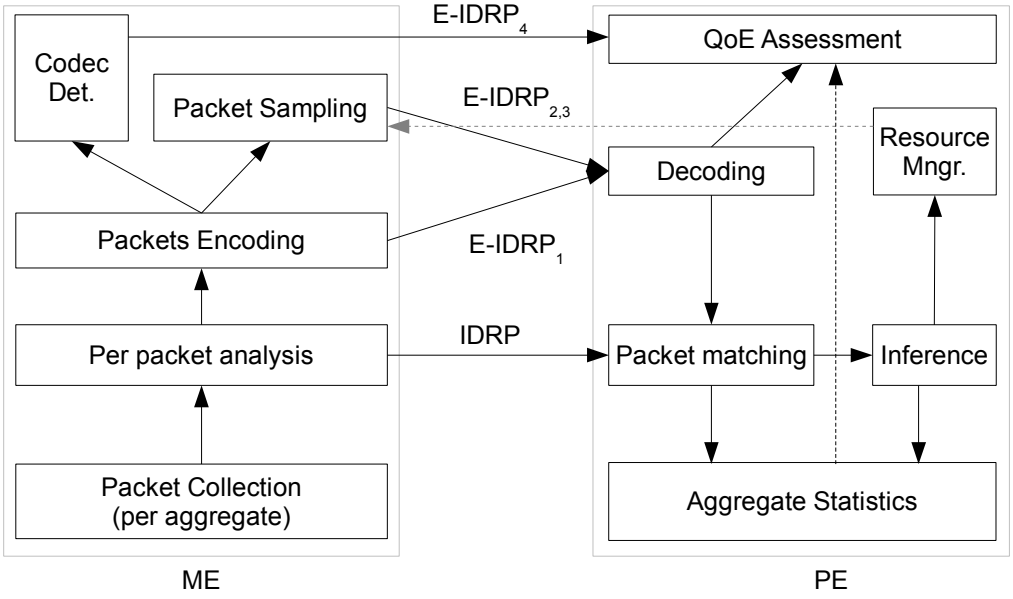


Figure 9.1: Block view of $(E)IDRP_4$

- F: a selection function defining unambiguously the packets taken from the stream selected for the metric. It takes two parameters, P, described below, and a packet. It will output the packet or `null` if the packet is not selectable.
- p: the specification of the packet type. This will define a F_P which holds the list of packets chosen for the metric.

Both T_0 and T_f form a time interval that determines the period of packet selection decided by `F`.

This metric is composed by OWD [5], IPDV [34], PLR [6] and the voice codec [59] used for the communication.

9.2.1.1 Algorithm

`Type-P-MOS` reports the MOS value obtained from the E-Model from the selected packets by F_p. The pseudo code for this metric follows in Algorithm 2.

The Input list of packets must have the sending and receiving timestamps of the packet, or in the case of packet loss the sending timestamp and a mark indicating packet loss.

Algorithm 2 `Type-P-MOS` pseudo code

Input: Packet[1..n], C {Packet's input stream, it might be unbounded. C is the used codec for transmission}
i = 1
t = $\mathbf{T_0}$
S = {} {Initialise S, it will hold the list of selected packets}
repeat
  $k \leftarrow F_p(Packet[i])$ {Selects the first packet of the stream provided it is a selectable `Type-P` Packet}
  if k **then**
    $S \leftarrow S \cup Packet[i]$
  end if
  i++
  $t \leftarrow getSendTimeStamp(Packet[i])$
until $t \geq \mathbf{T_f} \vee i \geq n$
Output: MOS(S, C)

9.2.2 Type-P-EMOS-*-Stream

`Type-P-EMOS-*-Stream` uses `Type-P-MOS` as a base for a new metric. Its parameters are:

- *Source*: source IP of a host.
- *Destination*: destination IP of a host.
- T_0: a initial time.
- T_f: a finish time.
- F: a selection function.
- p: the specification of the packet type to select. This will define a F_p which holds the list of packets chosen for the metric.

Depending on the selection function, the metric might have different behaviour. Hence, the `*` in `Type-P-EMOS-*-Stream`.

9.2.2.1 Selection function

`F` in this case will decide the intervals depending on p (F_p), between T_0 and T_f which are the proper thresholds for computing the `Type-P-MOS` value. Such function specifies the capture boundaries. A detailed description of selection functions is left as an important part of our future work. Initial possibilities:

1. `Type-P-EMOS-Periodic-Stream`: Regular non-overlapping time intervals, this computes MOS values periodically over time, regardless the contents of the voice transmission.

2. `Type-P-EMOS-Sliding-Stream`: Similar to `Periodic` but the time intervals overlap over time. This permits to keep a history of past events to avoid reporting independent MOS values.

3. `Type-P-EMOS-Talkspurt-Stream`: For this to work prior knowledge of the codec, silence detection algorithms and methods for payload examination of the traffic are needed.

9.2.2.2 Algorithm

This metric applies the `Type-P-MOS` metric to the packets contained in the limits expressed by T_0 and T_f. Pseudocode for this operations is shown in Algorithm 3. The output is an array of n MOS values.

The algorithm is straight-forward, it selects the lower and higher boundaries of the packet stream, it computes MOS over that fragment. The system monitors whether the MOS is within valid boundaries, triggering the required action if needed.

For off-line processing when all the packets in the input stream have been processed, the `mosArray` and the `timeArray` are returned.

9.2.2.3 Metric results

As shown in the metric definition the output is an array of values, this array gives the voice quality over time. With this information it is possible to have accurate reporting of the status of the voice quality. This can be used by service providers to give feedback to the users about the delivered voice quality.

Some statistics definitions for `Type-P-EMOS-*-Stream`:

Algorithm 3 `Type-P-EMOS-*-Stream` pseudo code

Input: Packet[1..n], C {Packet's input stream, it might be unbounded. C is the codec}
t_{th} = {0,0} {Contains the packet's interval it can hold timestamps or packet counts}
i = 0
while t_{th} = F(Packets,t_{th}) **do**
{Fills up t_{th} with the time interval decided by `F`}
$mosArray[i] \leftarrow$
$\texttt{Type} - \texttt{P} - \texttt{MOS}(Packet[t_{th_0}, t_{th_1}], C)$
if ActionNeeded **then**
TriggerEvent($mosArray[i]$)
end if
$timeArray[i] \leftarrow t_{th}$
i++
end while
Output: mosArray, timeArray

- `Type-P-EMOS-*-Mean`: Refers to the mean value of the *mosArray* output. This value is the closer to the original MOS as will be shown later.
- `Type-P-EMOS-*-Std`: Is the standard deviation of the *mosArray* output.
- `Type-P-EMOS-*-Percentile`: Given a percentile (`P`) value between 0% and 100% the value which has P% values below. This can be useful for outliers detection.
- `Type-P-EMOS-*-Median`: This metric is equivalent to the 50th percentile except when even number of values are returned, in that case the mean value between them is taken.
- `Type-P-EMOS-*-Minimum`: The minimum of all the `Type-P-MOS` values. This can be extended to the 0.01^{th} percentile.
- `Type-P-EMOS-*-Maximum`: The maximum of all the `Type-P-MOS` values. In this case the extension can be to the 99.9^{th} percentile.

9.3 Validation

Once the metric has been presented this section is focused on the validation of the system, we also present the tests and the results to verify the behaviour of the proposed metric in a real

scenario.

For the validation we only consider `Type-P-EMOS-Periodic-Stream`, extending the validation to other metrics is straight-forward.

E-MOS supersedes original MOS. Since `Type-P-EMOS-*-Mean` can deliver the same call quality as MOS with a bounded error ($\pm\varepsilon$). Where ε is `Type-P-EMOS-*-Std` because MOS algorithm uses *Mean Delays* and *Packet Loss Probability* of the whole conversation. For more detailed information on MOS computation refer to [59].

9.3.1 Methodology

As discussed previously in the literature [95] and in this thesis, testing is not a simple task. This section details the methodology we have used for guaranteeing the soundness of the results presented later.

9.3.1.1 Capturing environment

The set of tests prepared for this work use real applications (Linphone in this case), usually such end user tools are not suited for delivering detailed statistics about network information (i.e. per packet one way delays, packet losses), with this situation there is the need of complementary tools to perform such tasks.

A first approach could be the use of other tools to actively generate traffic which could resemble somewhat the actual voice traffic generated by the application. As seems obvious, in order to simulate the traffic flows is not a good approach given that the codecs used do not generate constant bit rate traffic, specially due to silence detection.

As a second option, there is the possibility to use available passive capturing tools (i.e. Ethereal, tcpdump...). The problem found with this approach is the need of computing one way delays and packet losses of the flows under test. This forces to set up two capture points, one in the source host and the second at reception. Moreover, this approach needs the development of an algorithm for flow detection and packet impairment at both ends from the trace files, which is inherently inefficient because the full packet payload must be captured and stored for later processing.

The problem with this second option is the (automatic) correlation between both traces. That's why we chose a third approach, which is using our NPAS implementation. It uses the

`libpcap` library for capturing the packets, the difference with other tools using this library resides in the fact that it permits to specify different capture points simultaneously.

9.3.1.2 Testbed

The testbed used for this work is composed by the two end-points and a Linux router. The goal is to keep it as simple as possible and to have control over the network behaviour using queueing mechanisms.

One of the main parameters needed for the metric are One Way Delays, to compute it it is mandatory to have the proper synchronisation on the equipment as the timestamps must be comparable. Therefore we used a mixed software and hardware approach via NTP and PPS. We enhanced further the precision of the clocks by using several stratum-1 reliable time sources, specifically we used two separated GPS servers in our lab.

Another source of noise is the variability among the tests. Voice encoding and generated traffic might vary depending on the silence periods or the actual voice of the speaker. For avoiding incorrect results caused by this variability we used a prerecorded conversation together with Linphone which lasted for 4 minutes 16 seconds. The conversation was a standard dialogue between two persons talking English.

We decided to use the previously recorded couple of files (one for each direction of the dialogue) for all the tests, this way potential changes on the conversation would not affect our tests. To guarantee proper interpretation of the results we fixed the transmission Codec (C) to G.711. To transmit the voice while avoiding echoes we installed two sound cards on each computer, one for transmitting the prerecorded WAV file, and connected with an external cable to the line-in input of the other card, which was the one actually feeding the data to the VoIP application.

All this set up permits to automate the generation of test. We were able to capture each test separately and to repeat them as many times as we needed.

9.3.1.3 Network characteristics

As the testbed is set up on a local network there is no congestion is encountered. We introduced controlled sources of variation using `netem` to emulate different kind of network behaviour.

`netem` is a traffic control (`tc` [75]) mechanism available on current Linux Kernel (we used 2.6.15 in our tests), which permits to set up different network conditions in an easy way. A

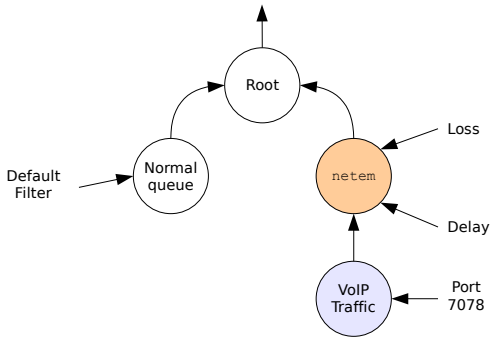


Figure 9.2: Queue hierarchy

deep description of this software is out of the scope of this chapter, here we will only describe the basic functionalities we used for emulating large one way delays, packet losses and high jitter.

Figure 9.2 shows the queueing hierarchy used for the tests. The key point in the proposed scheduling is the fact that the average Internet traffic doesn't get affected by the `netem` tweaked queueing. For doing so, we forced Linphone to use port 7078 and we filter all the UDP traffic outgoing from the machines in the specified port, this way the Linphone control channel is going through the standard Linux queueing mechanism.

Each performed test has different values for packet loss ratio, one way delay and jitter. Each change provides different conditions for computing E-MOS experimentally.

In one way delay we modeled (as `netem` permits) the delays around a normal distribution of the specified value, their details are discussed later.

9.3.2 Tests

We performed two different set of tests, first with constant network conditions and the second using variable parameters.

The first set of tests, along with its main characteristics are summarised in Table 9.2. Both delay and losses are added in a controlled way. The delays are modified by a pseudo random jitter of 3ms in delay to have a more realistic environment.

The goal of this set is to prove that standard MOS gives accurate results as long as the network metrics are stable during the tests. That is, the packet losses and the one way delays are equally distributed along the whole conversation.

Each tests was repeated several times for achieving statistical soundness. Table 9.2 summarises the Delay and Packet Losses obtained from testing. The table shows the computed mean for each type of test.

Set	Characteristics	Average	Std Dev
Test 1	Loss 50%	49.88%	0.8%
Test 2	Loss 25%	25.3%	0.3%
Test 3	Loss 10%	10.61%	0.1%
Test 4	Loss 5%	5%	$\sim 0\%$
Test 5	Loss 1%	1.4%	$\sim 0\%$
Test 6	Loss 0%	0%	0%
Test 7	Delay 500 (ms)	497.37	6.41
Test 8	Delay 300 (ms)	301.83	4.32
Test 9	Delay 100 (ms)	104.28	2.38
Test 10	Delay 50 (ms)	53.23	0.83
Test 11	Delay 0 (ms)	3.4	$1.9 \cdot 10^{-4}$

Table 9.2: Mean Delays and Packet Losses

We also performed tests with 0.1% loss ratio, but given the low packet rate of the voice flows the results are similar to the lossless case, thus are not shown in the table.

As it can be noted, some of the performed tests are not realistic, namely 50% losses or delays bigger than 200ms (although as shown in [69] some applications have bigger mouth to ear delays) but our goal is to highlight the improvement acquired by E-MOS over MOS.

In the second set of tests, the variation of delays was not constant, there was an increase in delays of 10ms each 10 seconds, starting with 1ms until 300ms of delay at the end of the test. Moreover, for having more variability, a jitter proportional at 10% of the delay value is forced. With this behaviour it is very easy to notice the inherent problems of the legacy MOS algorithm with only one value as result.

9.3.3 Results

Our main focus is to show the improvement we obtain by using E-MOS over standard MOS. For this purpose here we present the results obtained from the two different testset.

9.3.3.1 Homogeneous network conditions

As described in the Tests section, the set of performed measurements treat separately packet losses and delays. This way it is possible to isolate each metric effect over the final call quality all over the test.

Packet Losses Before studying packet loss effects on call quality two different aspects must be considered. First packet losses effects in the final MOS value is a work in progress as stated in ITU's reccommendation G.113 Appendix A [60]. Second the outcome of the results depend strongly on the codec. For this purpose we forced Linphone to use the G.711 Codec.

This tests with homogeneous network conditions have a twofold goal. On the one hand we validate the good results obtained standard MOS algorithm when network conditions don't change drastically over time. This highlights that MOS, as designed for circuit switched networks was a good approach, even if it should be adapted to the new network dynamics.

On the other hand we point out the improvement in reporting precision we obtain by using our E-MOS proposal.

Table 9.3 shows the mean values both of delay and loss for each testset with controlled packet losses. We used 1s, 3s and 5s boundaries for computing the parameters, the table shows the 1s case of `Type-P-EMOS-Periodic-Stream`.

In the table, E-MOS Mean and E-MOS Std. refer respectively to `Type-P-EMOS-Periodic-Mean` and `Type-P-EMOS-Periodic-Std` metrics defined in section 9.2.

As it can be noted E-MOS Mean is similar to the MOS as the network conditions are kept during all the test. The results show that for having a reasonable minimum quality, losses should be kept below 1.4% that corresponds to *Some users satisfied* entry on Table 9.1. With higher loss ratios MOS and E-MOS values are clearly below the threshold of admissible quality.

Figure 9.3 shows the evolution of E-MOS over the first minute for 1% losses test. The figure presents E-MOS computed for 1s, 3s, and 5s intervals respectively for `Type-P-EMOS-Periodic`. The MOS value of the whole conversation is also shown as baseline to which we can compare the other results.

	Test 1	*Test 2*	*Test 3*	*Test 4*	*Test 5*	*Test 6*
Loss Ratio	49%	25%	10%	5%	1.4%	0%
Delay(ms)	0.9	4.1	4.2	4.4	5.1	3.5
MOS	1.3	1.78	2.68	3.24	3.87	4.05
E-MOS Mean	1.36	1.94	2.85	3.34	3.89	4.05
E-MOS Std	0.12	0.41	0.56	0.53	0.33	0.14

Table 9.3: MOS with controlled Packet Losses (1s periodic)

With homogeneous network conditions, increasing the period of the metric tend to smooth the variability of the result. In lower timescales with the low packet rate of VoIP traffic the homogeneity is not preserved.

The figure illustrates the improvements brought by E-MOS. Where MOS reports a single static value, E-MOS delivers periodic feedback about the call quality. This can be used by operators to prove the service is being properly delivered, or it can even trigger the corresponding network control entities that can provision extra resources, or change the billing algorithm as decided on the customer's contract.

An interesting outcome of the analysis of E-MOS over the conversation is the amount of time an user feels good quality while having the conversation. Figure 9.4(a) illustrates that as the packet losses increase it's quality decreases as expected, up to the point of having almost all the conversation under minimum quality conditions for the 50% and 25% loss.

The case where everyone is satisfied doesn't have any occurrence as the G.711 codec has a maximum theoretical value of 4.11 which is below the 4.34 limit for this entry on Table 9.1. The histogram also highlights that when there are 10% or more losses then more that in 50% of the call time is very difficult or impossible to understand the conversation (e.g. MOS below 3.10).

Delay variations Delay analysis is performed in a similar way as the loss. Table 9.4 shows the obtained mean delay, MOS value and E-MOS mean. There is no Loss entry because given the good conditions of the network no losses occurred during the tests at network level. There-

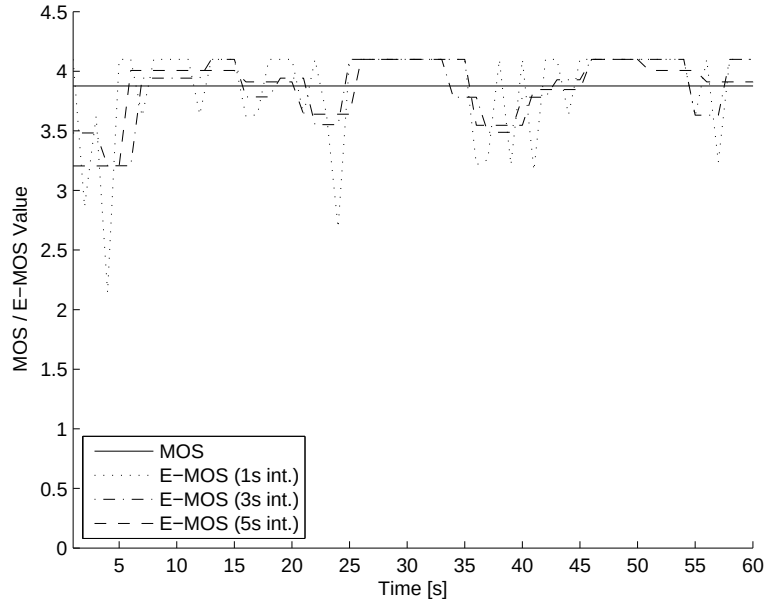


Figure 9.3: E-MOS evolution (1% packet losses)

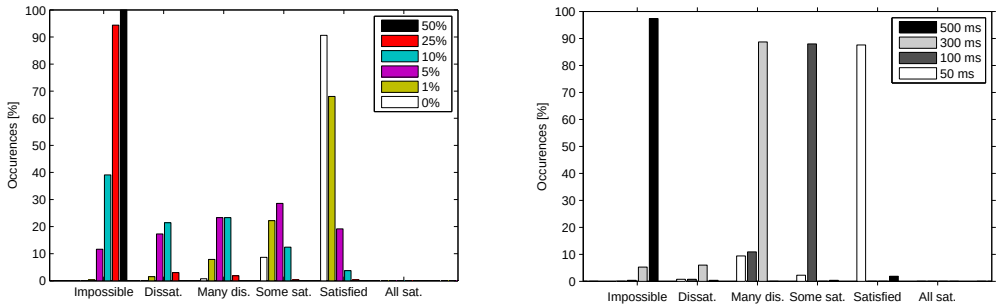
fore, the column related to 0ms delay is also omitted because the results are the same as in loss 0%.

	Test 7	*Test 8*	*Test 9*	*Test 10*
Delay	497.37	307.37	104.28	53.23
MOS	2.29	2.96	3.97	3.98
E. Mean	2.23	2.99	3.84	4.04
E. Std	0.32	0.36	0.33	0.01

Table 9.4: MOS with controlled One Way Delays (units in ms)

Related to packet losses, the testbed computes *Network* losses due to Network problems. During the tests there were some losses at application level, this is because of the real-time nature of the conversation.

In Figure 9.4(b) it can be seen the more deterministic effect of one way delays over the call quality. This happens because the network homogeneity for one way delay is preserved



(a) Packet losses

(b) One Way Delay

Figure 9.4: User satisfaction

regardless the considered timescale. While packet losses are discrete and have bigger impact at small time scales.

Another important implication is that one way delays bigger than 100ms stimulate a considerable conversation degradation, as it drops from *Satisfied* to *Some Satisfied* (see Table 9.1).

9.3.3.2 Variable network behavior

The second testset instead of keeping homogeneous network conditions is focused on studying the effects of increasing delay over MOS and E-MOS during the conversation. This time the difference between both metrics is much bigger as MOS does not react properly to high network variability.

Figure 9.5 shows the E-MOS value computed on 1s and 5s intervals, and the overall MOS value. There is a threshold at 250 ms delay where E-MOS reaches the lower bound and renders the conversation not understandable.

On the other hand, MOS equals to 2.6, which means that almost all users are not satisfied about the quality, while in reality it was perfectly good during 50% of the test duration. This difference is more noticeable with E-MOS results. It reports a mean of 1.97 with a standard deviation of 1.2, meaning that such values are not statistically significant.

This high variability is not common in the Internet, but highlights the point that with high jitter, or important changes in network conditions, MOS is not a proper metric for voice quality measurement. While enhancing it with E-MOS permits to differentiate clearly which parts of the conversation are good or not.

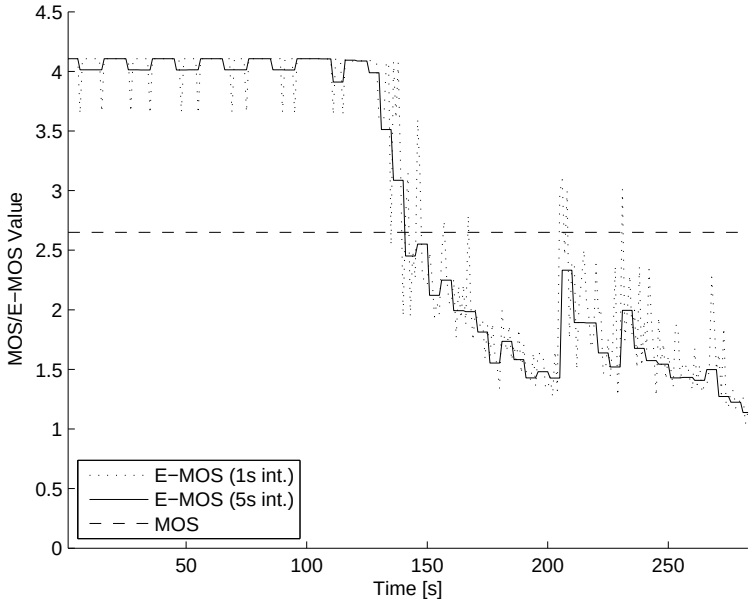


Figure 9.5: E-MOS Progressive Test

9.4 Conclusions

This chapter had three main purposes, first to study existing quality reporting tools and develop a new metric derived of currently existing methods. Second to study, under several network conditions the effects and differences between the developed metric and the original MOS approach. Such differences highlight the need and usefulness of our proposal for proper quality reporting in short timescales. And Third to incorporate in our NPAS some Quality of Experience capabilities.

This metric has been developed using definitions and methodology of ITU-T and IPPM. From ITU-T the MOS definition and implementation has been adapted to suit the new VoIP paradigm in the Internet. Regarding IPPM the low level network metrics used for measurement network performance (one way delay and losses) have been used together with IPPM's methodology for defining new metrics with the definition of `Type-P-MOS` and `Type-PEMOS-*-Stream` metrics.

The problems with this single value metrics arises when the network conditions change over time, which in currently available networks is more than likely to happen. The solution

for proper voice quality reporting is to use the proposed E-MOS metric, this enables potential operators or service providers to keep a more detailed track of the delivered quality over time.

Our most relevant contribution related to this chapter is [115].

10

Inter Packet Arrival Time detection

Up to now, in our work we have centred the SLA assessment in the network metrics estimation, together with some techniques in order to develop a scalable Network Parameter Acquisition System. As the last improvement presented in this work, we propose a novel mechanism and methodology that avoids as much as possible the computation of the performance metrics, while using other traffic information such as IPAT.

10.1 New paradigm

Classical approaches to SLA assessment, as we discussed during this thesis, are based on accurate network metric estimation to infer the actual SLA compliance. As we have seen, the most used metrics are:

- *One-Way Delay:* used to infer the degree of interaction in a data transmission.
- *Inter Packet Delay Variation:* which usually is used to decide application buffer sizes, and often also to determine the degree of interactivity of the network.
- *Packet Loss Ratio:* which in some context is the most difficult metric to estimate [110]. But nonetheless, one of the most important since it directly determines the quality of the communication.

As we already know, all these metrics determine the network quality. But from a practical point of view, they are only a mean for SLA assessment. Therefore its accurate estimation is not relevant, because our final goal is to detect network anomalies (in terms of SLA) and report them.

The solution to SLA assessment presented in this chapter differs from the previous alternatives in the sense that we do not use the metrics as the basis for the assessment, the main reasons are:

- Estimating the metrics implies to gather distributed information about the traffic and the synchronisation among the involved entities. Such control traffic is an important bottleneck of any solution using this approach as we have described previously in this thesis.
- Computing the metrics requires multiple capture points. And in the case of using active probing a traffic generation station located in some advantage point.

Therefore, such systems suffer from large scalability issues.

In this context, we do not intend to improve the accuracy on the QoS metric estimation. The originality of the contribution presented in this chapter relies on the statement that the accurate estimation of network QoS parameters is absolutely not required in most cases: specifically it is sufficient to be aware of service disruptions, i.e. when the QoS provided by the network collapses. In our original approach we just focus on the actual scalable detection and reporting of any potential violation of the SLA in the network. For this purpose, we propose a new approach which:

1. Works as much as possible with a single point of analysis.
2. Computes data very efficiently in order to have a scalable system.
3. Relies on the use of existing correlation between measured parameters and network quality.

We then propose to use Inter Packet Arrival Time (IPAT) because it complies with the above restrictions: IPATs can be easily computed at destination by getting the reception timestamps of the packets; IPAT computation only involves a subtraction of two integers (timestamps). Finally, and this is what we want to prove in the rest of the chapter, it exists a strong correlation between IPAT distribution and network performance: it was demonstrated by previous work that IPATs are tightly correlated with network congestion [98; 126]. We extend this correlation by mapping these IPATs with the actual network conditions, and by using some information about the real metrics. In particular, our proposal relies on statistical analysis of the IPATs, with the goal of detecting changes on the network status. This is done by comparing different IPAT

distributions using well-known algorithms, such as *Kullback-Leibler Divergence* and *Hausdorff Distance*. When our comparison is successful we assume that both distributions represent similar network conditions. In the case that both are different, we ask the ME about the real traffic information to compute metrics as we have been doing during this thesis.

10.1.1 Final extensions

To fit this new methodology to our system we must upgrade, once again, the functionalities of our IDRP. Here we present the final update, which brings the most complete version of the system, as we show in Figure 10.1.

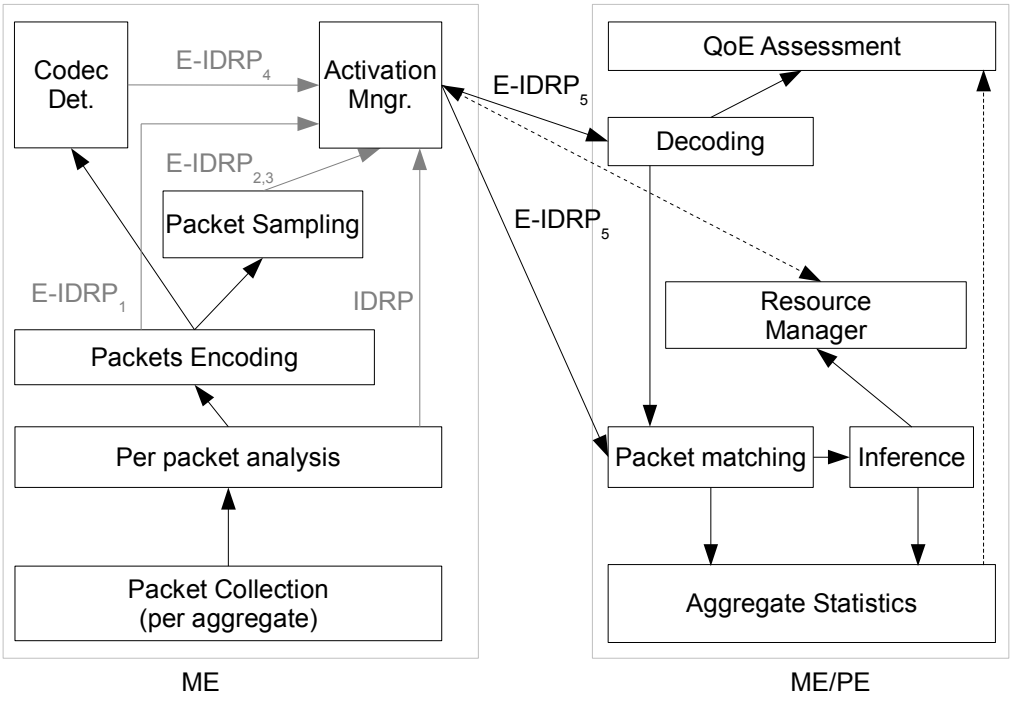


Figure 10.1: Final version of the building blocks of $E-IDRP_5$

Since the IPATs are gathered on the egress node, in order to support the required new set of functionalities, in our IDRP, we must distribute our PE, and provide a fully distributed SLA violation detection system.

As a consequence, the most important change in the building blocks of the system, is the

introduction of a new role on the system, the ME/PE, which performs the combined tasks of ME and PE together.

Another noticeable change is the apparition of the Activation Manager, which is the central block in the ME where all the subsystems send the information. Now the only interface between both entities is $E-IDRP_5$, we detail this whole new methodology during the rest of this chapter. This Activation Manager receives the instructions from the Resource Manager, which is the entity in charge of deciding when will the system deliver packet information.

10.1.2 Background: IPAT and anomalies

This section aims at showing empirically, with a very basic study, the relation between changes on IPAT and OWDs . As we already discussed, in the past some work has been done in order to correlate OWDs and IPAT [98; 126]. Hence, extensive analysis on the subject is not required. Nevertheless, we bring this correlation one step further, since it is clear that abrupt changes on IPAT, generally lead to changes on OWD (hence IPDV) and in Packet Losses. Figure 10.2 shows a very simple example where a Poisson like flow of 1Mbps is sent on an uncongested fast ethernet network, at some point we increased the OWD in the network while limitting the router buffers leading to uncontrolled packet losses. In the upper part of the figure we show the evolution of OWD together with the IPAT on each time interval, the X-Axis shows the packet sequence numbers, while Y-Axis show in two different scales the OWD and the IPAT, all of them in milliseconds. In the lower part of the figure the PLR and the same IPAT are shown. As it can be noted the IPAT change similarly to the changes in the OWD.

The rest of the chapter will present and evaluate a methodology that permits to detect this changes automatically, correlate them with the real metrics and exploit this correlation for SLA violation detection.

10.2 Base distance algorithms

As described before, we plan to use IPAT in order to infer violations in the SLA. Even with their good characteristics in terms of computational efficiency, just gathering IPATs is not enough to provide SLA assessment. First, IPATs do not contain information about the useful metrics of the network. Second, IPAT might change unexpectedly, sometimes due to real changes on the network conditions, but also due to the change in the traffic profile (e.g. change in the codec,

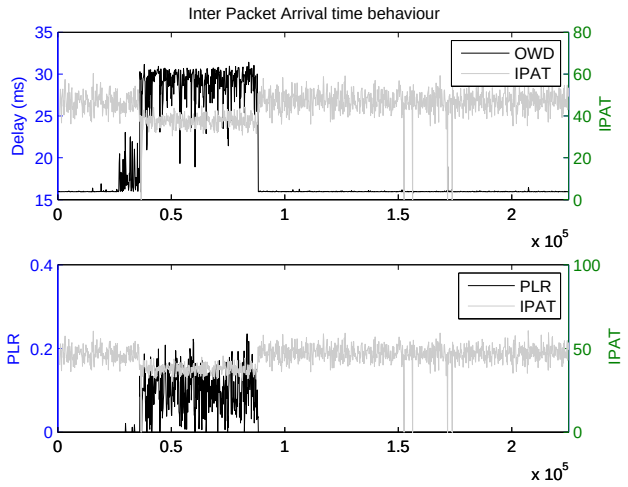


Figure 10.2: Effects of network anomalies to IPAT

silence on the conversation, etc.). Third, there is no direct mapping between IPAT and SLA violations.

Consequently, the most original contribution of this chapter is the detection of the SLA disruptions with minimal computation of the network performance metrics. We achieve this by periodically comparing the current IPATs distribution with a reference distributions set. Of course, getting the reference distributions set means integrating an on-line training process which records all new IPAT distributions observed on the network. It also makes, at the same time, the link between each of these new IPAT distributions and the current QoS parameters (by measuring them). Then, each IPATs distribution will be associated to a particular QoS level of the network. Then, in this section we focus on the generic description of the distance algorithms used, while in the next we will detail the full infrastructure using this information.

We use two different well-known distance computation algorithms. The first one, based and applied to distributions is called *Kullback-Leibler Divergence* [73]. And the other, originally intended for geometric distance calculations, namely the *Hausdorff Distance* [14].

Here we present the analysis of each distance computation algorithm. On the way, we propose an enhancement of the *Hausdorff Distance* in order to reduce its computational complexity in our scenario; we name this new approach *Simplified Hausdorff Distance*.

10.2.1 Kullback-Leibler Divergence

Entropy, in the area of information theory, is a measure of the uncertainty associated with a random variable. Sometimes the actual entropy value is not directly indicative of any interesting property. In this context, it is more useful to consider the *relative entropy* or *Kullback-Leibler Divergence*, which indicates the difference (i.e., how far) a distribution is from another. This relative value can give many insights about interesting properties on our dataset. *Kullback-Leibler Divergence* is defined as:

Definition 6. *Given two distributions $P(x)$ and $Q(x)$ from discrete random variables, the Kullback-Leibler Divergence [73] is defined as:*

$$K(P,Q) = \sum_i P(x) \cdot log \frac{P(x)}{Q(x)} \tag{10.1}$$

The above expression gives the degree of similitude between both distributions, taking P as the good (i.e. reference) distribution against Q, the one to be tested. The outcome is the divergence between both distributions.

It can be noted from expression (10.1) *Kullback-Leibler Divergence* is not symmetrical, as $K(P,Q) \neq K(Q,P)$, therefore, we have to select carefully the operands in the comparison.

In this work we use this divergence as a measure of the difference on the IPAT distribution, which indicates potential changes in the network conditions. Even if *Kullback-Leibler* cannot be considered a distance, to ease the comprehension, during this part of the contribution we use the terms *Distance* and *Divergence* seamlessly.

The distance computation must be very efficient, since it is performed massively in our algorithm. *Kullback-Leibler Divergence* has linear cost, which makes the algorithm a good alternative. It computes a division's logarithm for all the elements of each distribution, since both distributions must have the same size the cost is $O(n) = n$ where n is the distribution size (its number of bins).

10.2.2 Hausdorff Distance

Another classical algorithm used in the literature to compute distances is known as *Hausdorff Distance* [11; 67]. This algorithm, mostly used in image recognition and object location, is known for its good versatility in measuring the distance between two different geometrical objects.

Definition 7. *Hausdorff Distance is the maximum of all the minimum distances between two polygons.*

Formally, Hausdorff Distance gets a finite set of points $P = \{p_1, \ldots, p_n\}$ *representing a reference and compares it to a probe* $Q = \{q_1, \ldots, q_n\}$ *using:*

$$h(P,Q) = \max_{p \in P}\{\min_{q \in Q}\{g(p,q)\}\} \tag{10.2}$$

Where $g(p,q)$ stands for the geometrical distance between p and q. Analogously to *Kullback-Leibler*, the distance computation is not symmetric.

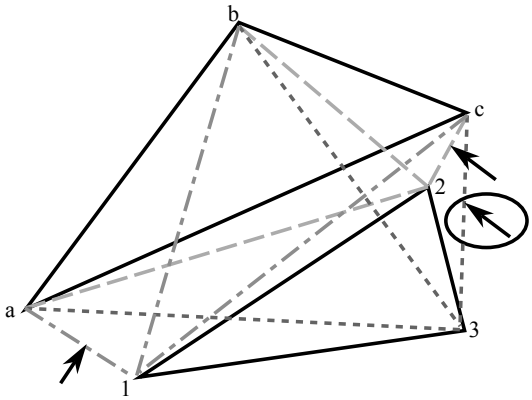


Figure 10.3: Hausdorff Distance example

As it can be observed, originally this algorithm is designed to work over geometrical objects instead of distributions, we illustrate in one example this method in Figure 10.3. Where we show in dashed lines all the computed distances, the arrows point to the minimum per vertex distance (solution for the first round), and the circle shows the final solution taken by the algorithm.

The efficiency, taken as computational cost is worse than. *Kullback-Leibler Divergence* Here we have to get the minimum distance from one element of P towards *all* the elements of Q, hence the cost is $O(n) = nm$ where n is the size of P and m is Q's. In general we can assume that $n \approx m$, thus $O(n) = n^2$.

This complexity for an on-line algorithm as the one we are proposing in this work represents a bottleneck on the system, but in general, as we will see in the evaluation section, *Hausdorff Distance* uses less network resources than *Kullback-Leibler*. Therefore, it is an interesting alternative in our environment.

10.2.3 The Simplified Hausdorff Distance

Hausdorff Distance was designed in order to operate in the geometric plane, but we use distributions instead of polygons, which allows us to reduce the algorithm's complexity.

Hausdorff Distance basically compares all the elements of a set with all the elements of the other. Although, when working with distributions, there is a new dimension that does not exist on the geometrical plane, namely the bin size. Therefore, to reduce the computational cost, we only need to compare similar bins (in position and value) opposed to the *"all-against-all"* policy of the original algorithm, these differences are illustrated in Figure 10.4.

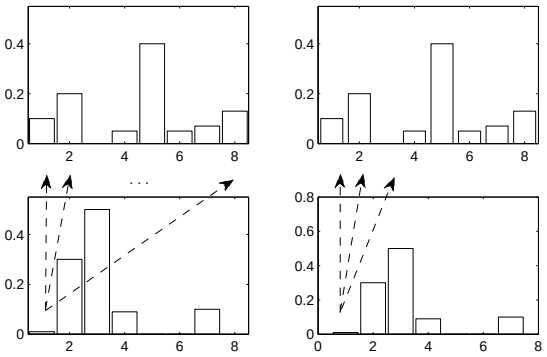


Figure 10.4: Hausdorff and *Simplified Hausdorff* Distance differences

Hence, we define the *Simplified Hausdorff Distance* as:

Definition 8. *Let's define o as the bin offset threshold, and P,Q two distributions, where P is the reference and Q the acquired distribution, with n and m elements respectively. We define the "Simplified Hausdorff Distance" as:*

$$h_S(P,Q) = \max_{i=1\ldots n} \{ \min_{j=i-o}^{i+o} \{g(P_i, Q_j)\}\} \tag{10.3}$$

With this enhancement, the *Simplified Hausdorff Distance* has $O(n) = n$, linear cost for small values of o $(0 \leq o \leq n)$, which in fact are the most useful in our context.

10.2.4 Behaviour of the distance computation

Both Kullback-Leibler Divergence and Hausdorff Distance work very well with stationary traffic and network conditions, as an example Figure 10.5 shows the same tests we discussed previously in Section 10.1.2 with poissonian traffic.

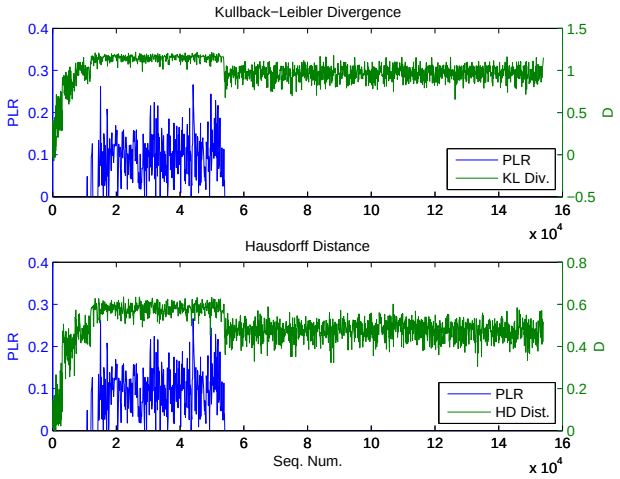


Figure 10.5: Kullback-Leibler and Hausdorff Distance behaviour

The figure shows clearly how the distance values adjust tightly to the PLR. The abrupt increase of the distances is due to the increasing one-way delay value, which is kept constant for the rest of the test. Even if this figure is with optimal conditions for the metric, it illustrates the good properties of the distance calculation.

10.3 SLA Violation Detection

Computing the distance among distributions determines how different are the reference and the acquired traffic profiles at the egress node, but this information alone is not sufficient to perform the SLA Assessment. In this section we present the methodology to detect SLA violations by using the IPAT information.

10.3.1 General Methodology: training and SLA violation detection

Informally, the base algorithm we use for the violation detection is the following: first, we collect the IPATs during a time period at the egress node. Second we compare their distribution with a reference distribution set. If the distributions are *similar*, we assume similar network behaviour in both cases. Otherwise, we query the ingress node to acquire detailed packet information, from which, we compute the real performance metrics. Finally we can assess the status of the network and report any encountered SLA violations.

In summary, we use the performance metrics to map the IPAT distribution to the real network status, and we use such distribution as a reference to infer the SLA compliance.

Algorithm 4 shows the general logic to perform the detection of SLA violation, regarding the collection of IPAT distributions, it is performed in a per flow f basis, during a predefined time interval t. The empirical distribution is computed in bins of width w (referring to IPAT ranges), therefore, a particular IPAT i falls in bin $k = \lfloor \frac{i}{w} \rfloor$. We defer the study of proper t and w values to Section 10.4.

As a simple example, Figure 10.6 illustrates this behaviour, we assume a single flow and a single time interval.

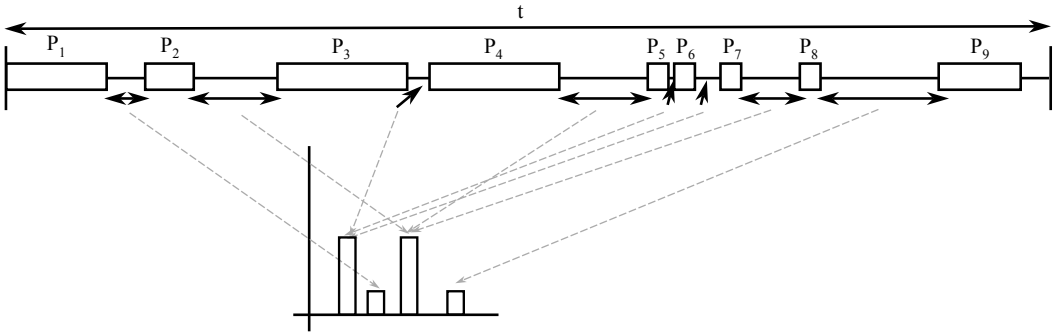


Figure 10.6: Flow to distribution process

After the distribution acquisition, to cope with the aforementioned IPAT variability, our algorithm considers the following actions:

- *Training and update* the traffic profile.
  - *Map* it to the actual network status.
  - *Update* the set of valid distributions if needed.
- *Compare* the current profile with the learned status.
  - *Decide* whether the traffic conditions changed or not.

The rest of the section performs a step by step description for the algorithm. We start by how the system learns the real offered network quality. Later we focus in the distributions comparison.

Algorithm 4 SLA Assessment

Input: f, $\mathcal{D}$ {f : Current Flow, $\mathcal{D}$: Global RDS}
$S \leftarrow getFlowSourceMP(f)$ {Source Monitoring Point}
$Q \leftarrow acquireDistribution(f)$
if $\mathcal{D} = \varnothing$ **then**
  $status \leftarrow Training(Q,S)$ {Does metric computation}
else
  $status \leftarrow compareDistributions(Q,S)$
end if
if $status < \nu$ **then** {Not valid network conditions}
  trigger $SLAViolation(status)$
end if
Output: status

10.3.2 Training and Update Strategy

Any system with adaptability requirements must have a robust Training mechanism. Before entering with the full description of the Training and Update Strategy, we need to define a few concepts.

Definition 9. *A Valid IPAT Distribution (VID) $\mathcal{V}$ is such a distribution where a function of the real metrics (OWD, IPDV and PLR) fall above a specified SLA threshold* ν. *The complete discussion about how $\mathcal{V}$ is computed is done in Section 10.3.4.*

Definition 10. *Let's define a Reference Distribution Set (RDS) $\mathcal{D}$, as a set of strictly different VID distributions. Where $|\mathcal{D}|$ is the cardinality of the set, $\mathcal{D}^1$ is the first element and $\mathcal{D}^{|\mathcal{D}|}$ the last one, with a maximum size for the RDS bounded by a predefined* Δ*, which limits the maximum memory usage of the RDS.*

The *Training and Update Strategy* is in charge of keeping an updated and valid version of the RDS. All the details are shown in Algorithm 5.

A prerequisite of the RDS is that all the stored distributions must represent good reference traffic conditions to compare with. Therefore, our system must have some means for correctly assessing them; we use the technique presented in [107], where the source ME (i.e., the ingress router) sends per packet information, such as transmission timestamps, which are matched on the destination ME/PE (i.e., the egress router), computing the relevant performance metrics.

Algorithm 5 Training and Update

Input: Q, S {Q : IPAT distribution, S : Flow's Source}
$status \leftarrow queryValidity(Q,S)$ {Computes Equation 10.5 and generates control traffic by acquiring the real metrics}
if $status < \nu$ **then**
  return $status$ {Do not keep Q in RDS, SLA violation}
end if
if $|\mathcal{D}| \geq \Delta$ **then**
  $expireLatestNotUsed(\mathcal{D})$ {Expiration policy}
end if
$\mathcal{D} \leftarrow \mathcal{D} \cup Q$
Output: status

This technique requires control traffic from source to destination to compute the metrics as discussed before. For the sake of efficiency and scalability it should be minimised. Nevertheless, this method reports exact values of the network metrics that we can use to map to the IPAT distribution.

Once the *real* validity is assessed, if it is below ν, the event is registered, the distribution discarded, and a *SLAViolation* event is triggered (see step 10 in Algorithm 4). Otherwise we insert the distribution on the RDS. In the case $\mathcal{D}$ is full we discard the oldest not used distribution. We propose this replacement algorithm for two different reasons: *i)* it is very efficient and easy to implement; *ii)* it honours the fact that if some distribution has not been representative of the traffic profile for a while, it is because the network status has changed; so the old distribution is not required anymore.

This Training algorithm is only invoked when the distribution comparison is not accurate (i.e., when network conditions are unknown).

10.3.3 Distribution comparison

The metric we chose to compare between two distributions is the distance, which can be described as *how far* one distribution is from another, or preferably, as the degree of similitude (dissimilitude) between two distributions. The higher the distance the more different the distributions (and so does the traffic profile).

Since RDS is composed by a set of distributions, the distance cannot be directly computed. Hence we define:

Algorithm 6 compareDistributions

Input: Q, S {Q :Acquired Distribution,S : Flow's Source}
$minD \leftarrow \infty$
for all $D \leftarrow \mathcal{D}$ **do**
  $d \leftarrow computeDistance(D, Q)$
  if $d < minD$ **then**
    $minD \leftarrow d$
    $P \leftarrow D$
  end if
end for
if $minD \geq \delta$ **then**
  $status \leftarrow Training(P, S)$ {Does metric computation}
else
  $updateUse(P)$ {Renew usage of the distribution to prevent expiration when $|\mathcal{D}| = \Delta$}
  $status \leftarrow getValidity(P)$ {Use value of *queryValidity*}
end if
Output: status

Definition 11. *Degree of Matching D_M, between a RDS $\mathcal{D}$ and another distribution Q is defined as:*

$$D_M(\mathcal{D}, Q) = \quad \min\{d(p, Q)\} \quad p = \mathcal{D}^1 \ldots \mathcal{D}^{|\mathcal{D}|} \tag{10.4}$$

where $d(p, Q)$ is a predefined distance algorithm between the distributions p and Q as presented on Section 10.2.

Then Q and $\mathcal{D}$ are considered similar if $D_M(\mathcal{D}, Q) \leq \delta$, where δ is our distance threshold. The critical point here is that different distributions do not mean different qualities, since the traffic profile can change over time, even with the same quality level. Therefore, when the distributions are considered different the system must learn the degree of quality of the new distribution. The *Training* procedure is then invoked with Q. Because training consumes system resources, as described previously, there is a trade-off here: the lower δ, the more resources (queries) will be needed. On the other hand, the higher δ, the lower amount of resources will be required, at the cost of losing some accuracy on the SLA assessment. The evaluation section has an extensive discussion about the effects of changing δ.

The complete pseudocode for this function is detailed on Algorithm 6.

10.3.4 Distribution Validity $\mathcal{V}$

The Training procedure queries the source monitoring point of the flow in order to get the *real* QoS parameters of the specified time interval. Once the real metrics have been acquired, the destination has to validate whether the SLA is being honoured or not.

In our work, as a proof of concept, we assume a simple linear SLA compliance policy, but any other function complying with the below restrictions can be applied seamlessly.

$\mathcal{V}$ is defined in the range $[0,1]$ indicating the quality of service experienced for the flow with respect to the SLA. Therefore, 1 stands for perfect quality, while 0 is absolute lack of it. To compute this value we consider the usual metrics (OWD, IPDV and PLR).

Another point to consider is that depending on the type of traffic, the QoS constraints might considerably differ, for example, videostreaming is robust to large OWD and high IPDV, but not to PLR, while videoconferencing is sensible to all the metrics. Hence, we define $\omega_O, \omega_I, \omega_P$ as weights specified for each particular metric, where $\omega_O + \omega_I + \omega_P = 1$.

Expression 10.5 computes $\mathcal{V}$, which is the degree of validity of the time interval.

$$\mathcal{V} = \mathcal{Q}^O(\overline{OWD}) \cdot \omega_O + \mathcal{Q}^I(\overline{|IPDV|}) \cdot \omega_I + \mathcal{Q}^P(PLR) \cdot \omega_P \tag{10.5}$$

Where $\mathcal{Q}^*(x)$ determines the exponential quality degrading function for the metric "$*$", defined as:

$$\mathcal{Q}^*(x) = \begin{cases} 1, & x \leq \mathcal{X} \\ \lambda e^{-\lambda(x-\mathcal{X})}, & \mathcal{X} < x < \mathcal{M} \\ 0, & otherwise \end{cases} \tag{10.6}$$

Where $\mathcal{X}$ is the metric dependent threshold of quality degradation specified by the SLA, and $\mathcal{M}$ the upper feasible bound for the quality of that particular metric. Finally, λ in $(0,1)$ is the decaying factor for the exponential quality degradation function.

Then $\mathcal{V} \geq \nu$ the network behaviour is considered stable. The closer is ν to 1, the stricter our system will be to SLA violations.

10.4 Theoretical Parameter Study

During the description of our SLA Violation Detection mechanism we have pointed out different parameters that can completely change the behaviour of the system. Such parameters are: the acquisition interval t, the bin size w, and the distance δ. In this section we perform a general

theoretical analysis of the effects of changing these parameters. The experimental study is left for Section 10.7.

10.4.1 The acquisition interval *t*

The acquisition interval is the first critical parameter we have to study to tune our SLA violation detection system. Tweaking such parameter has a number of implications depending on its size.

Here we detail separately the different effects caused by increasing and decreasing t.

Increasing *t*

Increasing the acquisition time interval has the following *positive effects*:

- *More samples to consider:* This implies that we will have more statistical soundness as the number of samples increases. Hence, our distributions are more statistically representative of the real network status.
- *Better Training periods:* If the acquired traffic is suitable to be part of the RDS then its distribution will be more indicative since it considers larger time periods. From the practical point of view this implies that the reference distributions will be more valid.

On the other hand, having large sizes of t derives in a number of *negative effects*:

- *Increased lag on the reporting:* The most important implication is that t determines the degree of *"Reactivity"* of the system, (i.e., the response time of the reporting). To understand this point, we have to have a clear idea about the system behaviour in general. First we gather IPAT during t time units, after this we compare its distribution with the RDS, and if necessary, we ask the ingress point about particular network metrics. The ingress point responds back with per packet information.

  As it can be noted, one of the critical parameters for this response time, is the time t itself, which bounds the time required to compute the distances, and therefore, start the violation detection process.

  Other factors such as the time invested by the comparison algorithm, or the network latency, are also relevant but out of the scope of this particular study.
- *More probability of varying traffic:* The longer we get statistics the more probable is that we have variations on the underlying network, which on its turn makes it more difficult to have representative distributions.

Decreasing *t*

Reducing the acquisition time *t* also has positive and negative effects over the system behaviour. The *positive implication* are:

- *More interactivity on the reporting:* the less time we spend on the gathering the quicker we will react to any SLA violation.

- *Less samples:* Having less samples implies that we will have faster computation, hence, faster reporting.

Nevertheless, it also has a number of *negative effects*:

- *Less samples:* At the same time of being a good thing, having less samples means that we will have less statistical information in our distribution, therefore, we could incurr in inaccuracies in the detection.

- *More variability:* Derived from the previous case, the less samples we have, the higher is the probability of having different distributions. Therefore, we will have more queries to the ingress node (even if these queries are faster and smaller because of the reduced number of samples per time interval). This does not necessarily mean that it will require more resources. In fact, as we show in below sections decreasing *t* usually implies also a reduction in resource requirements.

10.4.2 The bin size *w*

As we discussed previously, one of the most critical parameters of our SLA violation detection system is the bin size *w*.

Intuitively, *w* determines the degree of granularity of our system. Hence, for little values of *w*, we will have high resolution, which means that we distiguish among closer values per bin. On the other hand, if we increase the bin size, we are losing resolution, that is, we are aggregating a larger set of IPAT within the same bin. This, as expected, has a number of effects over the final result. In this section we perform a basic theoretical study about the effects of tuning the bin size. We defer the experimental analysis to section 10.7.2.

Increasing w

As introduced before, increasing w reduces our resolution in terms of higher aggregation degrees in our bins. When we have larger w, we can observe different effects. The good properties of having large w are:

- *Less bins to analyse:* Since our aggregation level is higher, for a given tests we are reducing the number of bins to consider per t interval.
- *More efficient:* Derived from the previous advantage, having less bins implies less processing time to calculate the IPAT, generate the distributions, and perform the distance comparisons.
- *Less network resources:* Reducing the amount of bins implies that we will have less probability of having different distributions. We can formally prove this:

  Proof. Let's define p as a reference distribution, and Q as the acquired distribution. Assuming that $w \to \infty$ for the acquisition interval then the probability P of having exact IPAT distributions is determined by:

  $$\lim_{w \to \infty} D_M(p,Q) = 0 \tag{10.7}$$

  Therefore, $P(D_M(p,Q) \leq \delta) = 1$ for any p, Q, and δ. Hence, the system will consider all the distributions as equal.

  On the other hand, if $w \to 0$ we have infinite bins, by basic statistics we know that $P(D_M(p,Q) \leq \delta) \to 0$ since we have more bins to match, therefore having two exact distributions is much less probable. □

The final effect then is that we invoke the *Training* process fewer times, hence, reducing the required network resources. But this reduction of the resources is not for free, it has some *bad implications* as we detail now:

- *Less accuracy:* Analogously to the fact that we have higher probability of having similar distributions, we will consider different traffic profiles as equal with higher accuracy, which in practice means that we will reduce the detection of SLA violations, mostly due to lack of *Training*.

Decreasing *w*

In the case that we are reducing the bin size, we will face opposite effects than before. In particular, the *positive effects* of decreasing *w* are:

- *Finer accuracy:* The more bins we have to compare, the more resolution we have in order to distinguish different IPAT distributions. This leads to the fact that in general we have more accuracy in the reporting.

But, as expected, this also leads to some *negative properties*:

- *More computation time:* Derived from above, if we have more bins to analyse, it implies that we will need more time to compute the IPATs distributions so we need more CPU power to do the on-line detection.
- *More network resources:* If we have finer detail in the IPAT distribution as detailed above the probability of equal distributions decreases, meaning that we invoke the *Training* process more often, which leads to higher network resource requirements.

As it can be noted there is a trade-off in this situation, normally we desire good accuracy, but in practice the network resources reserved for control traffic are limited. In Section 10.7.2 we experimentally show such effects.

10.4.3 The Distance threshold δ

The last system parameter to study is the distance threshold δ. It determines the confidence we have in our distance algorithm, high values represent loose constraints on the distance comparison, while small values indicate tight constraints that trigger the *Training* process more easily. Therefore, there is a trade-off selecting the size of this parameter.

Increasing δ

Increasing δ implies that we are relaxing the constraints of our distace algorithms. In practice, this means that we consider as similar very different distributions.

In detail, since the outcome of the distance algorithms range in $[0..1]$, for values of δ closer to 1 the algorithm cannot distinguish among the distributions and all of them are considered as equal. This, as expected, comes with an associated loss in accuracy.

The good side of this increase, is the considerable reduction in control traffic derived of such action. If we consider two distributions as equan we do not need to invoke the *Training*, thus, we reduce the required resources.

Decreasing δ

Logically, decreasing δ, gives the opposite effect, this means that for $\delta \rightarrow 0$ we are tightening the constraits of our similarity threshold, which make more difficult for our algorithm consider two different distributions as similar. This imples that we will increase the accuracy, toghether with an increase in the used network resources.

10.5 Tests and Testbeds

In order to validate our proposal we set up three different testbeds, where we performed several tests. In particular, *i)* synthetic traffic under controlled testbed conditions, *ii)* synthetic traffic over the European Research network Gêant, *iii)* real traffic over a controlled testbed.

10.5.1 Synthetic traffic under controlled testbed conditions

The first set of tests have been performed under a tightly controlled environment. We configured two end nodes with Linux Debian in order to generate and collect traffic. On the core of the testbed we installed two servers also with Linux Debian, Traffic Control and `NetEM` emulator capabilities. We then can change the network conditions according to our needs and experience a wide range of controlled network disruptions.

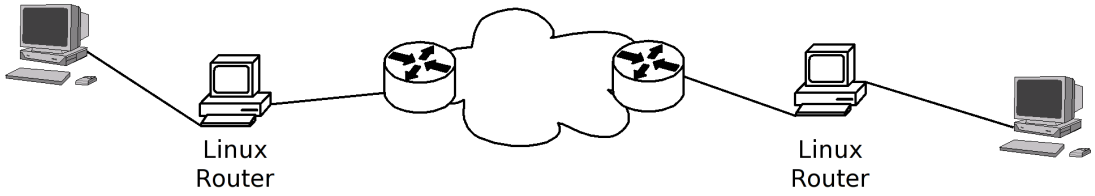


Figure 10.7: Controlled testbed

All the links on the testbed were configured with Fast Ethernet and no cross traffic. Furthermore, all the Linux boxes were synchronised by using GPS signal with NTP and PPS patches on the kernel for accurate timestamping.

In this testbed the set of emulated network conditions are:

1. *Good Network Conditions:* no SLA disruptions and good network behaviour all over the test.

2. *Mild Network Disruptions:* moderated increase of OWD with periods of high IPDV and some packet losses. Some traffic disruptions but only with few SLA violations per test.

3. *Medium Network Disruptions:* similar to the mild network disruptions but with limited buffers on the routers which leads to moderate periods of packet losses. Some SLA violations in many intervals during the test.

4. *Severe Network Disruptions:* random losses from 1% to 10% with variable OWD. Severe SLA violations in periodic intervals on the test.

All the tests have in common that the SLA disruptions are applied at regular time intervals all over the tests, combining periods of good behaviour with others with disruptions.

We performed tests with *Periodic*, *Poissonian* and *Synthetic Real Traffic* [93] traffic profiles with all the above network conditions. Since the controlled traffic conditions are tightly controlled, to have meaningful results it was enough to repeat 3 times each set of tests. Using these traffic profiles we have from predictable packet rates *(Periodic)* to unpredictable realistic profiles *(Synthetic Real Traffic)*.

10.5.2 Synthetic traffic over the European Research network

In this testbed we performed a set of more than 500 experimental tests during 2006 and 2007 using twelve different testbeds across Europe. We performed the tests at different hours, including weekends, to have a real variety of cross traffic and congestion. The testbeds were provided by the IST-EuQoS [45] partners, covering a total of 5 countries and 4 different access technologies (LAN, xDSL, UMTS and WiFi) with an overlay architecture over the Gêant research network.

We evaluated the performance of our system by actively generating UDP traffic on the network with different properties. Specifically, we generated periodic flows, with varying packet rates, from 16 to 900 packets per second among all the involved nodes in the testbed. We used different packet sizes ranging from 80 to 1500 bytes per packet. Specifically, we focus on three different sets of tests. The first one simulates a low rate, small size packets with a used bandwidth of $64Kbps$. We label this traces as (synthetic) `VoIP`.

The second type of traffic is a periodic flow with average packet rate of ~ 96 packets per second, with MTU size packets amounting to a total of $1Mbps$ of UDP traffic. We call this trace `UDP1`. Finally, the third kind of traffic is a average sized, high rate UDP flow with $\sim 1.4Mbps$. We call this test `UDP2`.

10.5.3 Real traffic over a controlled testbed

Generating synthetic traffic gives tight control over the different characteristics of the traffic to stress: rate, packet size, etc., but on the other hand, it does not reflect how a real application performs. Therefore, in order to have insights about the behaviour of our system with real applications we used the local testbed described before in Section 10.5.1 with a video streaming application, namely VLC, transmitting high quality video with variable bit rate over the network. In the same fashion as before, we inserted various degrees of network disruption to analyse the accuracy of our SLA assessment system.

10.6 Evaluation

The validation of the proposal is issued by performing a wide variety of tests in different testbeds as described on the previous section.

We focus the study in the system's accuracy, that is, in the SLA violation detection rate, measured in terms of false negatives (i.e., not detected SLA violations). We compare the three presented distance algorithms against the case of having perfect knowledge about the SLA violations. We also analyse the amount of resources required by the system; such resources are counted in terms of reduction ratio of the required bandwith used by the control traffic. Therefore, we compare the cost of reporting per packet information with our solution, which only demands information when there is a change in the traffic reception profile.

During all the analysis we use the same parameters across the tests for the estimation. In particular, we set up, as a proof of concept, the following values: distance threshold of $\delta = 3\%$, bin width of $w = 3ms$, and an acquisition time interval of $t = 175ms$. The discussion about the different parameters, such as distance threshold selection, or time bin size is done in Section 10.7.

10.6.1 Methodology

Analysing all the information obtained from the tests is complex. To ease the comprehension of the validation process, we unify the evaluation for all the tests and testbeds under the same methodology as follows:

1. For each test we collect the full trace on both end nodes.
2. We match the packets extracting the network performance metrics as described in [107], using them as reference quality (perfect knowledge), since the process gives exact results.
3. We identify the different SLA violation periods with the reference results acquired above.
4. We apply off-line our algorithm (by using *Kullback-Leibler*, *Hausdorff* and *Simplified Hausdorff*). Here we register: *i)* required control traffic due to *Training*. *ii)* estimated SLA violation periods.
5. Finally, we match the SLA violations with the ones obtained in Step 3.

We apply our system off-line with the goal of comparing the results. But in the actual deployment the system performs on-line assessment. By using this methodology we can guarantee accurate comparison between the perfect knowledge of the network behaviour and our system.

10.6.2 Accuracy and Resources requirements

In order to study the behaviour of our system, here we discuss the achieved accuracy together with the analysis of the required resources for each algorithm in the different testbeds.

10.6.2.1 Synthetic traffic with controlled network

The goal of this synthetic traffic generation is to evaluate the reaction of each algorithm in a controlled environment with the different traffic profiles.

We analyse in Table 10.1 the *Accuracy* and the *Resource* utilisation for the different generated traffic. The accuracy is computed for the overall test duration, counting the ratio of detected SLA violations over the total, while the required resources are computed by the ratio of the actual number of queries, over the maximum possible queries per test. Our goal is to achieve high accuracy with low resource consumption.

(a) Periodic

	Accuracy			
	Good	Mild	Medium	Severe
KL	1.000	1.000	0.987	1.000
Hausdorff	1.000	1.000	0.088	1.000
D. Haus.	1.000	1.000	0.868	1.000
	Resources			
	Good	Mild	Medium	Severe
KL	0.001	0.256	0.385	0.394
Hausdorff	0.001	0.020	0.019	0.394
D. Haus.	0.001	0.130	0.267	0.394

(b) Poisson

	Accuracy			
	Good	Mild	Medium	Severe
KL	1.000	0.250	0.940	0.893
Hausdorff	1.000	0.750	0.067	0.225
D. Haus.	1.000	1.000	0.994	0.999
	Resources			
	Good	Mild	Medium	Severe
KL	0.562	0.572	0.657	0.671
Hausdorff	0.122	0.143	0.132	0.190
D. Haus.	0.464	0.601	0.699	0.721

(c) Synthetic Real Traffic

	Accuracy			
	Good	Mild	Medium	Severe
KL	1.000	0.667	1.000	1.000
Hausdorff	1.000	1.000	1.000	1.000
D. Haus.	1.000	0.667	1.000	1.000
	Resources			
	Good	Mild	Medium	Severe
KL	0.002	0.002	0.397	0.397
Hausdorff	0.002	0.003	0.397	0.397
D. Haus.	0.002	0.002	0.397	0.397

Table 10.1: Accuracy and Resources for $\delta = 0.03$

As it can be observed in the table, the accuracy of the solution is higher for the extreme cases. When there are *Good* network conditions in the network we always estimate correctly, and with very low resource consumption in general. This is because our algorithm assumes correct network behaviour by design. In the case of *Severe* network conditions, where our contribution is more useful, we can detect with very good accuracy the SLA disruption periods. On the other hand, in the fuzzy case when there are few SLA violations, the accuracy of the system drops sensibly for some algorithms. The cause of this is the statistical resolution achieved when there are few SLA violations, where missing only one violation is statistically significative. Moreover, in a real deployment, such SLA violations are of no practical interest

since they represent very short, sporadic, periods of light congestion, with no high impact on the final network behaviour.

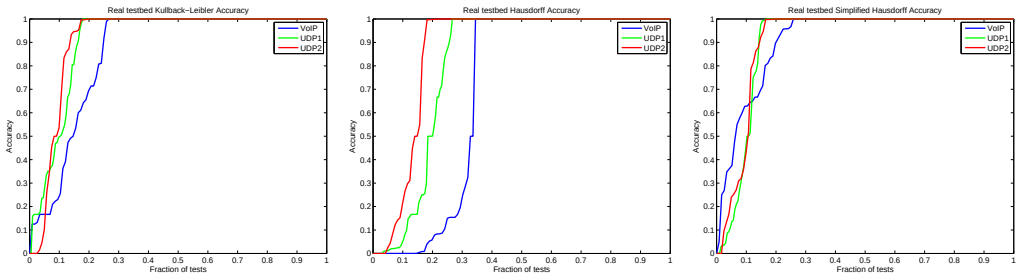
Comparing the various distance algorithms we can notice some general properties: first, the better accuracy in most cases is achieved by the *Simplified Hausdorff Distance* proposed as an extension in this work, with similar results for *Kullback-Leibler*. It is also interesting to highlight the poor performance obtained with *Hausdorff*, except in the case of synthetic real traffic. This is caused by the *"all-against-all"* comparison we pointed out previously.

The second consideration is the resources needed by the *Severe*, and some *Medium* network conditions. As it can be noted, in some traffic profiles, the results are exactly the same regardless of the algorithm used. This is because the algorithms always query the ingress node when an unknown IPAT distribution is found. With bad network conditions this situation is common. Hence it forces the system to query for exact metrics. Here the minimum number of queries is bounded by the amount of SLA violations, which in our experimental case is 0.39 as shown in the Table 10.1. In the specific case of Poissonian Traffic, we need more resources than this lower bound for *Kullback-Leibler* and *Simplified Hausdorff*. Notice though, that requiring less resources than that implies non detection of some SLA violations.

10.6.2.2 Synthetic traffic over the European Research network

In this testbed we plan to show the proper accuracy of our proposal in a real network with random quality, unexpected results and unknown cross traffic with different multi-hop paths.

In Figure 10.8 we show the different accuracy results for each algorithm and traffic profile. The X-axis of the figure has the test number (normalised to 1) and the Y-axis the accuracy. The figure considers all the tests, including the ones without SLA violations.



(a) Kullback-Leibler (b) Hausdorff (c) Simplified Hausdorff

Figure 10.8: Accuracy for Synthetic traffic over the European Network

We complement the figure with Table 10.2, which summarises the results of our experiments. We show the aggregated total amount of bins with violations, together with the amount our algorithm could detect. In the third column we highlight the overall accuracy and finally, the last column, details the average amount of resources needed for the reporting.

(a) Kullback-Leibler

	Violations	*Detected*	*Accuracy*	*Resources*
VoIP	7216	6096	0.845	0.198
UDP1	62264	60108	0.965	0.551
UDP2	24863	22265	0.896	0.338

(b) Hausdorff

	Violations	*Detected*	*Accuracy*	*Resources*
VoIP	7216	3163	0.438	0.024
UDP1	62264	58003	0.932	0.237
UDP2	24863	21384	0.860	0.221

(c) Simplified Hausdorff

	Violations	*Detected*	*Accuracy*	*Resources*
VoIP	7216	4765	0.660	0.044
UDP1	62264	59265	0.952	0.246
UDP2	24863	22345	0.899	0.232

Table 10.2: Violation detection under a real network, $\delta = 3\%$

It is important to notice that most of the failures in the SLA estimation are due to isolated bins with violations very close to the SLA agreement boundary with no practical interest, similarly to the case we found in the previous testbed.

In this set of tests, the best performing algorithm is *Kullback-Leibler*, but at the expenses of using more network resources than the other alternatives. As in the previous case, *Hausdorff* falls behind in terms of accuracy, which added to its higher computational complexity, makes it the worst presented algorithm.

In terms if resources, the average resource usage of the whole system is below 25%. It is lower when considering `VoIP` traffic (i.e., around 4%), while the minimum amount of resources required is $\sim 5.8 \cdot 10^{-4}$, and the maximum is 1 (meaning no reduction is achieved). Further investigating these tests causing more resource usage, we found that they are the ones representing highly congested links (in particular xDSL), with very high loss ratios and large

amount of SLA violations (around 90% of the bins affected with packet losses). This forced a large increase in the resource requirements in these cases as expected.

10.6.2.3 Real traffic over a controlled testbed

In this last set of tests, it is intended to observe the performance of our algorithms under real traffic with controlled network behaviour, in fact, the forced SLA violations on the testbed follow the same patterns as we introduced in the synthetic traffic case (i.e., *Good*, *Mild*, *Medium* and *Severe*).

We used VLC to perform the tests. Since the application generates two flows, one for audio and the other for video, we show them separately in Table 10.3. There, we can see the overall accuracy for the three algorithms.

(a) Kullback-Leibler

	Accuracy			
	Good	*Mild*	*Medium*	*Severe*
Audio	1.000	1.000	0.999	0.997
Video	1.000	0.450	0.621	0.874
	Resources			
	Good	*Mild*	*Medium*	*Severe*
Audio	0.962	0.966	0.947	0.728
Video	0.061	0.074	0.085	0.569

(b) Hausdorff

	Accuracy			
	Good	*Mild*	*Medium*	*Severe*
Audio	1.000	0.997	0.969	0.898
Video	1.000	0.450	0.610	0.758
	Resources			
	Good	*Mild*	*Medium*	*Severe*
Audio	0.019	0.016	0.023	0.086
Video	0.022	0.025	0.038	0.352

(c) Simplified Hausdorff

	Accuracy			
	Good	*Mild*	*Medium*	*Severe*
Audio	1.000	0.999	0.979	0.975
Video	1.000	0.450	0.677	0.848
	Resources			
	Good	*Mild*	*Medium*	*Severe*
Audio	0.114	0.114	0.144	0.397
Video	0.032	0.035	0.050	0.407

Table 10.3: Overall detection and resources for VLC traffic, $\delta = 3\%$

The accuracy of the studied SLA violations is higher than 99% for the audio flows, dropping sensibly in the case of video flows. The causes of such difference in accuracy are yet unknown

and subject to further study. Nevertheless, in the case of *Severe* network conditions the accuracy is still higher than $\sim 85\%$ in both algorithms. Again, in this case *Hausdorff* falls behind in terms of accuracy but keeping a very low resource consumption.

Regarding the resources, for audio flows, in the case of *Simplified Hausdorff*, they range from $\sim 11\%$ for *Good* network behaviour to $\sim 40\%$ in the case of *Severe* SLA disruptions. For the video flows it ranges from $\sim 3\%$ to $\sim 40\%$. As it can be noted, the cases for which the maximum resources are required are consistent with the ones found on the Synthetic Traffic with Controlled Network: the packet losses and delay constraints were similar as in this case. Again, using *Kullback-Leibler*, despite of having slightly better accuracy, requires $\sim 20\%$ more resources than *Simplified Hausdorff*.

10.7 Experimental Parameter tuning

Complementary to the theoretical analysis performed in Section 10.4, this section studies all the different experimental parameter selection and tunning for our system. Specifically, the considered parameters are: the acquisition interval t, the bin size w, and the distance δ.

All the parameters are studied by using the tests described above, in particular for the study of t and w we use *Real Traffic over a Controlled Network*, with the VLC flows. Regarding the δ we use the *Poissonian* flows generated under the EuQoS project.

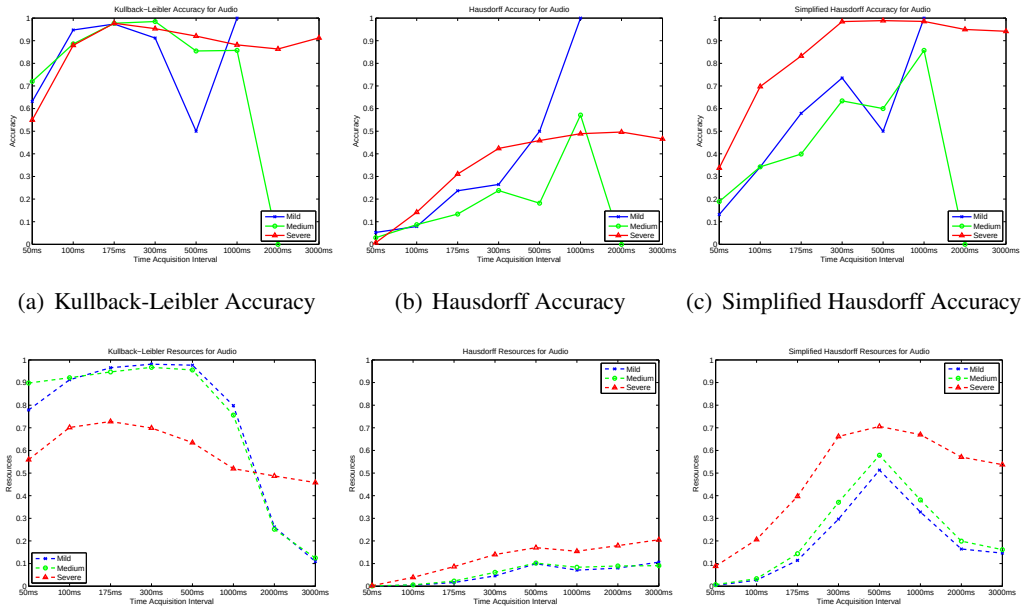
10.7.1 The acquisition interval t

As we discussed previously, increasing the acquisition time interval gives our system more statistical soundness, due to the presence of more samples in the acquired distribution. But at the same time it gives less responsivity to our system.

We performed the analysis by doing tests with several values of t. In particular we set up our VLC testbed to use values of t: $50, 100, 175, 300, 500, 1000, 2000$ and 3000 milliseconds. Then we estimated the SLA violations by using our three algorithms.

Figure 10.9 and 10.10 sumarise the obtained results for all the values, in the results we omit the *Good* quality network since its accuracy is always 1 and the resource consumption is in all cases lower to the other kinds of network disruptions.

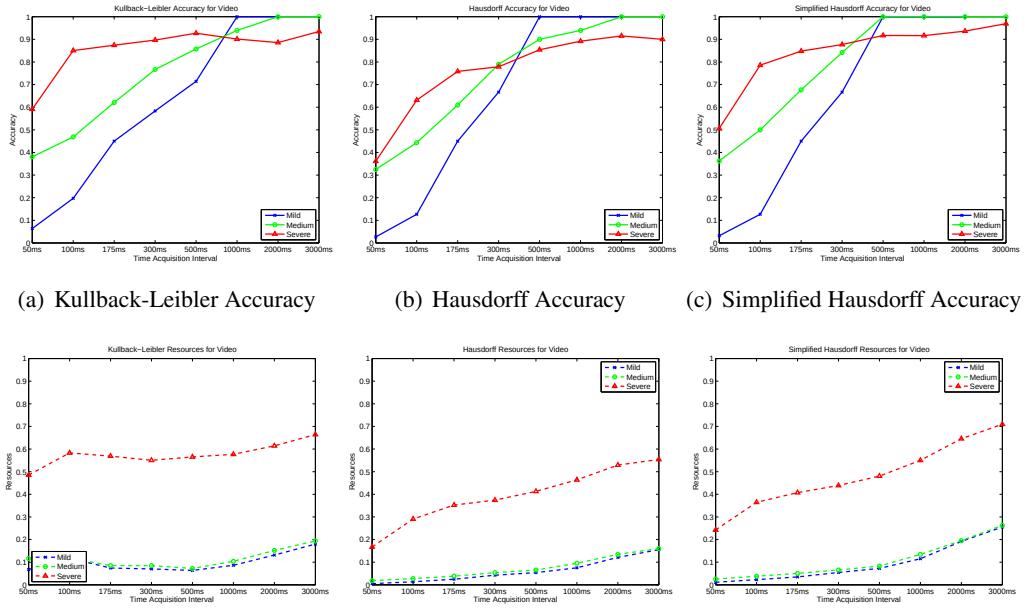
The results show the expected behaviour in all the cases. The trend is a clear improvement in the accuracy of the system as the time interval increases, the results are coherent holds with the theoretical analysis we performed in Section 10.4.



(a) Kullback-Leibler Accuracy (b) Hausdorff Accuracy (c) Simplified Hausdorff Accuracy

(d) Kullback-Leibler Resources (e) Hausdorff Resources (f) Simplified Hausdorff Resources

Figure 10.9: Time Acquisition interval effect over VLC Audio Traffic



(a) Kullback-Leibler Accuracy (b) Hausdorff Accuracy (c) Simplified Hausdorff Accuracy

(d) Kullback-Leibler Resources (e) Hausdorff Resources (f) Simplified Hausdorff Resources

Figure 10.10: Time Acquisition interval effect over VLC Video Traffic

As it can be noted there is a considerable drop in accuracy for audio traffic in the case of *Simplified Hausdorff* and in *Kullback-Leibler* for *Mild* traffic conditions (in fact it drops to 50%, in the specific case of 500*ms*). The cause for this is that when aggregating more packets on the bin, with few packet losses the aggregated quality for the period gets *"better"* in the sense that the packet loss ratio improves, therefore the number of total bins with violations drops (in our case to 2 for the whole test), and the three algorithms have more difficulty to detect them (in our case only one is correctly detected). Also as expected, this behaviour is repeated for the case of *Medium* traffic disruptions for higher bin sizes.

Similar results are obtained in the case of *Mild* traffic with $t = 1000ms$ and higher, where in this case the total amount of disruptions for our particular case is 0 therefore we can always estimate it correctly.

In the case of video flows, the results differ considerably, this is because this traffic uses higher packet rates, which at the end, implies that more packets get dropped, so, increasing the bin size helps the estimation.

As we already detected in the Evaluation section both *Simplified Hausdorff* and *Kullback-Leibler* have better accuracy than *Hausdorff* (specially for audio flows). Comparing both algorithms, in general *Kullback-Leibler* has better accuracy, but it requires too much resources in terms of bandwidth. In any case for high values of *t Simplified Hausdorff*, while using a fairly reasonable amount of resources accomplishes similar degree of accuracy.

Sumarising the findings about the accuracy, we can see that in general having large bin sizes helps the estimation, but this could be misleading, because for large timescales, small disruptions get undetected, while being relevant for shorter timescales. This is not a fault of our system but the actual reduction in the packet loss ratio per bin (therefore our traffic has a validity higher than ν), so depending on the requirements of our system we will have to select the appropriate values for t.

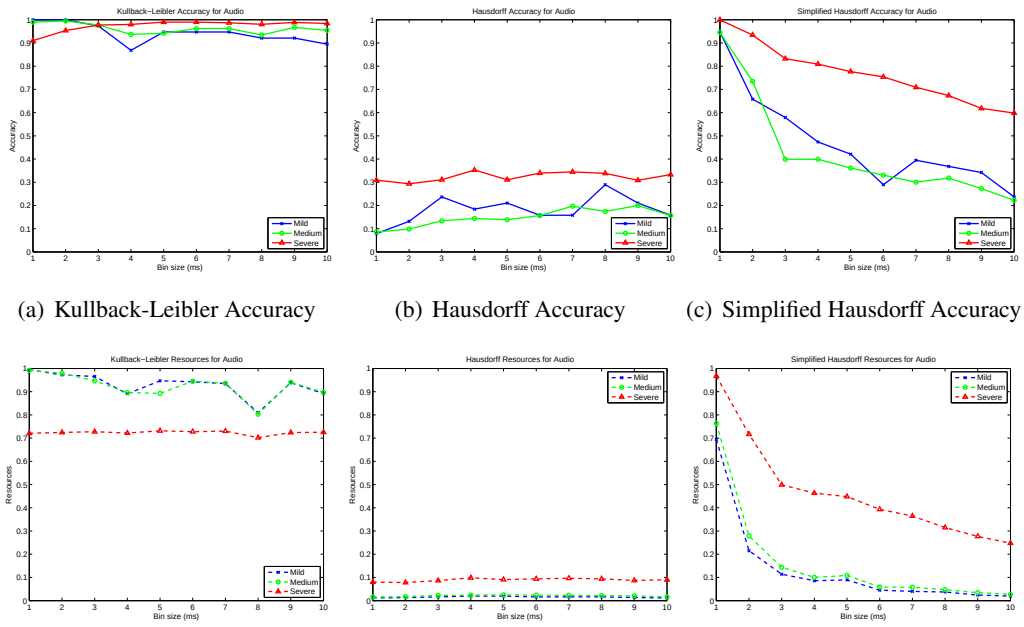
Regarding the required bandwidth resources, the most efficient algorithm is *Hausdorff*, but due to computational complexity, and specifically the lack of accuracy, it is not the best option for a production network. On the other hand, depending on the specific needs of the network under analysis, choosing *Kullback-Leibler* or *Simplified Hausdorff* are two good options, but the second tends to be more efficient in terms of resource usage.

10.7.2 The bin size *w*

As we already prompted, changing the bin size is critical to the accuracy of the system. In this section we study such effects from the experimental point of view. As a proof of concept we focus our study on the traces of *Real Traffic over a controlled testbed*, described before in Section 10.5.3, but our analysis could be extended to any other type of traffic in our system.

The analysis is performed by using w values from 1 to 10 milliseconds of IPAT. In the rest of the section we study these different values with the three proposed distance algorithms to see the effect over the accuracy and used resources.

In Figures 10.11 and 10.12 we show the results for *Kullback-Leibler*, *Hausdorff* and *Simplified Hausdorff* with values of $t = 175ms$, for the different w.

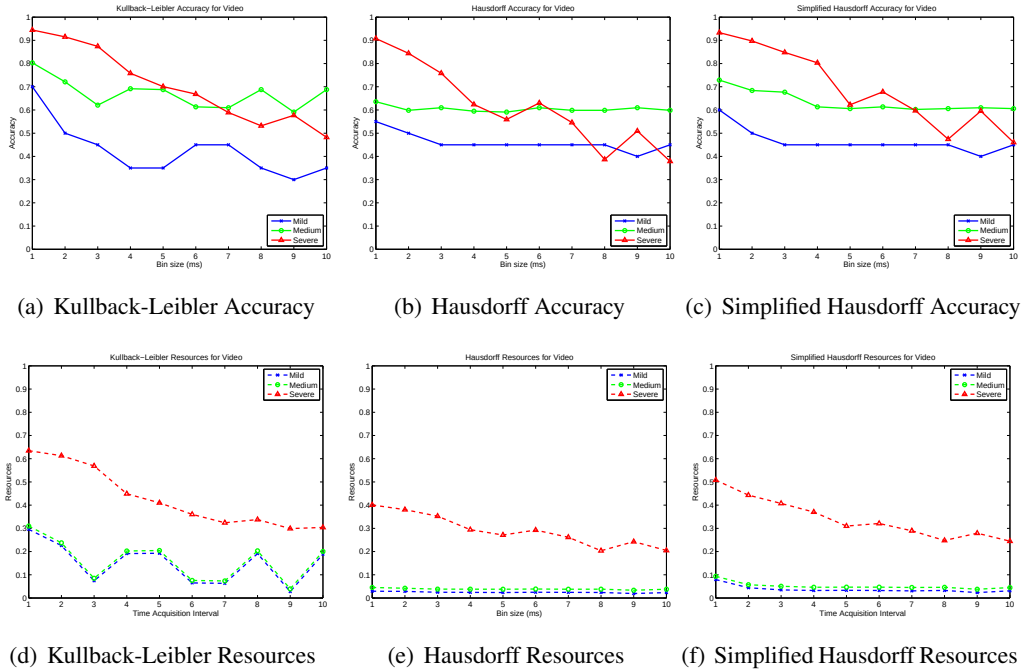


(a) Kullback-Leibler Accuracy (b) Hausdorff Accuracy (c) Simplified Hausdorff Accuracy

(d) Kullback-Leibler Resources (e) Hausdorff Resources (f) Simplified Hausdorff Resources

Figure 10.11: Bin Size effect over VLC Audio Traffic

It can be noted that the intuitive idea of resource consumption and accuracy is held in all the cases in general. In particular, for *Kullback-Leibler*, in the worst case the audio SLA Violation Detection accuracy drops from 1 to 0.89 for the *Mild* case. In the results obtained for *Medium*, the accuracy slightly increases for larger bin sizes, but this marginal gain can be considered as an outlier since is not the main trend in our results.



(a) Kullback-Leibler Accuracy (b) Hausdorff Accuracy (c) Simplified Hausdorff Accuracy

(d) Kullback-Leibler Resources (e) Hausdorff Resources (f) Simplified Hausdorff Resources

Figure 10.12: Bin Size effect over VLC Video Traffic

On the other hand, the video flows are more affected by the bin size increase, the reason being that such flows have a higher rate, this means that the IPAT tend to be smaller (in some cases lower than $1ms$). The issue with such small bin sizes is the huge grow in the amount of bins to analyse. And therefore, its computational demands.

With respect to the used resources, again, as expected the resource consumption for *Kullback-Leibler* with audio flows is consistent, reducing in ~ 0.10 units from $w = 1$ to $w = 10$ in general. While in the case of video flows, since we already are overestimating the good network behaviour, thus using less network resources, the differences are less noticeable even if decreasing as expected.

In the case of *Hausdorff*, the drop in accuracy both in audio and video is more noticeable in the *Severe* case. It is fairly well estimated for small bin sizes but its accuracy drops very fast as the bin size gets bigger. This highlights the point that increasing the bin size helps reducing the resolution, therefore, considering similar groups of IPAT which are very different in reality. Moreover here we can see in more detail the lack of accuracy provided by the *"all-against-all"* mechanism used by *Hausdorff*.

Looking closer at the results, using *Hausdorff*, it can be noted that for the case of $w =$

1 the accuracy is lower than in the rest of the cases. Although we need to investigate this behaviour with more detail, we can say that the most likely reason for this anomaly is the fact that aggregating IPAT in this case, since *Hausdorff* is not very accurate, played a positive role on the final result, again due to the *"all-against-all"* policy of the algorithm.

Regarding the used resources, differently than the unexpected behaviour of the accuracy. Their consumption is slightly reduced due to the increase of the bin size, but it is far less noticeable than in the case of *Kullback-Leibler*. This is mostly caused by the fact that, on the one hand, the bin size increase favors a reduction of the resources. While, on the other hand, the casual increase in accuracy favors the use of more resources as more queries are permitted.

Finally in the case of *Simplified Hausdorff* the obtained results are quite in-line with the ones obtained by *Kullback-Leibler*, in this case the accuracy is reduced, specially in the audio flows, where it drops from almost perfect estimation for the *Mild* case to the not so good 0.24 for the largest bin size. As expected in the *Medium* case the behaviour is similar. Regarding video flows the trend is similar but starting with less accuracy as we already discussed before. One interesting point to notice is that the bin size impacts more strongly to *Simplified Hausdorff* than *Kullback-Leibler*, the reason is that geometrical distances are more deterministic (are a real metric) than the relative entropies (that not are considered metrics), which being dimensionless do not have this spaciality difference expected on physical metrics.

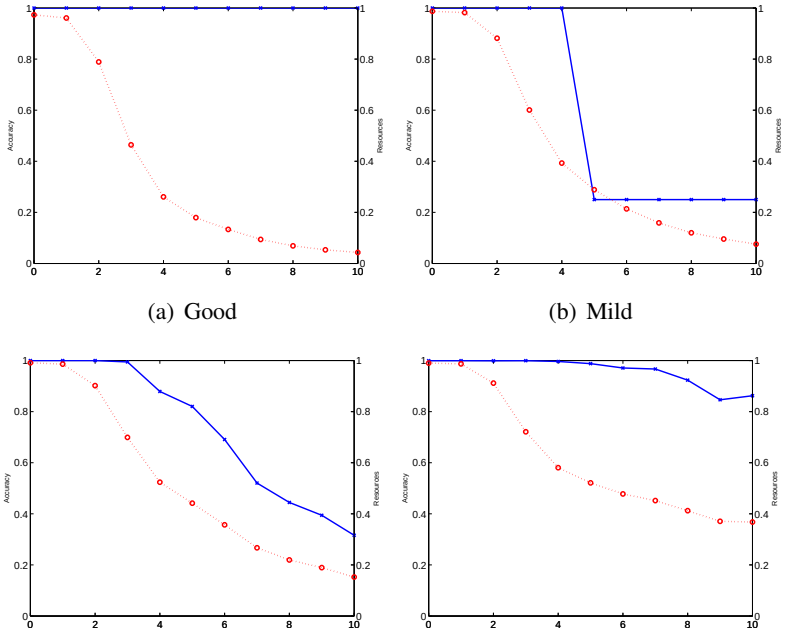
In *Simplified Hausdorff*, the used resources are largely reduced as the bin size increases as expected. This is in-line to the drop in accuracy we discussed before.

In summary, *Kullback-Leibler* and *Simplified Hausdorff* continue to be the reference algorithms, even if requiring more resources than *Hausdorff*, they deliver levels of accuracy which permit to deploy this system on a real enviroment.

10.7.3 Sensitivity analysis for δ

As we already discussed, querying the other end-point to acquire the network status is one of the bottlenecks of the system. The querying is triggered when $D(\mathcal{D}, Q) \geq \delta$ (Step 6 of Algorithm 5). Hence δ and the traffic itself determine the amount of queries of the system. Figure 10.13 details the effects of changing δ between 0 and 10% for controlled traffic generation with Poissonian Traffic using our algorithm *Simplified Hausdorff Distance* (the study would be analogous in the case of *Kullback-Leibler* and *Hausdorff*).

All the subfigures contain the Accuracy in the left Y-Axis with solid line, and the Resources needed on the right Y-Axis with dotted line. The X-Axis contains the different values for δ. As



(a) Good (b) Mild

(c) Medium (d) Severe

Figure 10.13: Distance Effect Poisson traffic. Accuracy and Resources are on the Y-Axis and each considered distance (in %) is on the X-Axis

it can be observed, in the case of Good and Severe traffic conditions, the effects of increasing the distance permits to reduce the required resources given the predictability of the outcome, specially in the Good case where the estimation is always correct. Again the important trade-off is on the fuzzy situation where not much SLA violations occur on the network, in that case, increasing the distance has a very noticeable effect on the final accuracy of the system, due to the statistical error incurred when having a small number of samples, being more noticeable when there are fewer disruptions (*Mild* case in the figure).

10.8 Discussion

In this work we presented a scalable SLA assessment system that draws the first steps towards reducing the overhead caused by the metric computation. In this section we complete our analysis by studying the cost of the solution in terms of computational demands. We also discuss the required steps to deploy our system in a real environment.

10.8.1 Overall Computational Cost for Simplified Hausdorff

To assess the feasibility of deploying the system we perform an analysis of the cost of the overall algorithm. All the expressions in this section are considered in a per-time interval t resolution.

The function *acquireDistribution* has a linear cost over the number of received packets n per flow in t: $O(n) = n$.

The critical path for *Training* is querying the Validity. This implies computing the OWD and the $IPDV$ of the time interval, which is linear over the number of packets. While PLR cost is constant. With a total cost of $O(n) = n$.

For the function *compareDistributions*, the cost consists of:

- Linear cost over $|\mathcal{D}|$ for the iterator of RDS.
- The cost of computing the distance. As described before, it is linear over the number of bins in the distribution ($|P|$).
- Training, which has linear cost as described before.

The total computational cost then is upperbounded by $O(n) = |\mathcal{D}| \cdot |P| + n$, knowing that it is not possible to have two Training periods in one round of the algorithm.

10.8.2 Deployment feasibility

All the basic steps of the algorithm are memory efficient and easy to implement in edge nodes of the network. It only involves computing IPAT, distances between distributions, and few things more. The only complex step is to give feedback to the system about the real metrics of the network, because it requires to match packets of the flows under analysis on the different edge nodes. This can be performed by using different techniques as described in our previous work [107; 109] with dedicated boxes forming an overlay network within the network under analysis. Therefore it is removing complexity from the edges.

10.9 Conclusions

We have presented a novel approach to on-line SLA Assessment, where differently of previous research, our work separates and reduces the performance metric computation and the interaction between the edge nodes of the network. This is accomplished by: *i)* a smart algorithm for

gathering the distribution of Inter-Packet Arrival Time (IPAT); *ii)* distance algorithms to compare the distributions; and *iii)* a robust Training methodology that delivers a very competitive solution regarding SLA violation detection.

As an additional contribution, we improved *Hausdorff Distance*, by using the knowledge about the data we are dealing with. With this improved version we can efficiently infer the network quality with very good accuracy and a very low amount of resources.

We validated our methodology with a set of different tests, which involved a controlled and European-wide testbeds, using synthetic and real traffic. The experimental results show that we can reduce the required resources considerably, with a low effect on the final accuracy of the system.

As lines left for further research, an interesting upgrade of the system would be to infer the real metrics of the network by comparing ingress and egress packet arrival times (Inter Packet Generation Time with Inter Packet Arrival Time).

11

Conclusions and future work

In this thesis we presented the main building blocks of an efficient and scalable end-to-end SLA Assessment infrastructure, based on network metric estimation. Each building block incrementally constructs a full-fledged and scalable solution, we start from the most basic per packet analysis and reporting, which we name *Network Parameter Acquisition System* (NPAS), to the efficient SLA violation estimation accomplished by a smart Inter Packet Arrival Time (IPAT) analysis. Passing by several levels of traffic sampling, which deliver a very good trade-off between resources and accuracy.

The main optimisations presented in this thesis are focused on the intra-domain environment, with some work in inter-domain, that opens several new lines for further research.

The incremental set of enhancements starts with, what we call, Time Classification, where we report the packet information in time intervals, using this technique it is possible to exploit simple compression techniques that give a very good reduction in the used resources. In fact the reduction we experimentally obtained in the worst case is $\sim 27\%$ without any loss in the accuracy of the solution.

Even with the very good optimisation of Time Classification, in order to deploy a competitive and scalable solution we need to further reduce the required resources in terms of bandwidth. Therefore we designed a Static Traffic Sampling solution, based on *hash sampling* that permits a customisable reduction on the required resources, with the trade-off of losing accuracy on the reporting. Using static sampling permits to accurately estimate the One-Way Delays even with very low sampling rates. On the other hand, the estimation of the discrete Packet Losses is much more complicated, having very poor accuracy in this regard.

We partially solve the packet loss estimation accuracy problem by using an Adaptive Traffic Sampling solution. Which, instead of statically applying a preconfigured sampling rate, goes one step further and permits custom allocation of the available resources to the flows with more requirements, the resource sharing is monitored by estimating the status of the metrics, as a proof of concept we used IPDVs and PLR. With this new approach we obtain: *i)* more control over the used resources due to control traffic. *ii)* much more accuracy estimating the packet losses by focusing on the troubled flows or segments of the network, achieving a total reduction in terms of resources of $\sim 96\%$ compared with the per packet reporting case, still with reasonable accuracy in OWDs.

As a complementary feature, we designed a user level Quality of Experience assessment for NPAS, namely we deliver VoIP assessment. The goal of this enhancement is to use the already present framework of SLA assessment for more complex tasks. The insertion of such QoE functionalities is not straight-forward since we have to enhance both the ME and the PE, the first needs to detect the used codec and the second needs to infer the quality of the flow(s) using the E-Model. In the way of giving such functionalities to NPAS we enhanced the already present MOS by porting its features to the more used packet switched networks.

Finally, the last part of the thesis discussed a disruptive approach to SLA assessment. This approach, which is the final improvement over NPAS, focuses on the decoupling of network metrics and SLA assessment. To do this we use information delivered by the IPATs and by using a simple methodology based on distance computation between distributions. We present a reliable, scalable, and very efficient mechanism to perform the SLA assessment. During the design of this technique we used the well-known Hausdorff Distance, which we improved largely, by using the knowledge of the specific problem at hand. The best feature of this new approach is that it can be easily implemented on the edge nodes of the network, where we minimise the need of a dedicated collection point. Moreover, the computational complexity of the solution is lower than the other alternatives, since the packet matching present on previous optimisations, for the metric computation is drastically reduced because it is inferred from the IPAT information.

Future Work

Our main lines for further research in this area can be separated on the following fronts:

- The first set of tasks left for future work are related to the traffic collection. It is broadly known that software capture interfaces are not reliable and fast enough in order to perform full link collection. The problem is aggravated if the collected data needs to be analysed in real-time. Therefore it is very interesting to assess the collection and analysis strains and how that affects to the SLA assessment accuracy. It is our belief, that even with some errors caused by the lack of collection power, with the proper correcting algorithms and knowledge about the effects it is possible to infer and react on time over these problems.

  Moreover, even with the imposed limitations, having deterministic boundaries over the collection and the accuracy in the analysis of the traffic, we still can use cheap software based collection platforms in several environments (e.g. small networks, or to monitor specific applications on the edges of the network).

- The second research topic left out of this study is related to the inter-domain analysis. In this work we have outlined possible alternatives for inter-domain SLA assessment but they lack the proper validation and detailed studies about the scalability, feasibility and accuracy of the presented solutions. Related to this, more work is required on techniques to aggregate different metrics. Aggregating metrics such as IPDV or PLR is not as straight-forward as could be aggregating simple OWDs.

  Also in inter-domain, another open issue are the security concerns of publishing internal status information about the network, even if only publishing metric information it can violate internal policies of the service provider, transit or access network.

- Third, with the huge increase of new Peer-to-Peer (P2P) platforms emerging in current networks, and with the new services appearing over them (e.g., IPTV, or P2PTV). It is becoming more and more relevant to develop new QoS metrics and new mechanisms for QoS detection, aware of such new paradigm.

  This is a challenging problem since in P2P networks a flow is no longer identified by the typical 5-tuple, now you can receive pieces of information from any peer on the overlay. Complicating considerably the QoS assessment. Moreover, if we add that the delivered content usually is video streaming, we end up with P2P aware Quality of Experience assessment.

- And the fourth topic left for further research is to complete the initial decoupling of the network metrics and the SLA assessment as we presented on the last part of the thesis. Currently it is still necessary to use monitoring points, and therefore we need to collect and analyse traffic in a distributed manner. Removing this constraint would pose definely a very good advance in efficient SLA assessment.

12

List of Publications

Here we present the list of publications in cronological order and separated by categories: Journals and book Chapters, Conferences, IST Project deliverables and Research Reports.

Journals and book Chapters

1. *Serral-Gracià, R.*, and Gil, Marisa: **A Linux Networking Study**, *In Operating System Review, Volume 38 Number 3 (SIGOPS ACM)*, July 2004.

2. Masip-Bruin, X., Yannuzzi, M., *Serral-Gracià, R.*, et al.:, **The EuQoS System: A Solution for QoS Routing in Heterogeneous Networks**., *IEEE Commun. Mag 45(2)* 96–103, 2007.

3. *Serral-Gracià, R.*, Domingo-Pascual, J., Bęben, A., and Owezarski, P.: *Chapter 2 - QoS Measurements in IP-based Networks* in **End-to-End Quality of Service Over Heterogeneous Networks**, *Springer, Eds: Braun, Torsten, and Staub, Thomas*, Aug 2008.

Conferences

1. Cabellos-Aparicio, A., *Serral-Gracià, R.*, Jakab, L., and Domingo-Pascual, J.: **Measurement Based Analysis of the Handover in a WLAN MIPv6 Scenario**, *Passive and Active Measurements (PAM)*, LNCS 3431, 207–218, 2005.

2. *Serral-Gracià, R.*, Cabellos-Aparicio, A., Julian-Bertomeu, H., and Domingo-Pascual, J.: **Active measurement tool for the EuQoS project**, *MOME 3rd International Work-*

shop on Internet Performance, Simulation, Monitoring and Measurements (IPS-MoMe), 2005.

3. Jakab, L., *Serral-Gracià, R.*, and Domingo-Pascual, J.: **A Study of Packet Losses in the EuQoS Network**, *MOME 4rd International Workshop on Internet Performance, Simulation, Monitoring and Measurements. IPS-MoMe*, 2006.

4. *Serral-Gracià, R.*, Jakab, L., and Domingo-Pascual, J.: **Out of Order Packets Analysis on a Real Network Environment**, *2nd Conference on Next Generation Internet Design and Engineering*, 2006.

5. *Serral-Gracià, R.*, Jakab, L., and Domingo-Pascual, J.: **Measurement Based Call Quality Reporting**, *2nd Workshop on Network Measurements in 32nd IEEE Conference on Local Computer Networks* 997–1004, 2007.

6. *Serral-Gracià, R.*, Barlet-Ros, P., and Domingo-Pascual, J.: **Coping with Distributed Monitoring of QoS-enabled Heterogeneous Networks**, *4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks (QoSIP - IT-NEWS*, 142–147, 2008.

7. *Serral-Gracià, R.*, Barlet-Ros, P., and Domingo-Pascual, J.: **Distributed Sampling for On-line QoS Reporting**, *Local and Metropolitan Area Network (LANMAN)*, Sep 2008.

8. *Serral-Gracià, R.*, Cabellos-Aparicio, A., and Domingo-Pascual, J.: **Network performance assessment using adaptive traffic sampling**, *IFIP Networking* LNCS 4982, 252–263, May 2008.

9. *Serral-Gracià, R.*, Cabellos-Aparicio, A., and Domingo-Pascual, J.: **Packet loss estimation using distributed adaptive sampling**, *End-to-End Monitoring (E2EMon)*, Apr 2008.

10. *Serral-Gracià, R.*, Labit, Y., Domingo-Pascual, J. and Owezarski, P.: **Towards efficient SLA Assessment**, *Submitted to Infocom*, 2009.

IST Project Deliverables

1. *LONG-Deliverable (4.4), Conclusions and Guidelines from Experiments*, Alberto García-Martínez (UC3M), Juan Francisco Rodriguez Hervella (UC3M), María José Perea (UPM), Javier Sedano (UPM), Juan Antonio Fernández (UPM), António Amaral (PTIN), Carlos Parada (PTIN), Jacinto Vieira (PTIN), Josep Mangues Bafalluy (UPC), Jordi Domingo-Pascual (UPC), René Serral-Gracia (UPM), Alberto Cabellos-Aparicio (UPC), Ignacio Grande (TID), Carlos Ralli (TID), Ruth Vázquez (TID), Mario Filipe (UEV), Miguel Ramos (UEV), 2002.

2. *LONG-Deliverable (2.4), Network Design and Deployment*, Jacinto Vieira (PTIN), Francisco Fontes (PTIN), Alberto García (UC3M), María José Perea (UPM), Alberto López (UPM), Javier Sedano (UPM), Ruth Vázquez (TID), Cristina Peña (TID), Ignacio Grande (TID), Carlos Ralli (TID), Mário Filipe (UEV), Miguel Ramos (UEV), Josep Mangues Bafalluy (UPC), Alberto Cabellos Aparicio (UPC), René Serral Gracia (UPC), Jordi Domingo Pascual (UPC), 2002.

3. *EuQoS Deliverable WP5 2.1.1 - 004503/WuT/DS/D2.1.1/A1, Definition of monitoring equipment and software and location points*, Wojciech Burakowski (WUT), Philippe Owezarski, Nicolas Larrieu (LAAS), Gerardo García, María Luisa García (TID), René Serral-Gracià, Jordi Domingo, Xavi Masip (UPC), Marek Dabrowski, Andrzej Beben, Piotr Pyda, Damian Duda, Halina Tarasiuk (WUT), Yannick Lizzi (Silogic), Enrico Angori, Giuseppe Martufi (Datamat), Regis Frechin (FTRD), Maxwel Carmo, Jorge Sá Silva, Paulo Simoes (UoC), 2005.

4. *EuQoS Deliverable WP5 5.1.1 - 004503/FTRD/DS/D5.1.1/A1, Technical requirements for the trial, tasks and scheduling*, Régis Frechin (FTRD), Pascal Le Guern (FTRD), Simões Paulo, Fernando Boavida, Jorge Sa Silva (UoC), Marc Brogle, Dragan Milic (UoB), Stéphane Statiotis (FTRD), Zbigniew Kopertowski (PTRD), Pablo Vaquero (Soluziona), Francisco Javier Ramón Salguero (TID), Sathya Rao (Telscom), Jordi Domingo-Pascual, René Serral-Gracià, Xavi Masip (UPC), Philippe Owezarski, Florin Racaru, Guillaume Auriol, Nicolas Larrieu (LAAS-CNRS), Laurent Dairaine, Mathieu Gineste, Patrick Sénac, Hervé Thalmensy (ENSICA-LAAS), Michal Obuchowicz, Pawan Cabell, Robert Parzydlo (PTC), Isabel Borges (PTIN), Abraham Gebrehiwot (UoPisa/CPR), Marek Dabrowski, Jaroslaw Sliwinski, Andrzej Beben, Piotr Pyda (WUT), 2005.

5. *EuQoS Deliverable WP5 5.1.2 - 004503/FTRD/DS/D5.1.2/A1, Connectivity and performance tests report for local and pan-European (across GEANT) testbed design for the Trial*, Régis Frechin (FTRD), Pascal Le Guern (FTRD), Francisco Javier Ramón Salguero, Pedro A. Aranda Gutiérrez (TID), Philippe Owezarski, Florin Racaru, Guillaume Auriol, Nicolas Larrieu (LAAS), René Serral-Gracià, Jordi Domingo-Pascual, Carles Kishimoto, José Núñez (UPC), Marek Dabrowski, Jaroslaw Sliwinski (WUT) Marc Brogle, Dragan Milic (UoB) , Zbigniew Kopertowski (PTRD), Luís Cordeiro (UoC), Abraham Gebrehiwot, Marco Sommani (UoPisa), 2005.

6. *EuQoS Deliverable WP5 5.1.3 - 004503/FTRD/DS/D5.1.3/A1, First individual based EuQoS System test report*, Marek Dabrowski (WUT); Dragan Milic, Marc Brogle (UoB), Jesus Bravo Alvarez (TID), Hugo Daniel Manaia (PTIN), Philippe Owezarski, Florin Racaru (LAAS), René Serral-Gracià (UPC), Antonio Pinizzotto (UoP), Maxweel Carmo (PTIN), Ciszkowski Tomasz (PTRD), Krystof Bonarski (PTC), 2006.

7. *EuQoS Deliverable WP5 5.1.4 - 004503/FTRD/DS/D5.1.4/A1, Testbed integration test plan*, Olivier Dugeon (FTRD), Francisco Javier Ramón Salguero, Pedro A. Aranda Gutiérrez (TID), Philippe Owezarski, Florin Racaru, Guillaume Auriol, Nicolas Larrieu (LAAS), René Serral-Gracià, Jordi Domingo-Pascual, Pau Capella (UPC), Marek Dabrowski, Jaroslaw Sliwinski (WUT) Marc Brogle, Dragan Milic (UoB), Zbigniew Kopertowski (PTRD), Luís Cordeiro (UoC), Abraham Gebrehiwot, Marco Sommani (UoPisa) Michal Obuchowicz, Adam Flizikowski, Krzysztof Bronarski, Pawel Caban , Slawomir Tkacz (PTC), Rubén Romero San Martin (Soluziona), 2006.

Research Reports

1. *Carlos Veciana Nogués, Alberto Cabellos Aparicio, René Serral Gracià, Jordi Domingo Pascual, Josep Solé Pareta*, **UPC-DAC-2002-07, Experimentos IPv6. Túnel manual sobre IPv4**, 2002.

2. *René Serral Gracià, Carlos Veciana Nogués, Alberto Cabellos Aparicio, Jordi Domingo Pascual, Josep Solé Pareta*, **UPC-DAC-2002-08, Pruebas distribuidas SABA2 UPC-UPM**, 2002.

3. *René Serral Gracià, Josep Mangues Bafalull, Jordi Domingo Pascual*, **UPC-DAC-2003-26, Differentiated Services Tests**, 2003.

4. *Adam Flizikowski, Jordi Domingo-Pascual, Josep Mangues-Bafalluy, Alberto Cabellos, René Serral, Carles Kishimoto*, **UPC-DAC-2002-46, Multicast IPv6 Tests on LONG network**, 2002.

5. *Kenan Cenanovic, René Serral, Jordi Domingo, Thomas Lindh*, **UPC-DAC-2004-13, Implementation and testing of the One-Way Active Measurement Protocol**, 2004.

6. *Lorand Jakab, Alberto Cabellos, René Serral, Jordi Domingo*, **UPC-DAC-2004-25, Software Tool for Time Duration Measurements of Handovers in IPv6 Wireless Networks**, 2004.

7. *René Serral-Gracià, Jordi Domingo-Pascual*, **UPC-DAC-RR-2006-24, Performance Evaluation of the EuQoS testbed**, 2006.

8. *René Serral, Loránd Jakab, Marcelo Yannuzzi, Xavier Masip, Jordi Domingo, Jarek Sliwinski, Andrzej Beben, Philipe Owezarski, María Ángeles Callejo, José Enríquez Gabeiras*, **UPC-DAC-RR-2006-40, Distributed Monitoring and Measurement System for Heterogeneous Networks**, 2006.

References

[1] IP Performance Metrics (IPPM) *http://www.ietf.org/html.charters/ippm-charter.htm*, Jan. 1990. 26

[2] EG ETSI 202 057-4: Speech Processing, Transmission and Quality Aspects (STQ); User related QoS parameter definitions and measurements; Part 4: Internet access. *Transmission and Quality Aspects (STQ)* (May 2008). 3

[3] ABARBANEL, B., AND VENKATACHALAM, S. BGP-4 support for Traffic Engineering. Draft 1, June 2000. 21

[4] AGARWAL, D., GONZLEZ, J., JIN, G., AND TIERNEY, B. An infrastructure for passive network monitoring of application data streams. In *Passive and Active Measurement Workshop (PAM)* (April 2003). 51

[5] ALMES, G., KALIDINDI, S., AND ZEKAUSKAS, M. A One-way Delay Metric for IPPM. RFC 2679, Sept. 1999. 28, 69, 128

[6] ALMES, G., KALIDINDI, S., AND ZEKAUSKAS, M. A One-way Packet Loss Metric for IPPM. RFC 2680, Sept. 1999. 28, 69, 128

[7] ALMES, G., KALIDINDI, S., AND ZEKAUSKAS, M. A Round-trip Delay Metric for IPPM. RFC 2681, Sept. 1999. 29

[8] ALMQUIST, P. Type of Service in the Internet Protocol Suite. RFC 1349, July 1992. 14

[9] Active Measurement Project (AMP) – http://amp.nlanr.net/. 45

[10] ANTONIADES, D., POLYCHRONAKIS, M., AND ET. AL. Appmon: An application for accurate per application network traffic characterization. In *BroadBand Europe* (2006). 59

[11] ATALLAH, M. Linear time algorithm for the Hausdorff distance between convex polygons. *INFO. PROC. LETT. 17*, 4 (1983), 207–209. 148

[12] BARFORD, P., AND SOMMERS, J. Comparing probe -and router- based methods for measuring packet loss. *IEEE Internet Computing Special Issue on Measuring the Internet* (2004). 50

[13] BATALLA, J. M. Calculating end-to-end QoS parameters from QoS objectives in Autonomous Systems. 12th Polish Teletraffic Symposium PSRT2005, 2005. 76

[14] BAUDRIER, E., GIRARD, N., AND OGIER, J. A non-symmetrical method of image local-difference comparison for ancient impressions dating. *Workshop on Graphics Recognition (GREC)* (2007). 147

[15] BLAKE, S., BLACK, D., CARLSON, M., DAVIES, E., WANG, Z., AND WEISS, W. An Architecture for Differentiated Services. RFC 2475, Dec. 1998. 10

[16] BOOTE, J. W., BOYD, E. L., DURAND, J., HANEMANN, A., KUDARIMOTI, L., APACZ, R. L., SIMAR, N., AND TROCHA, S. Towards multi-domain monitoring for the european research networks. *COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY* (2005). 50

[17] BORELLA, M., SWIDER, D., ULUDAG, S., AND BREWSTER, G. Internet Packet Loss: Measurement and Implications for End-to-End QoS. In *International Conference on Parallel Processing* (1998). 32, 103

[18] BOUTREMANS, C., AND ET. AL. Impact of link failures on VoIP performance. In *International Workshop on Network and Operating System Support for Digital Audio and Video* (2002). 127

[19] BRADEN, R., CLARK, D., AND SHENKER, S. Integrated Services in the Internet Architecture: an Overview. RFC 1633, June 1994. 10

[20] BRADEN, R., AND ZHANG, L. Resource ReSerVation Protocol (RSVP) – Version 1 Message Processing Rules. RFC 2209, Sept. 1997. 11

[21] BRAZIO, J., TRAN-GIA, P., AKAR, N., BEBEN, A., BURAKOWSKI, W., FIEDLER, M., KARASAN, E., MENTH, M., OLIVIER, P., TUTSCHKU, K., AND WITTEVRONGEL, S. Analysis and design of advanced multiservice networks supporting mobility, multimedia and internetworking. COST Action 279 Final Report, 2006. 76

[22] CABELLOS-APARICIO, A., SERRAL-GRACIÀ, R., JAKAB, L., AND DOMINGO-PASCUAL, J. Measurement Based Analysis of the Handover in a WLAN MIPv6 Scenario. In *Passive and Active Measurements, LNCS 3431* (2005), pp. 207–218. 40, 41, 62

[23] Cooperative Association for Internet Data Analysis (CAIDA). `http://www.caida.org/`. 45

[24] CARTER, J. L., AND WEGMAN, M. N. Universal classes of hash functions. *Journal of Computer and System Sciences 18*, 2 (1979), 143–154. 94

[25] CHENG, Y., RAVINDRAN, V., LEON-GARCIA, A., AND CHEN, H. New Exploration of Packet-Pair Probing for Available Bandwidth Estimation and Traffic Characterization. *Communications, 2007. ICC'07. IEEE International Conference on* (2007), 588–594. 43

[26] CHOI, B.-Y., PARK, J., AND ZHANG, Z.-L. Adaptive Packet Sampling for Accurate and Scalable Flow Measurement. In *Proceedings of IEEE Globecom '04* (2004). 102

[27] CISCO SYSTEMS. Introduction to Cisco IOS NetFlow - A Technical Overview, 2007. 59

[28] COLE, R. G., AND ROSENBLUTH, J. H. Voice over IP performance monitoring. In *ACM SIGCOMM Computer Communication Review, Volume 31, Issue 2* (2001). 126

[29] COZZANI, I., AND GIORDANO, S. Traffic sampling methods for end-to-end QoS evaluation in large heterogeneous networks. In *Computer Networks and ISDN Systems 30* (1998), pp. 1697–1706. 92

[30] CROVELLA, M., AND KRISHNAMURTHY, B. *Internet Measurement. Infrastructure, Traffic, and Applications*. John Wiley & Sons, Ltd, 2006. 57, 59

[31] DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION. INTERNET CONTROL MESSAGE PROTOCOL. RFC 792, July 1981. 42

[32] DARPA INTERNET PROGRAM, PROTOCOL SPECIFICATION. INTERNET PROTOCOL. RFC 791, July 1981. 14

[33] DAVIE, B., CHARNY, A., BENNETT, J., AND ET. AL. An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246, Mar. 2002. 85

REFERENCES

[34] DEMICHELIS, C., AND CHIMENTO, P. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393, Nov. 2002. 30, 69, 118, 128

[35] DIMES (Distributed Internet MEasurements & Simulations). http://www.netdimes.org/. 45

[36] DOVROLIS, C., AND JAIN, M. Pathload: A measurement tool for end-to-end available bandwidth. *Proceedings of the 3rd Passive and Active Measurements Workshop* (2002). 42

[37] DUFFELD, N. G. Sampling for Passive Internet Measurement: A Review. *Statistical Science 19*, 3 (2004), 472–498. 52, 56, 91, 102

[38] DUFFIELD, N. G., AND GROSSGLAUSER, M. Trajectory sampling for direct traffic observation. *IEEE/ACM Trans. Netw. 9*, 3 (2001), 280–292. 50, 51, 60, 92, 98, 99

[39] DURHAM, D., BOYLE, J., COHEN, R., HERZOG, S., RAJAN, R., AND SASTRY, A. The COPS (Common Open Policy Service) Protocol. RFC 2748, June 2000. 20

[40] EIDSON, J., AND LEE, K. IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems. *Sensors for Industry Conference, 2002. 2nd ISA/IEEE* (2002), 98–105. 55

[41] EKELIN, S., NILSSON, M., HARTIKAINEN, E., JOHNSSON, A., MÅNGS, J., MELANDER, B., AND BJÖRKMAN, M. Real-time measurement of end-to-end available bandwidth using Kalman filtering. *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium* (2006). 43

[42] Endace measurement systems - http://www.endace.com. 47, 63, 67

[43] Evergrow Traffic Observatory Measurement InfrastruCture. http://www.etomic.org/. 44

[44] ETTINGER, E., MCFEE, B., AND FREUND, Y. Pinpoint: Identifying Packet Loss Culprits Using Adaptive Sampling. Tech. rep., UCSD, 2007. 103

[45] IST-EuQoS – End-to-end Quality of Service support over heterogeneous networks - http://www.euqos.eu/, Sept. 2004. 22, 62, 77, 86, 162

[46] Ever-growing global scale-free networks, their provisioning, repair and unique functions. 45

[47] FEAMSTER, N., BORKENHAGEN, J., AND REXFORD, J. Guidelines for interdomain traffic engineering. *SIGCOMM Comput. Commun. Rev. 33*, 5 (2003), 19–30. 25

[48] FLOYD, S., AND JACOBSON, V. *Random Early Detection gateways for Congestion Avoidance*, vol. V. 1 N. 4. IEEE/ACM Transactions on Networking, Aug. 1993. 13

[49] GÉANT – http://www.geant.net. 75

[50] GEORGATOS, F., GRUBER, F., KARRENBERG, D., SANTCROOS, M., SANJ, A., UIJTERWAAL, H., AND WILHEM, R. Providing Active Measurements as a Regular Service for ISP's. In *Proceedings of Passive and Active Measurement* (2001). 45

[51] GROSSMAN, D. New Terminology and Clarifications for Diffserv. RFC 3260, Apr. 2002. 49

[52] HANEMANN, A., BOOTE, J. W., AND ET. AL. PerfSONAR: A Service Oriented Architecture for Multi-Domain Network Monitoring. In *Third International Conference on Service Oriented Computing, LNCS 3826, ACM SIGsoft and SIGweb* (2005), pp. 241–254. 49, 50

[53] HARRINGTON, D., PRESUHN, R., AND WIJNEN, B. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411, Dec. 2002. 58

[54] HEINANEN, J., BAKER, F., WEISS, W., AND WROCLAWSKI, J. Assured Forwarding PHB Group. RFC 2597, June 1999. 17

[55] HU, N., AND STEENKISTE, P. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications 21*, 6 (2003), 879–894. 42

[56] IETF. Internet Engineering Task Force. 26

[57] [IST] INTERMON – Inter-domain QoS Research - http://www.ist-intermon.org, Sept. 2003. 73, 77

[58] ITU INC. International Telecommunication Union: *http://www.itu.int*. 26

[59] ITU-T RECOMMENDATION G.107. The E-model, a computational model for use in transmission planning, 03/2005. 32, 36, 39, 100, 103, 125, 127, 128, 132

[60] ITU-T RECOMMENDATION G.113. Transmission impairments due to speech processing, 02/2001. 126, 136

[61] ITU-T RECOMMENDATION P.800.1. Mean Opinion Score (MOS) terminology, 03/2003. 36, 37

[62] ITU-T RECOMMENDATION P.862. Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, 2001. 38

[63] ITU-T RECOMMENDATION P.910. Subjective video quality assessment methods for multimedia applications, 1999. 37

[64] ITU-T RECOMMENDATION Y.1540. Internet protocol data communication service - IP packet transfer and availability performance paramteters, 12/2002. 15, 33

[65] ITU-T RECOMMENDATION Y.1541. Network Performance Objectives for IP-based Services, Review Jan 2005. 49, 57, 58, 70, 84, 85, 118

[66] JACOBSON, V., NICHOLS, K., AND PODURI, K. An Expedited Forwarding PHB. RFC 2598, June 1999. 17

[67] JESORSKY, O., KIRCHBERG, K., FRISCHHOLZ, R., ET AL. Robust face detection using the hausdorff distance. *Proceedings of Audio and Video based Person Authentication* (2001), 90–95. 148

[68] JHA, S., AND HASSAN, M. *Engineering Internet QoS*. Artech House Telecommunications Library, 2002. 19

[69] JIANG, W., KOGUCHI, K., AND SCHULZRINNE, H. QoS Evaluation of VoIP End-points. In *Passive And Active Measurements Workshop* (2003). 135

[70] KLEINROCK, L. *Queueing systems. volume I, Theory*. New York: Wiley, 1974. 106

[71] KOODLI, R., AND RAVIKANTH, R. One-way Loss Pattern Sample Metrics. RFC 3357, Aug. 2002. 32, 103

[72] KRISHNAMURTHY, B., WILLS, C., AND ZHANG, Y. Preliminary Measurements on the Effect of Server Adaptation for Web Content Delivery. In *ACM SIGCOMM Internet Measurement Workshop* (2002). 42

[73] KULLBACK, S., AND LEIBLER, R. On information and sufficiency. *Annals of Mathematical Statistics 22*, 1 (1951), 79–86. 147, 148

[74] KUROSE, J. *Computer Networking:A Top-Down Approach Featuring the Internet*. Addison-Westley, June 2004. 9

[75] Linux Advanced Routing and Traffic Control – http://lartc.org/. 61, 133

[76] LI, F., SEDDIGH, N., NANDY, B., AND MATUTE, D. An Empirical Study of Today's Internet Traffic for Differentiated Services IP QoS. In *ISCC '00: Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000)* (Washington, DC, USA, 2000), IEEE Computer Society, p. 207. 68

[77] Lobster – Large-scale Monitoring of Broadband Internet Infrastructures. `http://www.ist-lobster.org`. 48

[78] LOGG, C., NAVRATIL, J., AND COTTRELL, L. Correlating Internet Performance Changes and Route Changes to Assist Trouble-Shooting from and End-User Perspective. In *Proceedings of Passive and Active Measurement* (2004). 43

[79] MAHDAVI, J., AND PAXSON, V. IPPM Metrics for Measuring Connectivity. RFC 2678, Sept. 1999. 27

[80] MANKIN, A., BAKER, F., BRADEN, B., BRADNER, S., O'DELL, M., ROMANOW, A., WEINRIB, A., AND ZHANG, L. Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement Some Guidelines on Deployment. RFC 2208, Sept. 1997. 11

[81] MASIP-BRUIN, X., YANNUZZI, M., SERRAL-GRACIÀ, R., DOMINGO-PASCUAL, J., ENRÍQUEZ-GABEIRAS, J., CALLEJO, M., DIAZ, M., RACARU, F., STEA, G., MINGOZZI, E., ET AL. The EuQoS System: A Solution for QoS Routing in Heterogeneous Networks. *IEEE Commun. Mag 45*, 2 (2007), 96–103. 20, 22

[82] MATHIS, M., AND ALLMAN, M. A Framework for Defining Empirical Bulk Transfer Capacity Metrics. RFC 3148, July 2001. 31

[83] MATRAY, P., SIMON, G., STÉGER, J., CSABAI, I., AND VATTAY, G. Large scale network tomography in practice: Queueing delay distribution inference in the etomic testbed. In *IEEE Infocom* (2006). 73, 75

[84] MCCANNE, S., AND JACOBSON, V. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *USENIX Winter* (1993). 46

[85] MELANDER, B., BJORKMAN, M., AND GUNNINGBERG, P. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. *Global Telecommunications Conference. IEEE GLOBECOM* (2000). 42

[86] [IST] MESCAL - Management of End-to-end Quality of Service Across the Internet at Large, Jan. 2004. 22

[87] MILLS, D. L. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305, 1992. 54

[88] MILOUCHEVA, I., GUTIERREZ, P., AND ET. AL. Intermon architecture for complex QoS analysis in inter-domain environment based on discovery of topology and traffic impact. In *Inter-domain Performance and Simulation Workshop, Budapest* (March 2004). 47, 49

[89] The netfilter/iptables project - http://www.netfilter.org/. 46

[90] NGUYEN, H. X., AND THIRAN, P. Active Measurement for Multiple Link Faliures Diagnosis in IP Networks. In *Passive and Active Measurement Conference* (2004). 41

[91] NICHOLS, K., BLAKE, S., BAKER, F., AND BLACK, D. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, Dec. 1998. 14, 15

[92] NICHOLS, K., AND CARPENTER, B. Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification. RFC 3086, Apr. 2001. 16

[93] OWEZARSKI, P., BERTHOU, P., LABIT, Y., AND GAUCHARD, D. LaasNetExp: a generic polymorphic platform for network emulation and experiments. *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities* (2008). 162

[94] PÀSZTOR, A., AND VEITCH, D. Active Probing using Packet Quartets. In *ACM SIGCOMM Internet Measurement Workshop* (2002). 32

[95] PAXSON, V. Strategies for sound internet measurement. In *Internet Measurements Conference* (2004). 132

[96] PAXSON, V., ALMES, G., MAHDAVI, J., AND MATHIS, M. Framework for IP Performance Metrics. RFC 2330, May 1998. 27

[97] perfSONAR - http://www.perfsonar.net. 2, 45, 50

[98] PIRATLA, N., JAYASUMANA, A., AND SMITH, H. Overcoming the effects of correlation in packet delay measurements using inter-packet gaps. *Networks (ICON). Proceedings. 12th IEEE International Conference on 1* (2004). 144, 146

[99] Passive Measurement and Analysis (PMA) – http://pma.nlanr.net/. 48

[100] PRASAD, R. S., MURRAY, M., DOVROLIS, C., AND CLAFFY, K. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network 17*, 6 (Nov-Dec 2003), 27–35. 40

[101] QUITTEK, J., ZSEBY, T., CLAISE, B., AND ZANDER, S. Requirements for IP Flow Information Export (IPFIX). RFC 3917, Oct. 2004. 56

[102] RIBEIRO, V., COATES, M., RIEDI, R., SARVOTHAM, S., HENDRICKS, B., AND BARANIUK, R. Multifractal Cross-Trac Estimation. *Proceedings of the ITC Specialist Seminar on IP Trac Measurement, Modelling and Management* (2000), 18–20. 42

[103] RIBEIRO, V., RIEDI, R., BARANIUK, R., NAVRATIL, J., AND COTTRELL, L. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. *time 2*, 3. 43

[104] ROSEN, E., TAPPAN, D., FEDORKOW, G., REKHTER, Y., FARINACCI, D., LI, T., AND CONTA, A. MPLS Label Stack Encoding. RFC 3032, Jan. 2001. 18

[105] ROSEN, E., VISWANATHAN, A., AND CALLON, R. Multiprotocol Label Switching Architecture. RFC 3031, Jan. 2001. 10, 18

[106] A scalable monitoring platform for the internet – http://www.ist-scampi.org/. 47

[107] SERRAL-GRACIÀ, R., BARLET-ROS, P., AND DOMINGO-PASCUAL, J. Coping with Distributed Monitoring of QoS-enabled Heterogeneous Networks. In *4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks* (2008), pp. 142–147. 49, 80, 89, 122, 153, 164, 176

[108] SERRAL-GRACIÀ, R., BARLET-ROS, P., AND DOMINGO-PASCUAL, J. Distributed sampling for on-line qos reporting. In *LANMAN - Pending presentation* (2008). 102

[109] SERRAL-GRACIÀ, R., CABELLOS-APARICIO, A., AND DOMINGO-PASCUAL, J. Network performance assessment using adaptive traffic sampling. *IFIP Networking LNCS 4982* (May 2008), 252–263. 123, 176

[110] SERRAL-GRACIÀ, R., CABELLOS-APARICIO, A., AND DOMINGO-PASCUAL, J. Packet loss estimation using distributed adaptive sampling. In *End-to-End Monitoring* (Apr 2008). 63, 103, 123, 143

[111] SERRAL-GRACIÀ, R., CABELLOS-APARICIO, A., JULIAN-BERTOMEU, H., AND DOMINGO-PASCUAL, J. Active measurement tool for the EuQoS project. In *MOME 3rd International Workshop on Internet Performance, Simulation, Monitoring and Measurements. IPS-MoMe* (2005). 62

REFERENCES

[112] SERRAL-GRACIÀ, R., AND GIL, M. A Linux Networking Study. In *In Operating System Review-Volume 38 Number 3 (SIGOPS ACM)* (July 2004). 41, 60

[113] SERRAL-GRACIÀ, R., JAKAB, L., AND DOMINGO-PASCUAL, J. Out of order packets analysis on a real network environment. In *2nd Conference on Next Generation Internet Design and Engineering* (2006). 63

[114] SERRAL-GRACIÀ, R., LORÁND JAKAB, AND DOMINGO-PASCUAL, J. A Study of Packet Losses in the EuQoS Network. In *MOME 4rd International Workshop on Internet Performance, Simulation, Monitoring and Measurements. IPS-MoMe* (2006). 63

[115] SERRAL-GRACIÀ, R., JAKAB, L., AND DOMINGO-PASCUAL, J. Measurement Based Call Quality Reporting. In *2nd Workshop on Network Measurements in 32nd IEEE Conference on Local Computer Networks* (2007), pp. 997–1004. 141

[116] SHALUNOV, S., AND TEITELBAUM, B. One-way Active Measurement Protocol (OWAMP) Requirements. RFC 3763, Apr. 2004. 43, 53

[117] Passive Measurement and real-time analysis – http://www.cba.upc.edu/smartxac. 48, 78

[118] SOMMERS, J., BARFORD, P., DUFFELD, N. G., AND RON, A. Improving Accuracy in End-to-end Packet Loss Measurement. In *Proceedings of ACM SIGCOMM* (2005). 50

[119] SOMMERS, J., BARFORD, P., DUFFELD, N. G., AND RON, A. Accurate and Efficient SLA Compliance Monitoring. In *Proceedings of ACM SIGCOMM* (2007), pp. 109–120. 49, 50, 103

[120] STRAUSS, J., KATABI, D., AND KAASHOEK, F. A measurement study of available bandwidth estimation tools. *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement* (2003), 39–44. 32, 43

[121] TAKAHASHI, A., YOSHINO, H., AND KITAWAKI, N. Perceptual QoS Assessment Technologies for VoIP. *IEEE/Comm. Mag 42*, 7 (2004), 28–34. 36

[122] [IST] Traffic Engineering for Quality of Service in the Internet, at Large Scale, May 2004. 22

[123] TOBAGI, A., MARKOPOULOU, A. P., AND KARAM, M. J. Assessing the quality of voice communcations over internet backbones, 2002. 9

[124] VECIANA-NOGUÉS, C., CABELLOS-APARICIO, A., DOMINGO-PASCUAL, J., AND SOLÉ-PARETA, J. Verifying IP Meters from Sampled Measurements. In *Kluwer Academic Publishers. Testing of Communicating Systems XIV. Application to Internet Technologies and Services. IFIP TC6/WG6.1. TestCom* (2002), pp. 39–54. 91, 92

[125] VEITCH, D., BABU, S., AND PÀSZTOR, A. Robust synchronization of software clocks across the internet. In *Internet Measurement Conference (IMC '04)* (2004). 55

[126] VIVANCO, D., AND JAYASUMANA, A. A Measurement-Based Modeling Approach for Network-Induced Packet Delay. *Local Computer Networks (LCN). 32nd IEEE Conference on* (2007), 175–182. 144, 146

[127] WILLIAMSON, C. Internet Traffic Measurement. *IEEE INTERNET COMPUTING* (2001). 26

[128] WROCLAWSKI, J. The Use of RSVP with IETF Integrated Services. RFC 2210, Sept. 1997. 11

[129] ZIVIANI, A., FDIDA, S., DE REZENDE, J., AND DUARTE, O. C. M. Toward a Measurement-Based Geographic Location Service. In *Proceedings of Passive and Active Measurement* (2004). 42

[130] ZSEBY, T. Deployment of Sampling Methods for SLA Validation with Non-Intrusive Measurements. *Proceedings of Passive and Active Measurement Workshop (PAM)* (2002), 25–26. 49, 51, 91

[131] ZSEBY, T. Stratification Strategies for Sampling-based Non-intrusive Measurements of One-way Delay. In *Passive and Active Measurements* (2003). 91

[132] ZSEBY, T. Comparison of Sampling Methods for Non-Intrusive SLA Validation. In *E2EMon* (2004). 91

[133] ZSEBY, T., ZANDER, S., AND CARLE, G. Evaluation of Building Blocks for Passive One-Way-Delay Measurements. In *Passive and Active Measurements Conference* (2001). 50, 51, 60, 68