

Faculdade de Engenharia da Universidade do Porto



Monitoring Multicast Traffic in Heterogeneous Networks

Filipe Miguel Monteiro da Silva e Sousa

Thesis submitted fulfilling the requirements for the Degree of

Master in Electrical and Computers Engineering

Major in Telecommunications

Supervisor: Manuel Alberto Pereira Ricardo (PhD)

July of 2008

A Dissertação intitulada

“Monitoring multicast traffic in heterogeneous networks”

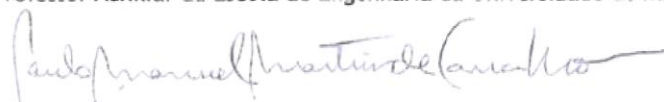
foi aprovada em provas realizadas 17/Julho/2008

o júri

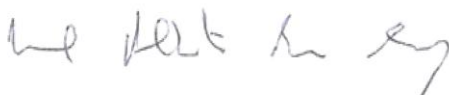
Presidente Professor Doutor José António Ruela Simões Fernandes
Professor Associado da Faculdade de Engenharia da Universidade do Porto



Professor Doutor Paulo Manuel Martins Carvalho
Professor Auxiliar da Escola de Engenharia da Universidade do Minho



Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado da Faculdade de Engenharia da Universidade do Porto



O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - Filipe Miguel Monteiro da Silva e Sousa



Faculdade de Engenharia da Universidade do Porto

Resumo

Num futuro próximo, diversos tipos de terminais móveis com uma panóplia de tecnologias sem fios integradas estarão ao dispor dos utilizadores. Será possível o estabelecimento de ligação com qualquer uma das tecnologias, de forma transparente, e permitindo a mobilidade do utilizador, sem qualquer quebra de serviço. Supletivamente, novos tipos de aplicações, como o *video streaming* de um para muitos, têm novos requisitos de tráfego de rede, como maiores larguras de banda e garantias de Qualidade de Serviço. O objectivo do projecto DAIDALOS é desenvolver, testar e demonstrar uma arquitectura baseada no protocolo de rede IPv6. Este é utilizado para fazer a integração de todas as tecnologias de rede com Qualidade de Serviço e mecanismos de segurança.

Neste contexto, os fornecedores de serviços de rede irão possuir um processo de controlo de admissão de sessões por forma a garantir a qualidade de serviço das aplicações. Esse processo aceitará as novas sessões conforme o estado das reservas já efectuadas. Por forma a permitir uma maior alocação de sessões do que a capacidade máxima da rede, novos algoritmos de controlo de admissão de sessões estão a surgir. Desta forma, ficará possibilitada uma utilização mais eficiente dos recursos da rede. Esses algoritmos requerem informação sobre a rede, nomeadamente sobre os recursos disponíveis e sobre os parâmetros de qualidade de serviço.

O objectivo desta tese é especificar, desenvolver e avaliar a arquitectura de um sistema de monitorização de rede que seja pouco intrusiva e transparente para os utilizadores, e que não afecte a qualidade de serviço das sessões em curso. Com este propósito, foram consideradas as duas técnicas de medidas existentes, medição passiva e medição activa. O modus operandi das medidas passivas caracteriza-se por calcular as métricas desejadas mediante a análise do tráfego actual. Esta técnica acarreta problemas de escalabilidade que podem ser colmatados com o recurso a técnicas de amostragem. Estas traduzem-se numa diminuição do esforço de processamento para o cálculo das métricas de QoS, associada a uma baixa taxa de erro. Por sua vez, as medidas activas também podem levantar alguns problemas devido à injeção de tráfego na rede que pode afectar as sessões em curso dos utilizadores. As medidas activas são utilizadas para testar o acréscimo de um utilizador a um grupo de *multicast*. O teste deverá ter um débito baixo mas suficientemente alto para emular o comportamento do utilizador. Nesta tese, é avaliado o nível de intrusão deste tipo de testes.

Abstract

In a near future, a user will have different types of mobile terminals with several wireless technologies integrated. A mobile terminal will be seamlessly connected to any technology, allowing the user's mobility and maintaining sessions' continuity. Applications such as video streaming from one-to-many have new traffic requirements, such as large bandwidths, and Quality of Service (QoS) guarantees. The DAIDALOS project objective is to develop, test and demonstrate an open architecture based on the network protocol IPv6. This protocol is used to integrate all the network technologies with QoS capabilities, and in a secure communication environment.

In the DAIDALOS framework, the network providers are developing a call admission control process to guarantee the quality of service for each application. This process will decide if it accepts a new session based on the current reservations. In order to maximize the number of sessions accepted, novel call admission control algorithms are being developed. These algorithms require network information, related to the resources available and QoS parameters.

The objective of this thesis is to specify, develop, and evaluate a network monitoring architecture used to monitor network resources in a non intrusive way, without affecting the QoS of the running sessions. To fulfill this objective, multicast admission control scenarios were studied and requirements were specified. From the scenarios, metrics and measurement techniques were defined.

In order to achieve our purpose, two existing measurement techniques: passive and active - were implemented and compared. The modus operandi of the passive measurements is to use the network traffic in order to calculate the required metrics; this technique is not scalable and, in order to overcome this problem, sampling methods were developed. The adoption of these sampling methods reduces the overall processing effort, with a small measured error. On the other hand, active measurements can also raise problems, since user sessions can be affected by injection of test traffic into the network. The test of a new user joining a multicast group demands an active measurement technique. The test should have low bit rate, but sufficient high to emulate the new user behavior. The intrusiveness of the test traffic is also evaluated in this thesis.

Acknowledgements

The author would like to thank a number of people for making this work possible. First and foremost, Dr. Manuel Ricardo for always believing in me and for invaluable advice and help.

I would also like to thank my colleagues at INESC Porto for their support, in particular Ricardo Morla for useful advice, but also Rui Campos, Gustavo Carneiro, Ricardo Duarte, and Carlos Pinho, for their valuable feedback. The INESC Porto institution in general also deserves my appreciation, for providing me with an excellent environment to complete this work, as well as the University of Porto.

Finally, I would like to thank my family and my fiancée for all the help I received throughout these last months.

The work presented in this report was partially funded by the EU project DAIDALOS (phase 1: IST-2002-506997, phase 2: IST-2005-026943).

The author

*“Computer Science is no more about computers
than astronomy is about telescopes.”*

E. W. Dijkstra

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem statement	2
1.3	Objectives and strategy	2
1.4	Thesis structure	3
2	State-of-the-art	5
2.1	IP version 6	5
2.1.1	IPv6 Header	6
2.1.2	IPv6 Addressing	7
2.2	IP Multicast	8
2.2.1	Multicast Listener Discovery Protocol	9
2.2.2	Protocol Independent Multicast	9
2.3	Quality of Service	10
2.3.1	Differentiated Services	10
2.3.2	Integrated Services	11
2.3.3	Quality of Service Parameters	12
2.3.3.1	One-way-delay	12
2.3.3.2	Delay Variation (Jitter)	13
2.3.3.3	Packet Loss Ratio	14
2.4	Network monitoring	14
2.4.1	Active measurement	15
2.4.2	Passive measurement	16
2.4.3	IPFIX architecture	17
2.4.4	Sampling schemes	19
2.5	Multicast monitoring tools	22
2.5.1	Multicast Reachability Monitor (MRM)	22
2.5.2	Multicast Beacon	23
2.5.3	Multicast Quality Monitor	24
2.6	Context for network monitoring - DAIDALOS Project	25
2.6.1	Network Architecture	25
2.6.2	Classes of Service	27
2.6.3	Monitoring System Integration	28
2.7	Summary	28

3	Network monitoring requirements and system specification	31
3.1	System requirements	31
3.1.1	Multicast QoS signaling scenario	31
3.1.2	Network measurements for call control admission	33
3.2	Architecture design	36
3.2.1	Measurement Platform	36
3.2.2	CMS architecture	37
3.2.3	NME architecture	43
3.3	Implementation of NME functions	44
3.3.1	ActiveMeasure	44
3.3.2	QoSMeasure	46
3.3.3	QoS Calculator	50
3.4	Integration	52
4	Measurements results analysis	55
4.1	Test Scenario	55
4.2	Results	56
4.3	Evaluation	60
5	Conclusions	65
5.1	Summary	65
5.2	Achievements	66
5.3	Future Work	66
A	Monitoring platforms evaluation	69
A.1	Platform Description	69
A.1.1	Open Internet Measurement Project (OpenIMP)	69
A.1.2	IP Probes	70
A.1.3	Monitoring Platform for Mobile Flows (MPMF)	70
A.1.4	NeTraMet	71
A.2	Evaluation	71
A.3	Licensing and availability	75
A.3.1	OpenIMP	75
A.3.2	IP Probes	75
A.3.3	MPMF	75
A.3.4	NeTraMet	76
A.4	Conclusions	76
B	DAIDALOS testbed	77
B.1	Accident and University scenario	77
	Index	81

List of Figures

1.1	Internet Growth.	1
2.1	IPv6 and IPv4 header.	6
2.2	Scenario with multicast and unicast flow.	8
2.3	DiffServ Node.	11
2.4	One-way-delay between two hosts.	12
2.5	Delay variation (Jitter).	13
2.6	Packet Loss between two hosts.	14
2.7	Active Measurement.	15
2.8	Passive Measurement.	16
2.9	Reference Architecture.	18
2.10	IPFIX Device.	18
2.11	IPFIX Template.	19
2.12	Sampling Schemes.	20
2.13	Multicast reachability monitor.	23
2.14	Multicast Beacon.	23
2.15	Multicast quality monitor centralized control.	24
2.16	Daidalos QoS network architecture.	26
3.1	QoS multicast signaling.	32
3.2	Multicast measurement scenario.	35
3.3	Message sequence chart for multicast measurements scenario.	35
3.4	OpenIMP architecture.	37
3.5	CMS - Architecture and interfaces.	38
3.6	DAIDALOS components interfacing with CMS.	38
3.7	Central Monitoring System (CMS) composite structure diagram.	39
3.8	Class Diagram of CMS SOAP API.	41
3.9	NME component diagram.	43
3.10	NME Composite Structure Diagram.	44
3.11	Graphical user interface for configuring the function QoSMeasure.	52
3.12	Graphical user interface for configuring the function ActiveMeasure.	53
4.1	Test Scenario.	55
4.2	Active measurement results	57
4.3	Passive measurement results - One-Way-Delay.	58
4.4	Passive measurement results - Jitter	59
4.5	Passive measurement results - Packet Loss	60
4.6	One-Way-Delay Error	61
4.7	Jitter Error	61

4.8	Packet loss sampling error	62
4.9	Metrics Computation Time.	62
B.1	DAIDALOS testbed	78

List of Tables

2.1	Sampling Schemes Overview.	20
2.2	QoS requirements for four classes.	27
3.1	Types of Data exchanged with the CMS.	39
3.2	SOAP measurement interface overview.	42
3.3	SOAP NME management interface overview.	42
3.4	NME interface overview.	44
A.1	Comparison between the monitoring platforms	72

Acronyms

4G Fourth-Generation Communications System	33
A4C Authentication, Authorization, Accounting, Auditing and Charging	25
AAA Authentication, Authorization and Accounting	31
AG Accounting Gateway	28
AN Access Network	26
ANQoSB Access Network Quality of Service Broker	27
AR Access Router	26
ARM Advanced Router Mechanisms	27
BGP Border Gateway Protocol	9
CAC Call Admission Control	65
CMS Central Monitoring System	xi
CNQoSB Core Network Quality of Service Broker	26
DAIDALOS Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services	24
DiffServ Differentiated Services	10
DSCP DiffServ Code Point	27
ER Edge Router	26
GUI Graphical User Interface	36
ICMPv6 Internet Control Message Protocol Version 6	9
IGMP Internet Group Management Protocol	9
IGMPv3 IGMP version 3	9
IntServ Integrated Services	10
IP Internet Protocol	6
IPFIX Internet Protocol Flow Information eXport	17
IPPM IP Performance Metrics	36
IPsec IP security	6
IPv4 Internet Protocol version 4	5
IPv6 Internet Protocol version 6	5
L3 Layer 3	33
MAC Medium Access Control	5

MIB Management Information Base	15
MLD Multicast Listener Discovery	9
MLDv1 MLD version 1	9
MLDv2 MLD version 2	9
MMSP MultiMedia Service Proxy	27
MT Mobile Terminal	26
NME Network Monitoring Entities.....	27
OID Object IDentifiers	15
OpenIMP Open Internet Measurement Project	36
PBNMS Policy Based Network Management System.....	27
PIM Protocol Independent Multicast.....	9
PIM-DM PIM Dense Mode	9
PIM-SM PIM Sparse Mode	9
PIM-SSM PIM Source Specific Multicast	10
PSAMP Packet Sampling.....	36
QoS Quality of Service	10
RP Rendezvous Point	9
RSVP Resource Reservation Protocol	11
RTP Real Time Protocol	14
SLA Service Level Agreement	16
SNMP Simple Network Management Protocol	15
SPP Service Provision Platform	26
SSM Source-Specific Multicast.....	10
TTL Time To Live.....	6
VoIP Voice over IP	5

Chapter 1

Introduction

1.1 Motivation

The networks and communications paradigms are changing because new communications scenarios and applications are arising. The voice communications are transported not only by circuit-switched but also by packet-switched networks. Moreover, applications like peer-to-peer and video streaming have been adopted world wide, increasing the network traffic and establishing new requirements. To overcome the resources starvation, network operators need to manage the resources, by adopting protocols or call admission control functions.

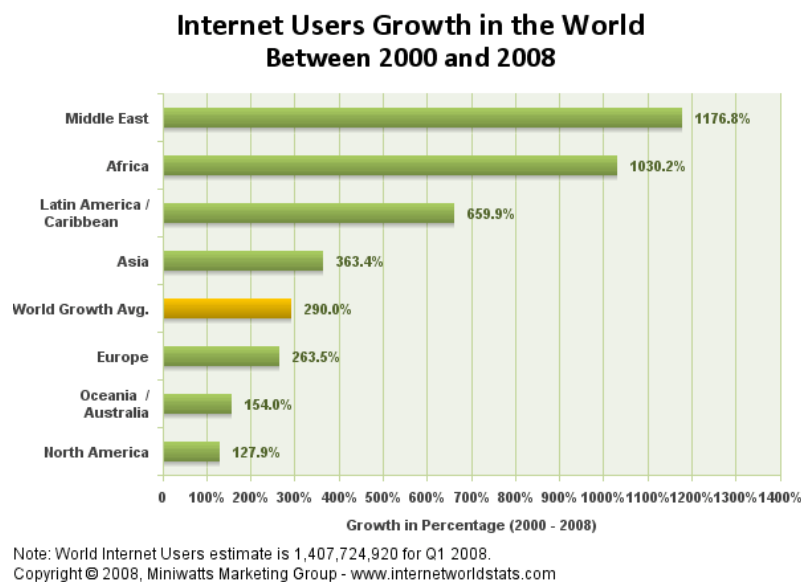


Figure 1.1: Internet Growth.

Multicast is a good solution to help solving the resources starvation; it reduces the amount of traffic in the network when streaming a video from one source to several receivers. The resources reserved for a flow may consider statistical multiplexing effects. The

real traffic profile of the flows may be unpredictable. In order to improve the efficiency of resources usage, and take advantage of the statistical multiplexing gains, we propose the use of a network monitoring system to monitor the available resources in the network. The monitoring results are then used for optimized admission decisions and to multiplex data streams.

1.2 Problem statement

The focus of this study is to use measurements to aid in the admission control of new multicast flows. When a client needs to join a multicast session, for instance a video streaming, the network operator should decide if it accepts this new session based on the current reservations. Since the profile of the flows, that are already transported over the network, is unpredictable, the network resources usage can be improved by admitting more flows than the maximum network capacity. This concept is defined as overbooking and may also benefit from the "statistical multiplexing gain", characteristics of the bursty (variable-bit rate) traffic. Using this technique, the network operator can allocate more sessions than those enabled by the real network capacity. The problem of overbooking is that it can cause network congestion and degrade the quality of all user sessions. A possible solution to overcome this problem consists in monitoring the state of network resources before accepting new sessions. In order to pursue this requirement, the monitoring system should perform tests aimed at evaluating the network resources available.

1.3 Objectives and strategy

The DAIDALOS project is the main driver for this work and most of the requirements are defined taking into consideration the DAIDALOS objectives. The focus of the project was to design, prototype and validate the necessary infrastructure and components for efficient distribution of services over diverse network technologies beyond 3G. The prototype is the result of the integration of complementary network technologies in order to provide pervasive and user-centered access to services.

In order to evaluate the distribution of services over the network, it is necessary to monitor these services and also the network resources. The objective of this work is to specify and develop a solution to monitor the network resources in a non intrusive and seamless way, without affecting the quality of service of the running sessions. One major requirement is that the intrusiveness of these tests should be minimum to avoid disturbing the running sessions.

The strategy adopted for this research work is to:

1. Define requirements for the network monitoring system;
2. Study the available measurements tools;

3. Specify a network monitoring architecture;
4. Develop software components;
5. Evaluate the solution.

The definition of requirements for the network monitoring system is important in order to specify the granularity of measurements, the necessary metrics, and the network measurement points. Scenarios for call admission control for multicast sessions with QoS shall also be studied.

Since the required measurements can be complex, the integration of available open source measurement tools should be evaluated. Moreover, the exchange of the measurement results should be researched, as well as the existence of libraries providing this functionality.

The specification of a network monitoring architecture based on the state-of-the-art and on the main forums is the next task. Moreover, it is necessary to identify which entities are going to interface with the network monitoring system and how this interface is going to be implemented. It is also important to characterize how measurement results are going to be computed and exported to other entities.

The next step is to develop specific measurement functions that will perform the functionality drawn during requirements specification. These functions will be responsible for collecting and executing network tests in order to produce the metrics specified in the early stages.

An evaluation of the specified architecture and developed measurement functions is performed. The idea is to evaluate if the objectives were fulfilled and the requirements met.

1.4 Thesis structure

The chapters are organized as follows. Chapter 2 provides an introduction to the basic mechanisms of IP multicast and QoS, including an analysis of potential problems. Next, the main network monitoring mechanisms and the reference architecture are described. At the end, the context of network monitoring and the summary of the adopted technologies, as well as their importance for the performed work, are presented.

Chapter 3 specifies the requirements and the design of the network monitoring system architecture. The implementation of the measurement functions and the impact on the architecture are also provided. The last section describes how the measurements functions were integrated into the network monitoring architecture.

The collected measurements are analysed on Chapter 4. First, test scenarios and performed tests are defined. From these, measurements were collected and analysed. An evaluation of the architecture and the implemented functions is provided taking into consideration the collected measurements.

Chapter 5 concludes the thesis work. It presents an overview of the studied technologies, advantages of deployed architecture, and an evaluation of the implemented solution. At the end, the major achievements are presented and the plans for future work drawn.

Chapter 2

State-of-the-art

The chapter presents the technologies and protocols mentioned in the thesis, and describes their importance for the performed work. Since IPv6, multicast and QoS are important for the work description, a small summary for each one is provided. After that, the major network monitoring topics and the reference monitoring architecture are introduced. Next, multicast network monitoring tools are presented, emphasising on the less intrusive techniques. At the end, a brief description of the project DAIDALOS architecture and QoS entities is provided in order to better define the context of problem.

2.1 IP version 6

Internet Protocol version 6 (IPv6) is a network layer protocol that ensures end to end (source to destination) packet delivery. This protocol is designed as the successor of the Internet Protocol version 4 (IPv4) [1] and it was designed to overcome the address exhaustion caused by the small size of IPv4 addresses and the increasing number of Internet nodes. The IPv6 protocol increases the network address size from 32 bits to 128 bits, or approximately 3.4×10^{38} addressable nodes, allowing a larger address space and more flexibility in the assignment of network addresses. The multicast routing scalability is improved by adding a new field, called scope, to multicast addresses. Moreover, it simplifies the auto-configuration of addresses using a stateless procedure. When the host connects to the network, it sends a router solicitation request, and the router replies with a router advertisement, containing the network prefix. The host receives the network prefix and composes the final network address using its network interface Medium Access Control (MAC) address. Another major modification is the header format simplification. By making some of the header fields optional, or even dropping them, the processing cost for handling an IPv6 packet on each router is reduced. Although the bandwidth cost is also limited, by dropping some header fields, the packet overhead is higher (the network address' size increased four times), especially when the payload size is small, like for Voice over IP (VoIP) applications. In these cases, a reduction of the overhead is

possible by adopting some header compression methods. **IPv6** also improves the support for extensions and options, making more efficient the packet forwarding and adding more flexibility for introducing new options in the future. The new extensions, such as authentication, data integrity and, optionally, data confidentiality, allow more secure and private communications that were optional in **IPv4**, by the adoption of IP security (**IPsec**). Mobility is also enhanced in **IPv6** because it supports stateless auto-configuration of Internet Protocol (**IP**) addresses and it avoids triangular routing, typical on **IPv4** networks, when using Mobile **IP**.

2.1.1 IPv6 Header

The **IPv6** packet is decomposed in two main parts: the header and the payload. The Figure 2.1 depicts the main differences between the **IPv4** and **IPv6** header [2]. As mentioned in Section 2.1, the **IPv6** header resulted from a simplification of the **IPv4** header. Several fields, that were available on the **IPv4** header are not present in the **IPv6** header, like identification, flags, fragment offset, header checksum, options and padding. The elimination of the header checksum from the **IPv6** header happened because the verification is already performed by the lower layers. This way, the processing cost is diminished, as it is no longer necessary to recompute the header checksum on each router (each time a packet is forwarded by the router, some fields, like the Time To Live (**TTL**), are changed and the checksum has to be recomputed).

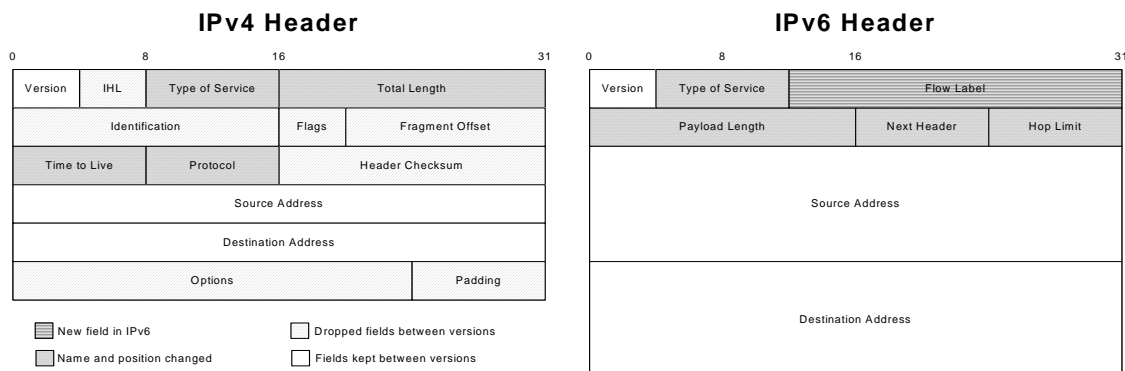


Figure 2.1: IPv6 and IPv4 header.

The first 40 octets (320 bits) of the packet contain the IPv6 header which is composed by the following fields:

Version - (4 bits) IP version, for IPv6 its 6.

Traffic class - (8-bits) packet priority delivery value. The packet is classified and prioritized according to this field.

Flow label - (20 bits) specifies special router handling from source to destination(s) for a sequence of packets.

Payload length - (16 bits) Payload length in bytes.

Next header - (8 bits) specifies the next encapsulated protocol. The protocol field of IPv4 is replaced by a Next Header field. This field usually specifies the transport layer protocol used by a packet's payload.

Hop limit - (8 bits) replaces the **TTL** field of IPv4, but both fields specify the hop limit for a packet.

Source address - (128 bits) source address of the node.

Destination address - (128 bits) destination address of the node.

The main difference between both headers is the size. When the options field in **IPv4** is not used the header size is 20 octets but in the **IPv6** header is 40 octets. The **IPv6** header overhead is given by formula:

$$Overhead = \frac{IP\ Header\ Size}{IP\ Header\ Size + Payload\ Size}$$

For a small payload size, the overhead will be very high, which is the typical scenario for VoIP applications. To overcome this problem, header compression methods like ROHC, presented in [3], can be applied.

2.1.2 IPv6 Addressing

In [4], it is specified the addressing architecture of the **IPv6** protocol and addresses formats. **IPv6** addresses are 128 bit identifiers for an interface, or for a set of interfaces, and are divided into three types of addresses:

- Unicast;
- Anycast;
- Multicast.

The unicast address is used to uniquely identify a single interface. The communication between two unicast addresses is a one-to-one communication. A packet with a unicast address in the destination address field is delivered to the interface identified by that address. Unicast **IPv6** addresses can be decomposed in the following types:

- Global unicast addresses;
- Link-local addresses;
- Site-local addresses;
- Unique local **IPv6** unicast addresses;

- Special addresses.

The global unicast addresses are used to communicate over the Internet, while the Link-local and Site-local have a local scope and are limited to link or site communications, respectively.

The anycast address identifies a set of interfaces. The communication between a unicast addresses and a anycast address is considered a one-to-one communication. When a packet is sent with an anycast address in the destination address field, it is delivered to only one of the interfaces that belongs to that anycast address. The adopted metric to select the interface from the group is the shortest routing distance.

A Multicast address is an identifier for a set of interfaces that typically belong to different nodes. The communication between a unicast address and a multicast address is considered a one-to-many communication. This means that multicast is used for group communications. A packet with a multicast address in the destination address field is delivered to all the interfaces that belong to this group (i.e. are identified by that address).

IPv6 multicast addresses are defined in [5], and begin with the prefix FF00::/8, and their second octet identifies the addresses' scope, i.e. the range over which the multicast address is propagated. Commonly, used scopes include link-local (0x2), site-local (0x5) and global (0xE).

2.2 IP Multicast

The IP multicast is a routing technology that was developed to conserve the links bandwidth by reducing the traffic when it is necessary to stream information from one source to thousands of receivers. The alternative to this scenario is to establish n connections (where n is the number of receivers) between the information source and the receivers, as it is depicted in Figure 2.2. In some links, the information is duplicated, exhausting the link bandwidth.

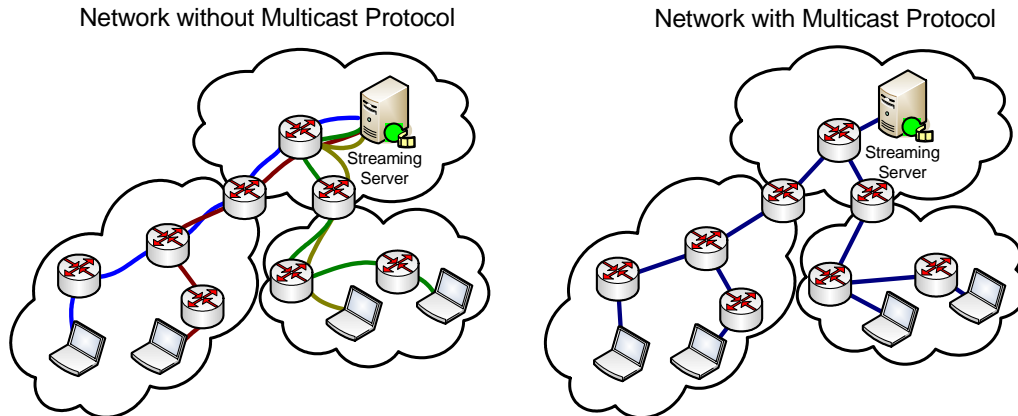


Figure 2.2: Scenario with multicast and unicast flow.

The solution is to replicate the information in some routers resulting in a more efficient delivery of data to multiple receivers. This benefits low-bandwidth applications if we considered thousands of receivers. In the case of high-bandwidth applications, like videoconferencing, corporate communications, and distance learning, the advantage is even bigger. The multicast technology basically allows a group of receivers to subscribe an interesting data stream. The hosts are not limited to any physical or geographical boundaries. The hosts subscriptions in the local IPv6 networks are managed by the Multicast Listener Discovery (MLD) protocol [6]. The routing information is then disseminated across the domain using protocols such as Protocol Independent Multicast (PIM) [7].

2.2.1 Multicast Listener Discovery Protocol

The MLD protocol is implemented by routers, and by end hosts, to manage multicast groups on IPv6 local area networks. This protocol is similar to the Internet Group Management Protocol (IGMP) for IPv4 access networks. The end hosts request to join a multicast group to the access router (i.e., the router directly attached to the host link). If the end host is accepted to join the multicast group, it becomes a multicast listener, and it receives all the packets intended to the multicast group. Instead of creating a new protocol, MLD is based on Internet Control Message Protocol Version 6 (ICMPv6) messages. The first version was MLD version 1 (MLDv1) and it is specified in [8]. The translation of the IGMP version 3 (IGMPv3) [9] protocol for IPv6 semantics gave raise to the next version, MLD version 2 (MLDv2). The latest version allows a multicast listener to receive packets from a specific multicast source address. Both versions of the MLD protocol are interoperable.

2.2.2 Protocol Independent Multicast

PIM is used to provide multicast routing information for the routers in a domain. It enables the distribution of data one-to-many and many-to-many. This protocol relies on other routing protocols, like Border Gateway Protocol (BGP), to discover the network topology. Depending on the number of receivers, and how they are distributed on a domain, different variants of this protocol can be adopted. There are four variants of PIM:

PIM Sparse Mode (PIM-SM) - suited for sparsely populated domains, it builds unidirectional shared trees rooted at a Rendezvous Point (RP). The RP is a router that is used as the root of the shared tree for a multicast group. The join messages from the receivers are sent towards this router and senders transmit the data to the same point [7].

PIM Dense Mode (PIM-DM) - suited for densely distributed receivers and the shared tree is build using a flooding mechanism. The source starts sending multicast datagrams for all the areas of the network. After some time if some areas of the network

do not have any subscriptions (i.e. join messages), the protocol will prune off the forwarding branch by instantiating a prune state [10]. The main differences between PIM-SM and PIM-DM are quite obvious, PIM-DM does not have a RP and only explicitly triggered prunes and grafts are transmitted.

Bidirectional PIM - it is a variant of PIM-SM that builds shared bi-directional trees connecting multicast sources and receivers, more information is available in [11].

PIM Source Specific Multicast (PIM-SSM) - it is a specific application of PIM-SM using a Source-Specific Multicast (SSM) [12] service. This one is identified by a (S,G) pair and it is denominated as channel. When SSM service is deployed, an IP datagram is transmitted by the source S to a SSM destination G. Receivers need to subscribe the channel (S,G) using a MLDv2 or IGMPv3 in order to receive the IP datagram.

2.3 Quality of Service

The common scenario in our days is a best-effort network connection. This means that network operators do not offer Quality of Service (QoS) for their users. In order to provide a good quality of service for all applications and users, the network operators just over-provision the capacity so that it would be possible to accommodate the peak traffic load. This solution is not efficient, considering that the internet traffic is increasing very rapidly, close to doubling each year since 1997 and that users are demanding for quality of service for some new applications like VoIP. A way to solve the problem is to provide different priorities to different applications or ensuring a certain level of performance to a data flow.

In wireless networks, the capacity is limited and very important to guarantee the quality of service for real-time applications such as VoIP or online games. The VoIP applications are very sensitive to packet delays and packet losses, so these metrics must be kept below the required values. This can be achieved by dynamically controlling the scheduling priorities for that session, taking into consideration the desired monitored metrics; this is called Differentiated Services (DiffServ) and it is defined by [13]. Another solution is to reserve resources on the path for that data flow, guaranteeing the same metrics, this technique is defined as Integrated Services (IntServ) [14].

2.3.1 Differentiated Services

DiffServ is a coarse-grained, class-based mechanism for traffic management that enables the deployment of scalable service discrimination in the Internet. This is achieved by implementing on each node queue management disciplines where each packet can be differentiated, i.e. a per-hop behavior. The architecture assumes that packets are marked in some edge-router and other routers treat this packet according to this marking. This is a

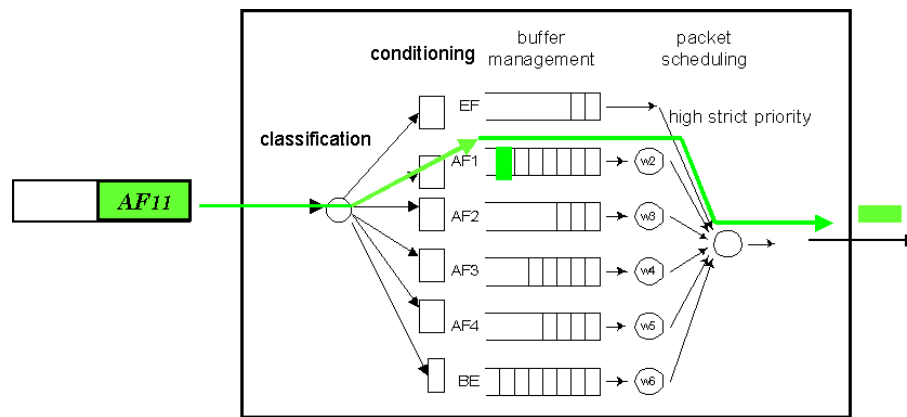


Figure 2.3: DiffServ Node.

scalable architecture because routers only need to keep the state of a few classes. There are defined three per-hop behaviors:

Default PHB - which is typically best-effort traffic; when the traffic does not match any of the defined classes, it is treated as best-effort traffic.

Expedited Forwarding (EF) PHB - traffic is treated as high strict priority. This is used for low-loss, low-latency real-time applications, like VoIP [15].

Assured Forwarding (AF) - it provides assurance of delivery as long as the traffic does not exceed the initial specified rate [16]. When congestion occurs, the traffic that has a higher probability of being dropped is the one not complying with the pre-defined rate.

Four AF classes were defined, and in each class, a minimum bandwidth was guaranteed, and it is possible to differentiate them among three different levels of drop precedence. The definition does not require, but does imply, that each AF class should use a separate queue.

2.3.2 Integrated Services

Integrated Services specify a fine-grained QoS system, where resources are explicitly managed in order to meet application requirements. A protocol is used to explicitly request resources for an application. Resource Reservation Protocol (RSVP) [17] is used by hosts to request specific levels of QoS for application data streams or flows. The routers between the sender and listener receive a PATH message with the traffic specification (TSPEC) and request specification (RSPEC). Each router needs to decide if it can support the reservation and send a reservation message (RESV) back to the sender. All routers involved need to store a state for each flow and police it. A soft state is used in order to cancel a reservation when a node does not refresh the reservation path. The individual routers may, at their option, police the traffic to check that it conforms to the flow specs.

2.3.3 Quality of Service Parameters

A quality indicator is needed in order to analyse the quality of a multicast session. QoS parameters are used to evaluate not only the session quality but also the performance of the network. The most important parameters are described next, taking into consideration that these are imperative for multicast and multimedia environments. For real-time communications, one of the most important parameter is one-way-delay because it can reduce the interactivity dramatically. The delay variation, together with the packet loss ratio can also affect the applications performance, e.g. for VoIP applications, these are critical parameters.

2.3.3.1 One-way-delay

The definition of delay is the latency between the transmission of packets by one host and successful reception by other host. The delay must consider not only the network latency, (i.e. time difference between the transmission of packet by a network interface and the reception by another one that belongs to a different host) but also the time that it takes to transmit the packet from the application to the network interface.

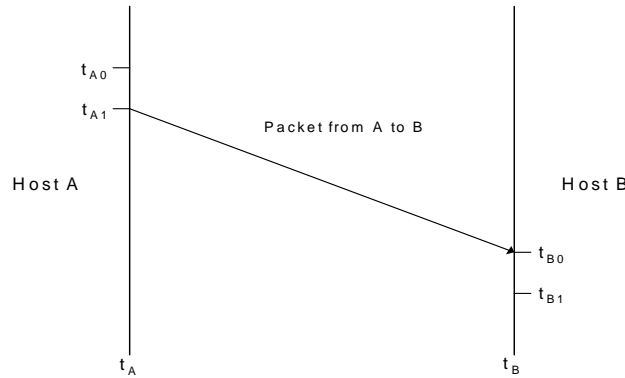


Figure 2.4: One-way-delay between two hosts.

One-way-delay (OWD) measures delay in one direction, e.g. from host A to host B. This is useful for video streaming applications that use unidirectional flows, but also for bidirectional flows used by video conference applications, because the delay can have different values in each direction that must be considered. The hosts clocks should be highly synchronized to measure OWD. An example is given in Figure 2.4, where each host has its own independent time line (t_A and t_B).

$$\Delta t_{OWD} = t_{rcvd} - t_{sent} \quad (2.1)$$

In the time $t_{A0} = t_{sent}$, the application sends a packet to host B with a timestamp inside. The packet leaves the network interface from host A at t_{A1} . This delay is introduced by the operating system and the networking hardware. Host B receives the packet from

A at $t_{B0} = t_{sent}$. The application running on host B receives the packet from the network interface on t_{B1} , and extracts the timestamp sent by host A. The OWD is calculated by Equation 2.1, and it is the difference between the received time (by host B application) and the sent time (by host A application). In order to synchronize the clocks of both hosts, it is necessary either, to use NTP protocol, or a GPS on each host.

2.3.3.2 Delay Variation (Jitter)

The delay variation can be defined as the inter-arrival variation of packets at the receiving host over a period of time [18].

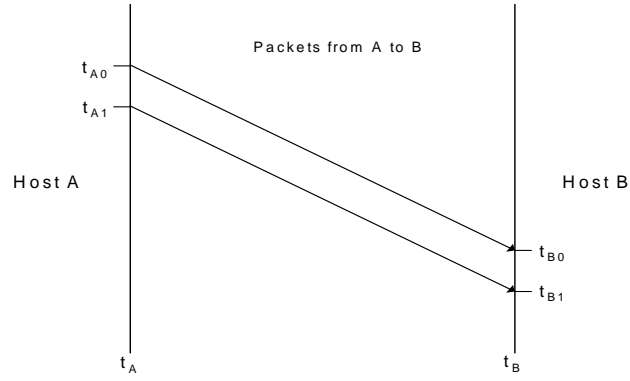


Figure 2.5: Delay variation (Jitter).

There are two different mechanisms to calculate jitter, one assumes that a timestamp is sent inside the packet and the other one uses the difference between time of arrivals between packets. For each method, it is applied a percentile (usually 90th or 95th) in order to eliminate the single, and exceptionally high, delays.

The delay variation is calculated using the OWD measurements. Equation 2.2 presents this method that relies in the clocks synchronization and on a timestamp that is sent inside the packet.

$$\Delta t_{jitter} = \max_k \left(\left| \frac{\sum_{i=0}^n \Delta t_{OWD_i}}{n+1} - \Delta t_{OWD_k} \right| \right) \quad (2.2)$$

The variation of the interarrival time can be calculated based on the inter-time arrival of packets to host B. For this calculation, a constant flow with a well-defined inter-packet distance is required. This method does not require to have the clocks synchronized and to send a timestamp inside packets because it relies only on the host B clock. In Equation 2.3, it is shown how the interarrival time is calculated and only based on the time of the arrival of packets to host B. The interarrival variation is given by Equation 2.4.

$$\Delta t_{interarrival_n} = t_{B_n} - t_{B_{n-1}} \quad (2.3)$$

$$\Delta t_{jitter} = \max_k \left(\left| \frac{\sum_{i=1}^n \Delta t_{interarrival_i}}{n+1} - \Delta t_{interarrival_k} \right| \right) \quad (2.4)$$

Multimedia application already uses similar methods to calculate the jitter. Real Time Protocol (RTP) [19] introduces these mechanisms in order to manage the playback buffer implemented in the application. A timestamp is sent by the source, inside the RTP header, to the receiver which calculates the jitter. Nevertheless, the jitter is a parameter that should be carefully calculated in order to predict the behavior of an application.

2.3.3.3 Packet Loss Ratio

The real-time applications are very sensitive to high packet losses, so this is in fact one of the most important parameters. High packet losses are the cause of the lack of resources on over-provisioned networks.

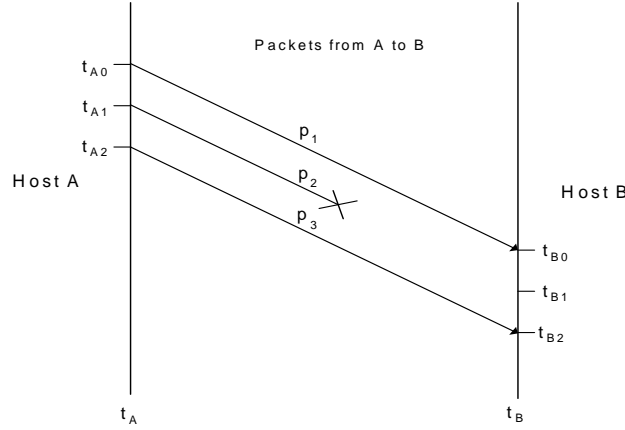


Figure 2.6: Packet Loss between two hosts.

To measure packet loss, a sequence number on the sender, i.e. host A, must be inserted. An host detects that a packet was lost when it receives non-consecutive sequence numbers. Figure 2.6 gives an example of how it can be calculated. Host A adds a sequence number to each packet it sends. Host B receives p_1 and then p_3 , and since the subtraction of both sequence numbers is higher than 1, it is detected that packets were lost. The number of packets lost is given by the difference between the sequence number minus 1.

RTP protocol provides a sequence number on the header that can be used to calculate the packet loss ratio.

2.4 Network monitoring

Network monitoring is an important function for network management because it facilitates the detection of fault links and network performance degradation. Moreover, it helps planning the network growth by profiling the users traffic.

Traditionally, the monitoring function is implemented by polling a network element for the current status. Simple Network Management Protocol (SNMP) is used as communication protocol to poll the network elements. The SNMP protocol was developed to manage network elements and used to communicate with the SNMP agents. Essentially, SNMP agents expose management data on the managed network elements as variables (e.g. system uptime). The variables are organized in hierarchies that are described by Management Information Base (MIB). Each variable is identified by Object IDentifiers (OID). MIBs use the notation defined by ASN.1.

For applications that require flow-based IP traffic measurements, a SNMP architecture does not solve the problem because MIBs do not record the flow states for further processing and metrics computation. So it is important to have a standard way of exporting information related to IP flows to QoS monitoring applications, for instance. This section defines several techniques to monitor the QoS parameters of a network and presents the reference architecture that was adopted to measure and export the monitoring results to collectors.

2.4.1 Active measurement

The active measurement is performed by injecting traffic into the network. In the most typical case, there is a sender that transmits traffic to a receiver that is listening. The

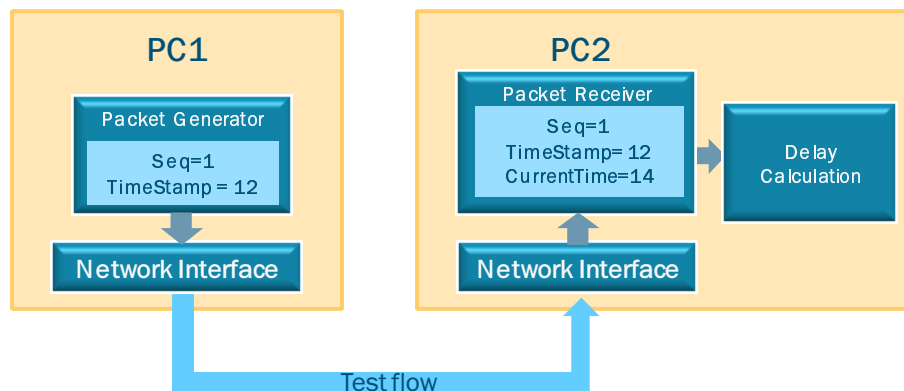


Figure 2.7: Active Measurement.

traffic distribution pattern (e.g. periodic, poisson, burst, etc) is configured in the sender. Additionally to this parameter, it also requires the definition for each distribution of some extra ones, e.g. for the periodic traffic pattern, the parameters are the packet rate and the payload size. The sender must send inside each packet a sequence number and a timestamp. This one will correspond to the time at which the packet was sent by the sender, as for the sequence number, it is necessary to track the number of packets that are lost or to check if they reached the receiver in a different order. In the receiver, the sequence number and the timestamp are used to compute several QoS metrics. To compute OWD, the receiver extracts the timestamp from each packet and subtracts the time of arrival of

that packet. Moreover, packet loss can also be calculated using the sequence number. This metric has different definitions, but the most common one, adopted by [RTP](#), just counts a packet as lost if it just arrives in a different order or if it simply does not arrive because it was discarded by the network. These errors are detected by the receiver when the sequence number, that travels inside the packet, is not consecutive with the one received previously. The main disadvantage of this measurement is that this solution is intrusive from the network point of view, since it injects traffic. It also demands a time synchronization between both probes. The advantages are enormous, it is up to the probe to schedule the tests, it can select the desired traffic pattern, and the amount of injected traffic can be controlled. The solution is considered scalable because the volume of information and the number of probes is controllable and so is the computation time for the intended metric. Active measurements are used in different applications but most of them are related, either to network maintenance, or to network performance monitoring. In the case of maintenance, it is used to detect the point of failure in the network and also to debug the problem. Network performance is important because it can affect both users and services performance, so it requires to know in advanced the normal network baseline. This way, when a pre-determined threshold is reached, an alarm is sent. Besides the presented applications, active measurement is also used to verify Service Level Agreement ([SLA](#)) conformance, to aid the call admission control algorithms, and for traffic engineering.

2.4.2 Passive measurement

The passive measurement relies upon the traffic gathered from the network. It is necessary at least a probe to collect measurements. The most common application is traffic volume

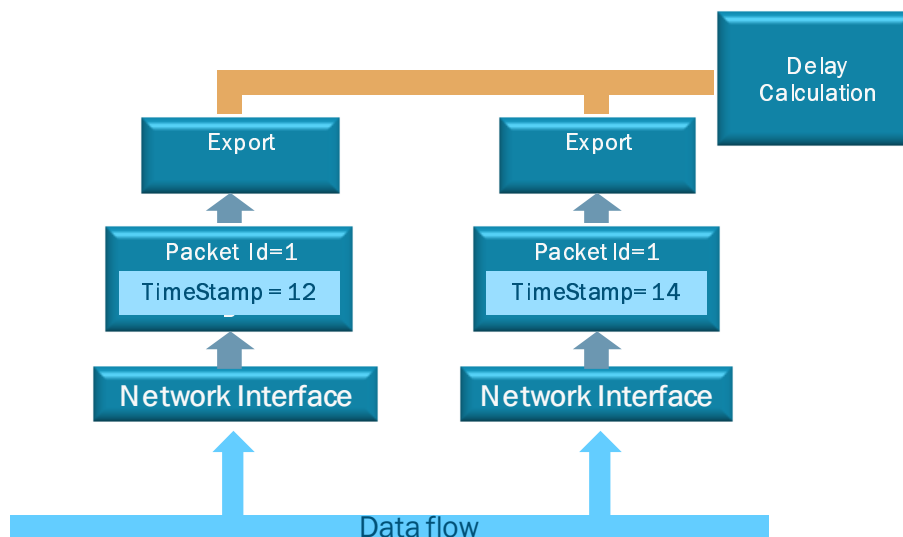


Figure 2.8: Passive Measurement.

accounting. For metrics more complex, extra calculation has to be performed. This is

done correlating the information between the probes. To measure the One-Way-Delay, the same packet must be detected in different points of the network. In this case, to compute a hash for the packet on each probe is mandatory. That later one can be used to correlate packets and compute delay. For each packet captured by the probe, a packet identifier must be extracted. A hash function can be used to calculate a unique identifier for a packet. Moreover, the time of arrival of the packet is also collected. The packet identifier and the timestamp are sent to a calculation function that can be co-located in one of the probes or in a different one. The packets are correlated and the timestamps are compared. The delay can be considered as the subtraction of both timestamps, taking into account that the probes are located in different points of the network. The disadvantage of this solution is that all the information on the network must be captured which, in some cases, is not scalable. The main advantage is that it is not intrusive like the previous one.

2.4.3 IPFIX architecture

This section defines an architecture for IP traffic flow monitoring, measuring and exporting. It provides a high-level description of an Internet Protocol Flow Information eXport (IPFIX), device's key components and their functions. The Figure 2.9 depicts the reference architecture that is described in detail in [20]. The functional components appear in the figure, inside brackets. The applications and their interface are not defined by the standard. IPFIX protocol is used to exchange measurement information between the exporting processes and collecting processes. An export can be done to one or more collectors. Usually, there is an IPFIX device with an observation point running a metering and an exporting process, but there are some configurations where measurement results can be aggregated and exported to a collector. The IPFIX exporter exemplifies this scenario, and as it can be observed, only the exporting process is running. The applications, like traffic accounting or QoS monitoring, are the final consumers for the monitoring results.

The figure 2.10 shows a typical IPFIX Device where the IPFIX components are shown in rectangular boxes. An IPFIX device must always have at least one Metering Process that is associated to an Observation Domain. The function of the Metering Process is to observe the packets that pass through Observations Points, associate a timestamp to each one and classify them into flows. The Metering Process must then maintain a database with all the flow records collected from an Observation Domain and also the statistics associated with the metering process itself. A Metering Process may select only a set of packets that transverse the Observation Point using a specific criteria. Two different methods were defined to select packets: sampling and filtering. The sampling function determines which packets are selected for measurement, based on a sampling criteria (e.g. select the first 10 packets received at an Observation Point). The filter function only selects the packets that match the header fields with the filter condition (e.g. *protocol == TCP*). In order to better characterize a flow record, the measured Observation Point must be known, so IPFIX devices should send this information to collectors. An Observation

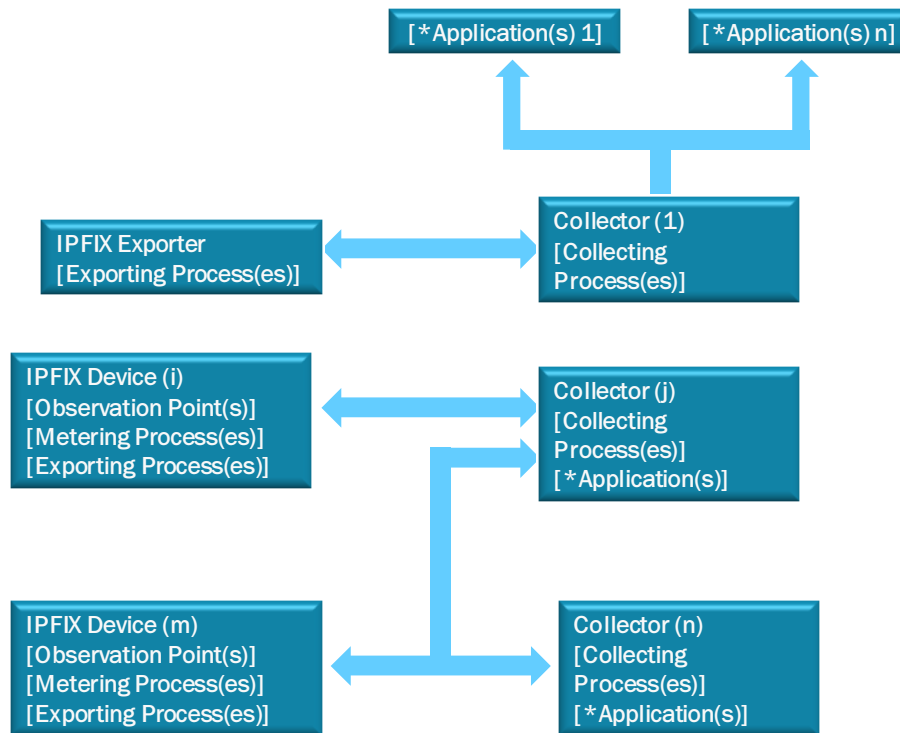


Figure 2.9: Reference Architecture.

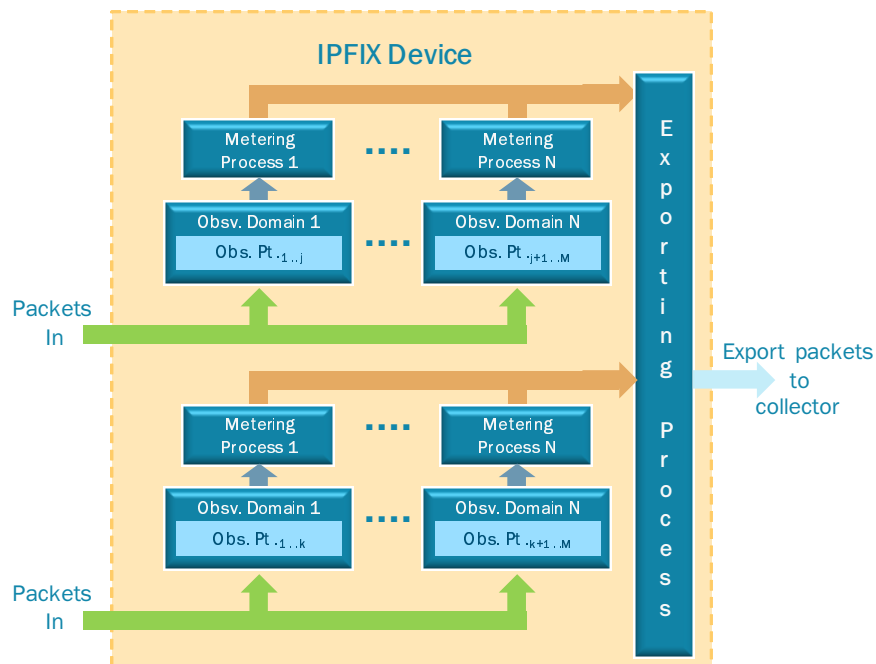


Figure 2.10: IPFIX Device.

Domain is a logical block that groups several Observation Points within an IPFIX Device. The Exporting Process is the functional block that interfaces with Metering Process(es) and with the Collecting Process on the Collector(s). It gets the flow records from the Metering Process(es) and sends data to one or more IPFIX Collectors using the IPFIX protocol. The Collecting Process must receive and store the control information. After that, it can decode and store the flow records using the control information. To interpret Flow Record's Information Elements, an IPFIX Collector needs to know the template used.

In order to transmit IP Traffic Flow information from an Exporting Process to a Collecting Process, a common representation of flow data is required, following the format of template and data sets defined by IPFIX.

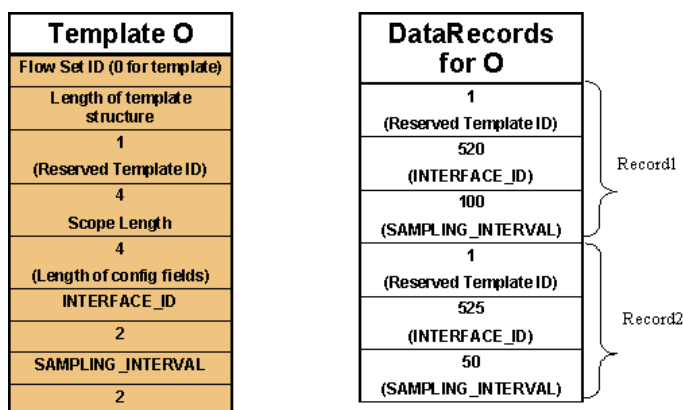


Figure 2.11: IPFIX Template.

As it can be seen on Figure 2.11, the message contains the IPFIX header and two IPFIX Sets: One Template Set that introduces the build-up of the Data Set used, as well as one Data Set, which contains the actual data. The template set only needs to be transmitted one time, since the Collection Process buffers the templates each time it receives a new one.

2.4.4 Sampling schemes

The problem with passive measurement is that the number of captured packets cannot be controlled, so a problem arises when a metric for a large amount of information has to be computed. Moreover, in the case of multi-point measurements, to compute one-way-delay for instance, the probe must export the collected information to one point, so that it will be possible to compute one-way-delay as explained before. This creates an additional problem related to the amount of measurement results that are exported from one point and travel through the network till the collector. From the network management point of view, measurements should represent a small fraction of the total operational cost, so the amount of resources spent in order to fulfill this task should be limited and small. The

best solution is to reduce the amount of traffic and computation needed by collecting the minimum amount of information.

Table 2.1: Sampling Schemes Overview.

Selection Scheme	Deterministic Selection	Content-dependent	Category
Systematic Count-based	X	-	Sampling
Systematic Time-based	X	-	Sampling
Random n-out-of-N	-	-	Sampling
Random Uniform probabilistic	-	-	Sampling
Random Non-uniform probabilistic	-	(X)	Sampling
Random Non-uniform flow-state	-	(X)	Sampling
Property Match Filtering	X	(X)	Filtering
Hash Function	X	X	Filtering

Sampling techniques appear as the best solution in order to provide information about a specific characteristic of the parent population with a representative sample. Different sampling methods are presented in [21] and [22]. A sampling method is characterized by three parameters, the sampling algorithm, the trigger type used to start the sampling interval, and the interval length. The sampling algorithm is responsible for the selection of each sample and it can be decomposed into three basic processes: systematic, random and stratified. The systematic sampling is a deterministic process that selects the start and duration of the selection intervals. There are two different systematic sampling schemes: count-based and time-based. In the count-based sampling scheme, the start and stop trigger of the sampling interval are defined accordingly to the packet spatial position, (i.e. packet count). The time-based sampling scheme defines the same triggers using time-based intervals (i.e. start time and stop time). Table 2.1 shows that both schemes are content-independent selection schemes. On Figure 2.12, these schemes are the ones that require less processing effort because they are based in simple timers and counters.

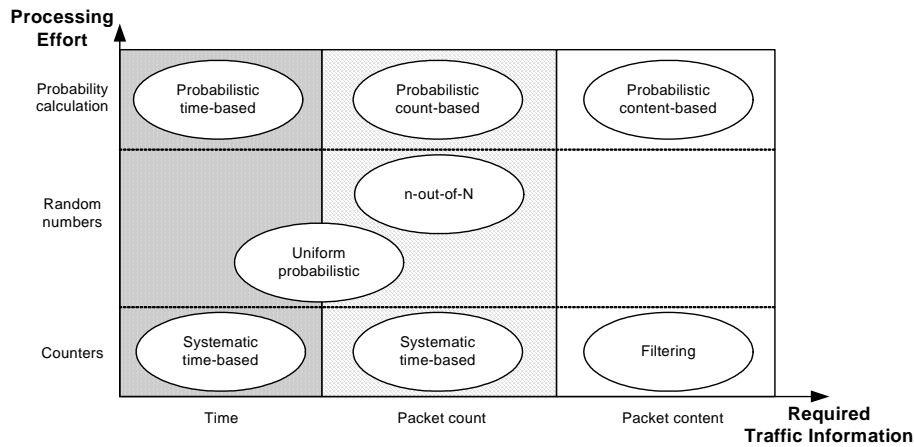


Figure 2.12: Sampling Schemes.

The random sampling scheme relies on a random process to select the starting points of the sampling interval. With this scheme, each obtained element is independent from the other, guaranteeing an unbiased estimation. It implies the generation of random numbers that require an higher processing effort when compared with the systematic sampling scheme. The standard [22] defines four different methods for random Sampling:

n-out-of-N - from N , the parent population, n elements are randomly selected. The best example is the generation of an array with n elements, and each element is a random number in the range $[1, N]$. A packet is then selected if the packet position is equal to one element of the array; in the end, n packets are selected.

Uniform Probabilistic - sampling packets are selected independently with a uniform probability p . One example is to flip a coin for each packet, and select that packet, if the coin shows tails.

Non-Uniform Probabilistic - in this method, the sampling probability depends on the selection process, so it is not constant like before. The objective is weight the sampling probabilities to enhance the chance of selecting rare packets, which may be important for the measurement process.

Non-Uniform Flow State Dependent - a packet selection depends not only on the state of the flow to which the packet belongs to, but also on the state of the other flows that are being observed.

The random sampling methods require more processing effort than systematic ones because it implies to compute a random number, which is more complex than just counting packets.

The stratified sampling is composed by several steps. The first step is to group packets from the parent population into subsets using a specific criteria, and then apply to each of them the sampling method. One example is to segment the parent population into time intervals, using a time-based sampling scheme. For each time interval, a random sampling process selects packets with a probability p .

Filtering is a deterministic process for selecting packets based on the packet content. The disadvantage is that it requires more traffic information than the other deterministic processes (count-based and time-based sampling). There are two filtering techniques:

Property Match Filtering - a packet is selected when a field of the packet header is matched with a predefined value (e.g. `ipv4.protocol == TCP`). In [23], there is a specification of all **IPFIX** flow attributes and a support for the usage of masks and ranges by adding explicit fields (e.g. netmasks for **IP** addresses).

Hash-based Filtering - a hash function is used to generate a pseudo-random variate based on the packet content, or some portion of it. The packet is selected if the

generated variate is an element of the pre-defined selection group. Using this technique, the selection of the same subsets of packets in different Observation Points is possible.

The hash-based filtering needs more processing effort than any other schemes because it has to compute a hash based on the packet content. Moreover, it can requires more information because the hash can only be calculated by using some specific fields, or in some cases, using the payload. This is performed in the scenario of multi-point measurements, where it is necessary to compute metrics using measurements from more than one Observation Point. To guarantee that the same packet is selected, the computed hash number should be the same on both points. To do so, the header fields and the packet payload must be equal. In some cases, some fields have to be excluded (e.g. Hop Limit on IPv6 changes every hop).

2.5 Multicast monitoring tools

The section presents some measurement tools that are able to compute QoS metrics for multicast networks. Each of them uses a different procedure or method to obtain similar results.

2.5.1 Multicast Reachability Monitor (MRM)

The multicast reachability monitor (MRM, [24]) was developed to facilitate the automated fault detection and fault isolation in large multicast networks. It works on a centralized way with several probes scattered over the network and it can send alarms in real-time when there is a reachability problem. The MRM is composed by three different components, the MRM manager, the MRM test sender and the MRM test receiver. The MRM manager controls the measurement process by starting and stopping the MRM test sender/receiver components and it informs the diagnostic tests to run. It also specifies the type of reports that MRM test receiver should send and collects the generated reports. Once the MRM manager starts a test, the MRM test sender creates a packet stream and sends it into the network. The MRM test receivers analyse the packets sent and report the acquired information to the MRM manager.

Figure 2.13 provides an overview of the MRM components interaction. The MRM manager is running on a router, it configures the other routers and end-systems to either perform the MRM test sender function or MRM test receiver. There is also one case where a router assumes both functions. The MRM test senders generate a packet flow that is received by the MRM test receivers. Each MRM test receiver computes the measurement results, like packet loss, and if it is below a threshold, it estimates that the router is not reachable.

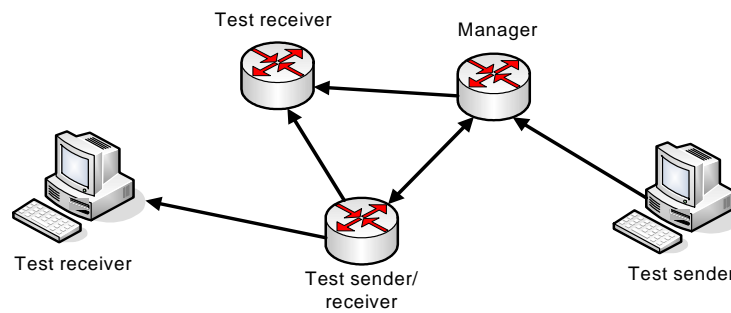


Figure 2.13: Multicast reachability monitor.

2.5.2 Multicast Beacon

The NLANR/DAST (National Laboratory for Applied Network Research/Distributed Applications Support Team) created the Multicast Beacon on 2004. The tool was developed in *perl* and was intended to be a multicast diagnostic tool that would provide statistics and diagnostic information about a multicast group's [QoS](#). Multicast beacon working principle is very similar to the MRM. An example of a configuration is depicted in [Figure 2.14](#).

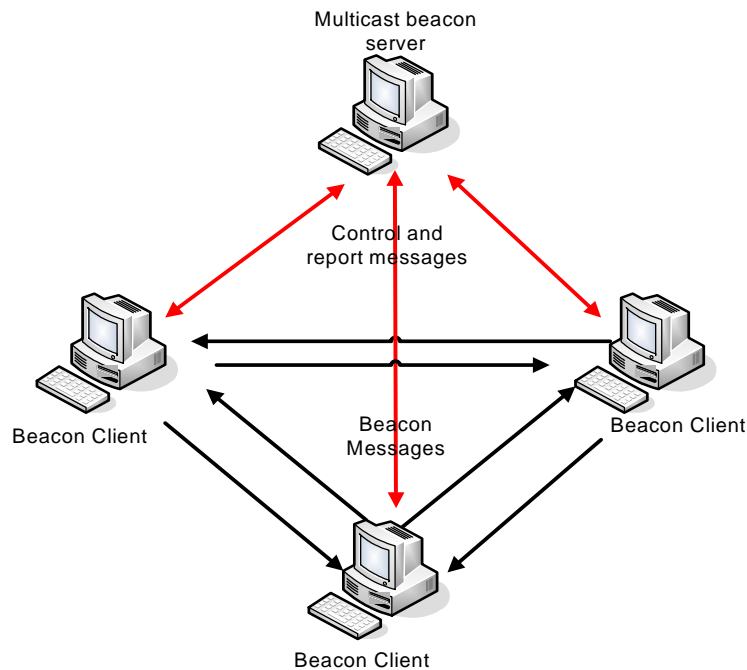


Figure 2.14: Multicast Beacon.

There is a multicast beacon server that controls the multicast beacon clients and which also collects the measurements results and computes the QoS parameters. The multicast beacon clients are periodically sending IP multicast packets using the RTP protocol to a predefined multicast group. The multicast beacon clients reports the measure data to

the multicast beacon server. The server computes a matrix with performance indicators for all the active multicast beacon clients. These results can be accessible through a web server.

2.5.3 Multicast Quality Monitor

The multicast quality monitor (MQM) was developed to measure the reliability and the quality of service of an IP multicast network [25]. The main objective of the MQM is to aid a network administrator to detect a fault, or a performance degradation, of an IP multicast network. This is achieved by measuring the connectivity between several nodes and estimating the QoS for each of these connections. In Figure 2.15, an example of a centralized management scenario is shown. This tool also allows a distributed scenario, but in order to interact with other Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services (DAIDALOS) components, a central station that manages all the measurements and collects the information is required.

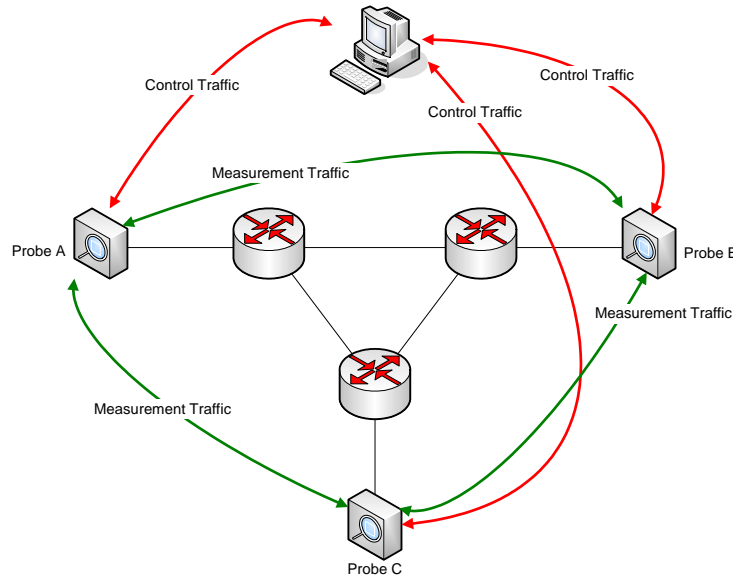


Figure 2.15: Multicast quality monitor centralized control.

The control traffic is independent from the measurement traffic. A management station controls all the probes and collects the measurement results. Next, the collected information can be used to calculate QoS parameters such as the one-way-delay. The probes just inject measurement traffic (i.e. RTP streams) into the network and collect the results that later are exported to the management station. This solution is similar to the ones presented before but it can calculate more metrics and it is less intrusive.

2.6 Context for network monitoring - DAIDALOS Project

The [DAIDALOS](#) [26] project's vision is that mobile users need to access to different services seamlessly and they are not concerned about the underlying technology. They also demand for personalized services that are transparently provided. To achieve this vision, there must be an integration of a complementary range of heterogeneous network technologies, in a scalable and seamless way. Moreover, for the network operators and service providers this becomes a good opportunity to develop new business models and new profitable services (voice, data, multimedia) for this integrated mobile world. The main objective of the project is to develop, test and demonstrate an open architecture based on a network protocol (IPv6) that integrates all the network technologies with [QoS](#) capabilities, under a common Authentication, Authorization, Accounting, Auditing and Charging ([A4C](#)) framework, and in a secure communication environment. The provision of seamless end-to-end [QoS](#) in such a demanding and heterogeneous scenario, with no perceived service degradation for the user when moving across different access technologies, is one of the main challenges in Daidalos. Briefly, the network architecture and their elements focusing in the network monitoring and QoS area will be described, namely because these are the main topics of this thesis.

The architecture is composed by to three main areas; [QoS](#) management: policy based management and network monitoring. The [QoS](#) management is performed by several elements and their main function is to implement admission control mechanisms, negotiate the [QoS](#) for each service and application and implement traffic shaping and policing techniques. Moreover, policy based management system manages and configures the network elements through policies, facilitating the administrator work and also the negotiations with other administrative domains. Network monitoring aids in the admission control process and collects information about the user traffic volume that is exported to the billing system.

2.6.1 Network Architecture

Figure 1 depicts the proposed QoS architecture that supports several access networks, each of them capable of handling several access technologies, like UMTS, DVB-T, 802.11, 802.16.

The shown QoS architecture allows for different operators to work in a common environment, with support for access services and other transport and advanced services. All operators may have special contracts between each other and/or federation mechanisms, enabling a better integrated service to the end user. [DiffServ](#) is used to support [QoS](#) in the core network, achieving scalability and performance.

In broad terms, this architecture is more flexible, and presents a more comprehensive set of characteristics, such as: a fully integrated approach to [IP](#)-based communication

with different types of applications and protocols, including adaptive applications, multicast and broadcast; the customization/optimization of the architecture according with the expected service mix to support; and the integrated support of multiple QoS service models, according to the overall network configuration (defined by operator policies).

In Figure 2.16, several access networks are depicted, connected to a core network; each administrative domain is connected to other domains through Edge Router (ER). In each access network, Mobile Terminal (MT), Laptops and PDAs, are connected to the network through Access Router (AR). Each AR may incorporate a QoS client able to request QoS resources (and/or QoS services) to the network in an implicit or explicit way. The

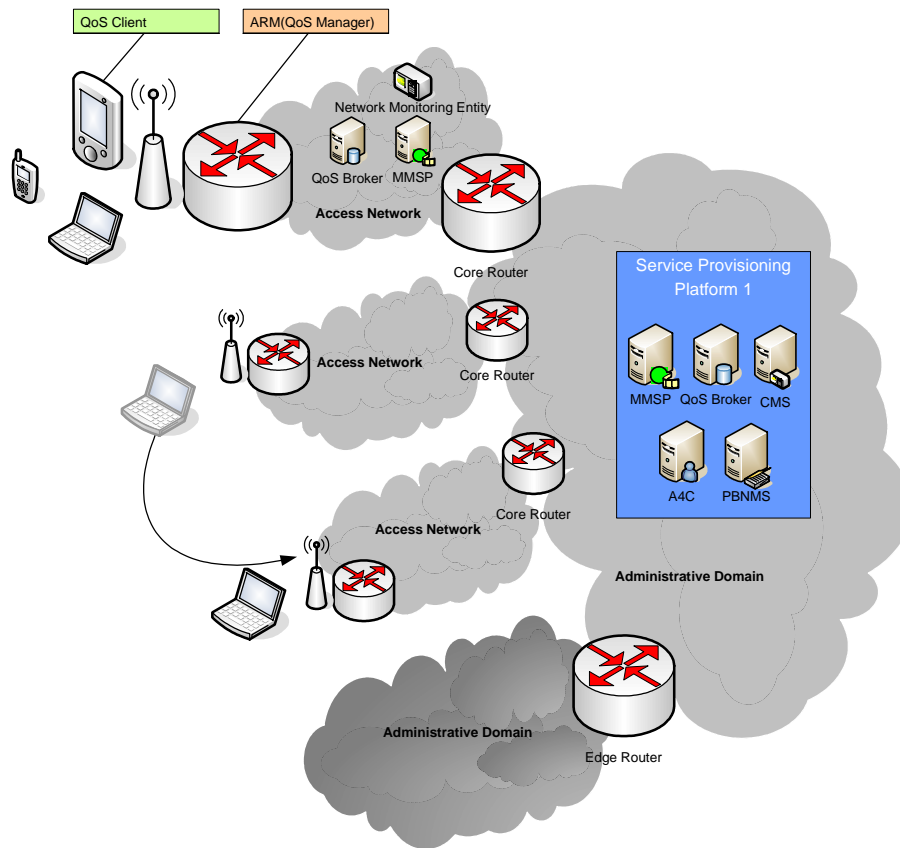


Figure 2.16: Daidalos QoS network architecture.

QoS Broker performs admission control and manages network resources. It also performs load balancing of users and sessions among the available networks (possibly with different access technologies) to optimize the usage of operator resources and maximize operator's income, by setting off network-initiated handovers.

The Core Network Quality of Service Broker (CNQoS Broker) manages the core resources in terms of aggregates, and the communication with other administrative domains.

While basic QoS services are provided intrinsically by the Access Network (AN), more advanced services are supported by a Service Provision Platform (SPP) in the core network.

In the **AN**, service proxies are deployed for efficient service provision. The MultiMedia Service Proxy (**MMSP**) controls the multimedia sessions. **MMSP** and **QoS** Broker in the Access Network Quality of Service Broker (**ANQoS**) can, together, provide the adequate network-level **QoS** to a multimedia stream, through the high level knowledge of active services and the available network resources. The **QoS** definitions at the domain level are provided by a Policy Based Network Management System (**PBNMS**), and then provided by the **ANQoS**s to the **AR**s in the different **AN**. For authentication and accounting purposes, an **A4C** server is also present in each domain. The **AR** contains a set of advanced functions, which comprises connection tracking, per-application flow DiffServ Code Point (**DSCP**) marking, and the means to translate other QoS reservation mechanisms, such as **IntServ RSVP** reservations, into **DiffServ DSCP** marking and **QoS** Broker requests. The entity supporting all these functions is denominated by Advanced Router Mechanisms (**ARM**).

To aid in the admission control procedure performed by the **QoS** Brokers, this architecture also includes a real time network monitoring system, which comprises Network Monitoring Entities (**NME**) located in several points of the network, and a **CMS**. The **NME**s can perform passive and active probing of the network, and the **CMS** controls the monitoring process, processing the measurements, and propagating the measurement results to the **QoS** Brokers in the network and other entities (e.g, **A4C** server for charging and **SLA** conformance testing).

2.6.2 Classes of Service

For real-time services, QoS becomes very important since these services demand for a minimum quality.

Table 2.2: QoS requirements for four classes.

Application	Delay (ms) upper bound on mean delay	Jitter (ms) upper bound	Packet loss (%)
Interactive audio and video	150	50	0.1
Transaction data interactive	400	50	0.1
Video and audio streaming, short transactions, bulk data	1000	Not specified	0.1
Legacy applications, low cost services	Not specified	Not specified	Not specified

Each network QoS class ensures certain edge-to-edge QoS guarantees, described by parameters as delay, jitter, packet loss and bandwidth availability. Based on the QoS requirements of the Daidalos architecture, it was proposed to implement 4 network QoS

service classes: conversational, transactional, streaming and best effort traffic. Performance parameters of the network service QoS classes are derived from ITU-T Y.1541. The defined service set can be treated as a subset of service classes defined in that recommendation. Since the ANQoS only has information on the network services to be delivered, it is required to map the application QoS parameters to network QoS parameters. This mapping can be made in the QoS client, ARM or in the MMSP, depending on the signaling strategy used. The network service is described by two parameters: the network service QoS class (where the class is specified by a set of QoS parameters), and the bandwidth to be reserved. The definition of the network service classes is conformant with DiffServ network architecture.

2.6.3 Monitoring System Integration

The network monitoring system is composed by a CMS and several NMEs scattered across the network. The CMS is the controlling and aggregator element for the whole monitoring system. This unit interfaces with other entities such as QoS Brokers, Accounting Gateway (AG), PBNMS. The interface with the QoS Brokers is used to fetch the network QoS information for traffic admission control algorithm. The interface with the AG is to perform SLA validation and to exchange accounting information. The NMEs are located at strategic points in the network and may perform passive or/and active measurements. Periodically, measurements information is sent to the CMS using the IPFIX [27] protocol.

2.7 Summary

The DAIDALOS project envisions a user with different types of mobile terminals with several wireless technologies integrated. For the user applications, all these wireless interfaces should appear seamlessly, maintaining session's continuity. Moreover, the network should provide mechanisms to guarantee the quality of service for some applications. IPv6 was selected for network protocol because of these requirements. It supports sessions mobility more efficiently than IPV4, by avoiding triangular routing. It also has a larger address space that allows a user, in a near future, to have several mobile devices with public IP address.

Multicast is the most efficient form to stream contents from one-to-many or even many-to-many communications. The MLDv2 was adopted because it supports IPv6 and allows the mobile terminals to join a multicast group. PIM-SM is the selected protocol to exchange multicast routing information between routers.

In DAIDALOS, information about resources usage is required in order to manage the admission of new multicast flows into the network. This information can be obtained by a network monitoring system, which coordinates the measurements tasks and collects the measurements results. Moreover, several methods can be deployed to collect the required QoS metrics that aid in the admission control decision algorithm. Active measurements

are one possible approach but they can be very intrusive. The passive measurements method can be a good solution but it raises several scalability and privacy issues. Sampling appears as the best solution to overcome the lack of scalability of the passive measurements method. A monitoring platform should be compliant with the IPFIX standard in order to be scalable and interoperable with the DAIDALOS architecture.

Chapter 3

Network monitoring requirements and system specification

The chapter is devoted to the definition of the system requirements and the presentation of the designed architecture, that complies with the work objectives and the specified requirements. The implemented metering functions are also presented and so is the way by which they are integrated on a multicast monitoring scenario.

3.1 System requirements

This section contributes to the identification of the network monitoring system requirements. These are drawn both from the DAIDALOS project and from this thesis work. The approach selected to define these requirements consisted in deriving the role of the network monitoring System in the multicast QoS signaling scenario.

3.1.1 Multicast QoS signaling scenario

In Figure 3.1, a message sequence diagram with the multicast source subscription, and a MT that issues a join to the multicast group announced by the source, are depicted. The SSM model is considered, namely its deployment in the PIM-SM protocol. The multicast source should be registered at an AN QoS broker in order to perform the resource and QoS management of the future listener requests. A Multicast Subscription System is used to communicate the needed parameters (source and multicast address, class of Services, bandwidth, etc.) between source and ARs. The ARs forward the incoming requests to the ANQoS using DIAMETER¹ protocol. CNQoS performs the management of inter-AN and inter-domains joins to the multicast group.

The multicast shared tree must have the same QoS levels on the different branches, i.e. all multicast listeners are restricted to a single level of QoS. This means that the multicast source specifies the QoS parameters for a group and multicast listeners can either accept

¹Successor of the RADIUS protocol used for Authentication, Authorization and Accounting (AAA)

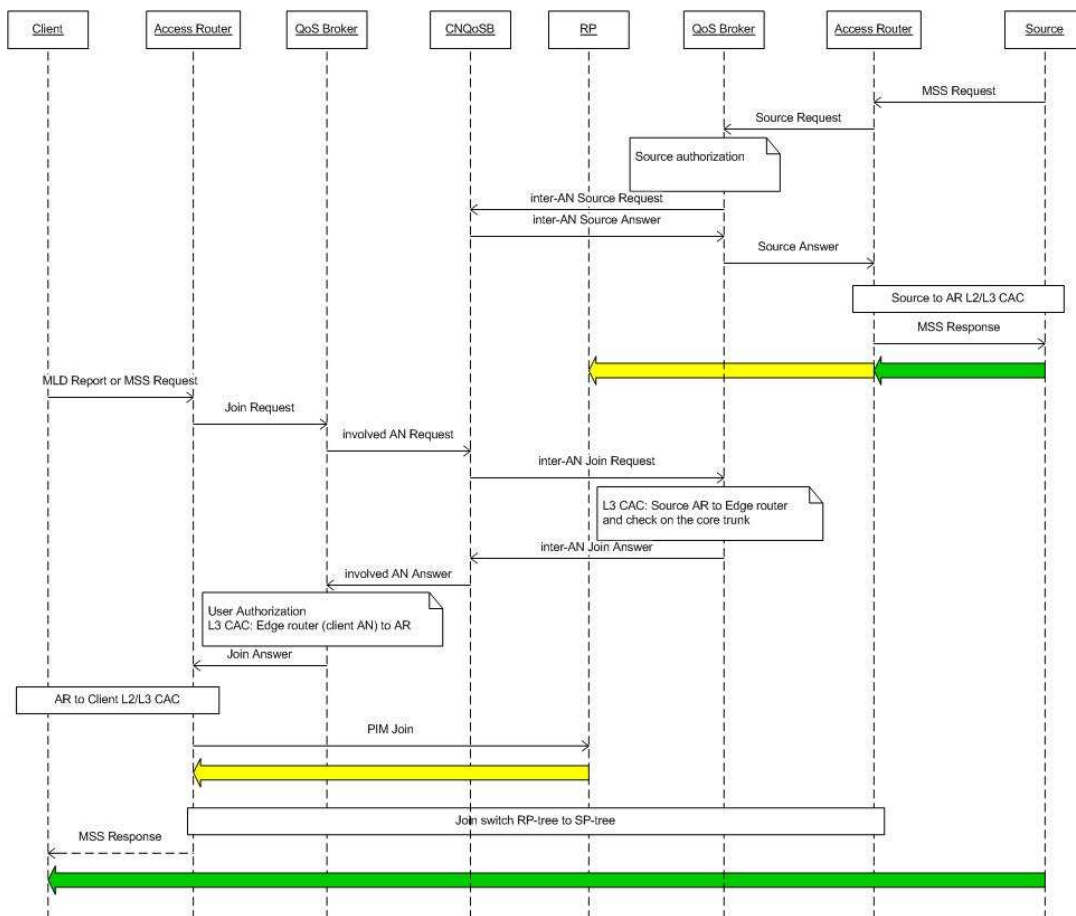


Figure 3.1: QoS multicast signaling.

or reject it. This scenario assumes that both multicast sources and listeners are registered and authenticated in the A4C. The user requests for a set of multicast groups/sources by sending a MLD Report to the AR, this one encapsulates this request into a DIAMETER message and sends it to the ANQoSB. The request is forwarded to the CNQoSB that sends the request to the ANQoSB in the AN where the source is located. This ANQoSB performs the Layer 3 (L3) call admission control, using network measurements in order to verify if there are resources available between the AR and the ER, both from the AN where the multicast source is located. It is also checked that the core trunk, connecting both ANs, has enough capacity to support the new flow. When the request is accepted, the message is forwarded through the CNQoSB back to the ANQoSB from the AN where the multicast listener is located. Before accepting the multicast flow, this Broker performs, like before, a L3 call admission control. The objective is to check the available resources between the AR that received the MLD report and the ER from the AN. If it accepts, a MLD Query message is sent to the MT (or multicast listener) and a PIM-SM join message is sent to the RP. At the end, the multicast session is established between the MT and the multicast source. PIM-SM switches from the shared tree to the shortest path tree when the listener AR receives the first multicast packets indicating the source address (thus using the RP only for source discovering).

Multicast is complex and it requires the allocation of multiple network resources. The monitoring system can provide information about the network availability, in real-time. The monitoring system can play an important role in the access network selection procedure too, gathering information on resources allocated for multicast flows in Fourth-Generation Communications System (4G), accessing technologies and evaluating the possibility of vertical handovers.

3.1.2 Network measurements for call control admission

The network measurements are used to trigger QoS control functions. The admission control function is responsible for the acceptance or rejection of a new flow (unicast or multicast), keeping the information on the resources assigned to each one of them. Although the resources reserved for each flow may take into consideration statistical multiplexing effects through the reservation of an effective bandwidth, in order to perform an accurate reservation, it is required, *a priori*, to know exactly the description of the flow traffic profile. Since this one may be unpredictable, in order to improve resource usage efficiency, making use of the statistical multiplexing gains, it should be possible to monitor the available resources in the network. The monitoring results are used for optimized admission decisions. Network measurements can also be used to manage the network connections between different access networks, or even, between administrative domains.

In Section 3.1.1, a scenario for the establishment of a multicast session with QoS is presented. From this scenario, it can be concluded that edge-to-edge active measurements are required to perform admission control for multicast flows (e.g. accept or deny a join

from a user). This requirement implies that measurements between the edge router **ER** and the **ARs** (i.e. the first hop for the terminals) must be performed. Since it is possible to have asymmetric routing, active measurements on the reverse path, i.e. from the **AR-to-ER** should be executed. Passive tests can also be performed but they require the existence of multicast traffic on the **AN**.

To accomplish the requirements presented above, several types of network measurements must be performed:

- passive metering probes - for observation of **QoS** and used bandwidth;
- active metering probes - for continuous **AR-to-ER QoS** measurements.

For some QoS metrics and decision processes based on them, to have statements about the one-way QoS in the network need to be done, e.g. about one-way delay, jitter, and loss. In this case, multi-point measurements can be used, as they are performed between two or more probes which take part in the same monitoring task. Such task can be performed using passive or active measurements. The results can be used by entities such as brokers or routers which closely interact with the network to control their behaviors, i.e. influence admission, policing, and queuing on these entities. The admission control and resource management functions that run inside **ANQoSs** need to receive regularly measurements results with this information. Based on it, they decide what flows to accept and reject.

Figure 3.2 shows how multicast measurement can be implemented. Like in Section 3.1.1, it is assumed that a source is already subscribed and there are two listeners on **AN₁**. Before admitting new multicast flows on **AN₂** and **AN₃**, network measurements must be done. A measurement task is scheduled per **AN** in order to collect measurements information about the available resources. This task schedules an active measurement between the **AN NMEs**. It can be observed that the **NME** closer to **ER** sends a multicast flow to the other **NMEs**.

In Figure 3.3, the message sequence chart of the scenario described above is shown. After starting the measurement tasks on all the **NMEs** (senders and receivers), the probes start sending packets to test the connections. **NME₁** is considered the entry point of the **AN₂** for the multicast flow, and it is the probe in the **AN** closer to the source. The other **NMEs**, the receivers, are closer to the terminals, usually installed on the first hop routers. The active tests are done either using permanently assigned multicast addresses or setting up a new multicast group. **NME₁** sends traffic to the other **NMEs** using a multicast address. Each receiver can then compute the delay based on the timestamps that are carried inside the test packets. To cover the asymmetric routing problem, the receivers also send unicast traffic back to **NME₁**. The measurement results (i.e. delay, jitter, packet loss) are then exported to CMS-DB where they can be aggregated.

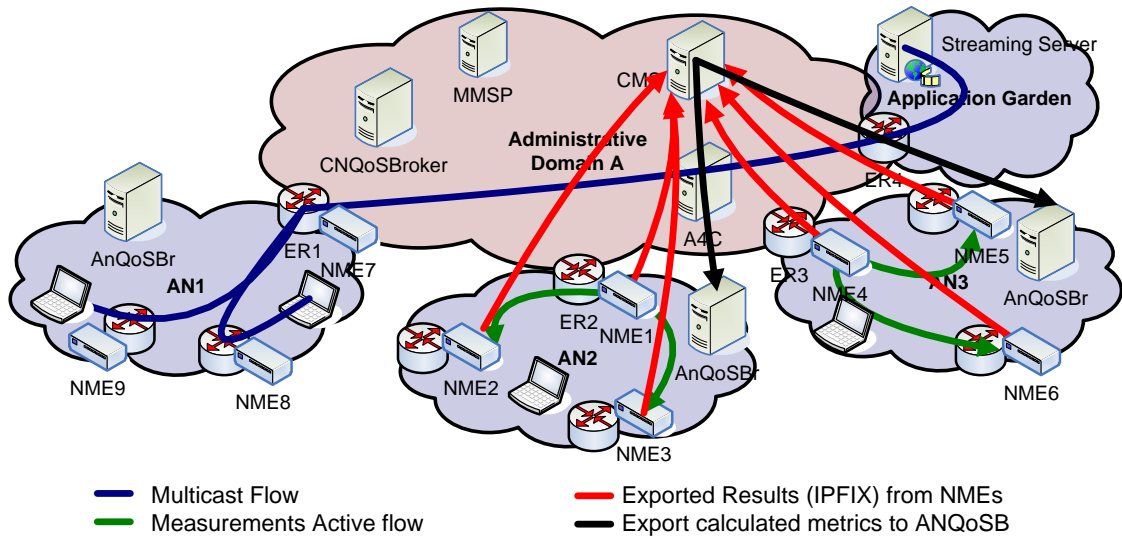


Figure 3.2: Multicast measurement scenario.

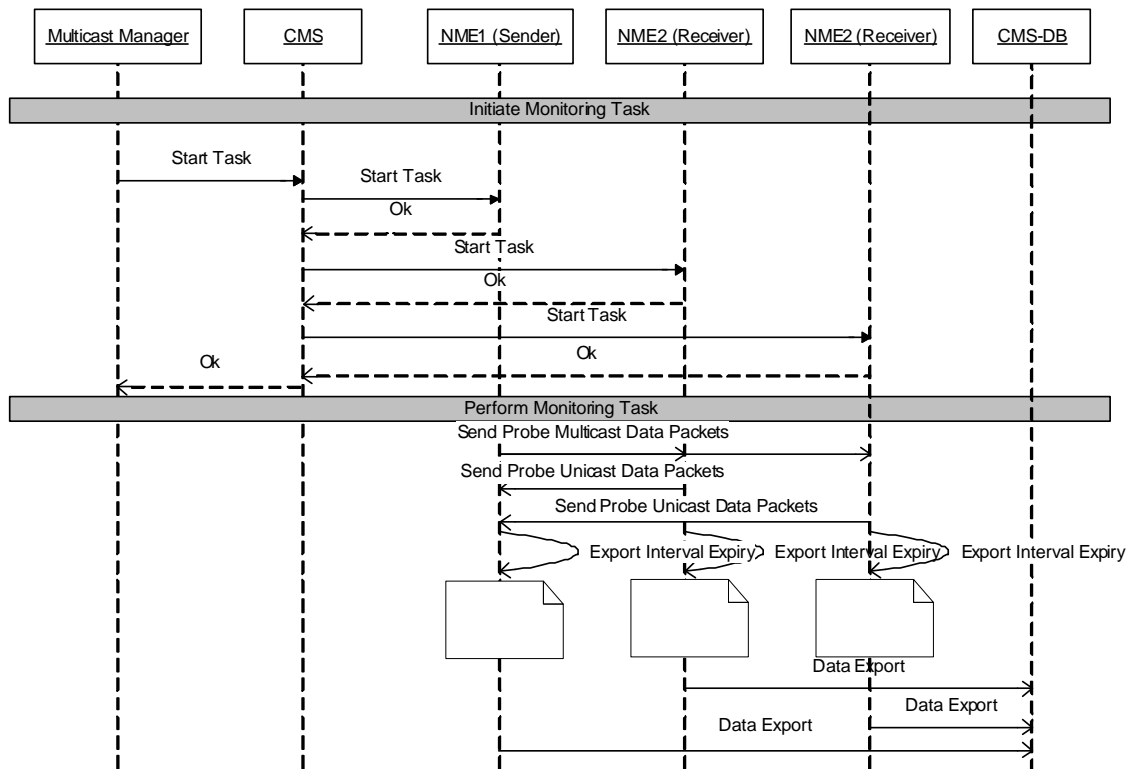


Figure 3.3: Message sequence chart for multicast measurements scenario.

3.2 Architecture design

The **CMS** is the main component of the monitoring architecture. Besides interfacing with the other **DAIDALOS** components, its main function is to manage measurement tasks and collect the measurements results in a database. The **DAIDALOS** components request the desired measurement tasks and it is up to **CMS** to configure them and to collect the results. If the measurements task is accepted by **CMS**, the results will be re-exported from the database to recipient component, i.e. the component that requested the measurement task. From this brief description, it can be observed that the required functionalities are complex. It implies a function to manage the measurement tasks requests from the other components, a function to control the probes (i.e. **NMEs**) configuring the required measurement task, and also a protocol to transfer measurements results between the components. It was decided that a platform could facilitate some of the required functionalities, so a study was conducted to evaluate four different ones. This study is briefly presented in Appendix A. The conclusion is that Open Internet Measurement Project (**OpenIMP**) platform is the broadest platform because it supports both **IPv4** and **IPv6**, active and passive measurements and it is compliant with the standards IP Performance Metrics (**IPPM**) [28], **IPFIX** [27] and Packet Sampling (**PSAMP**) [29].

3.2.1 Measurement Platform

The Figure 3.4 shows the OpenIMP Internet measurement platform which was developed for distributed **IP** traffic and quality of service measurements. This platform uses one central measurement control unit and multiple measurement units (probes). The probes are distributed within the network, and they can passively monitor network traffic or do active performance measurements. The measurement tasks are configured on the measurement control unit which distributes single measurement tasks to the remote probes. The measurement results are sent to the measurement collector, using **IPFIX** protocol, and are stored in the central results database. It is possible to do further evaluations and to display the measurement results using a Graphical User Interface (**GUI**).

The OpenIMP system consists of the following components:

Passive probes - devices connected to network taps or optical splitting boxes at special network locations (e.g. at **ARs**) that passively filters incoming traffic and perform special operations like **IP** flow classification.

Active probes - they generate and receive test traffic.

Collector - it requests/collects the measurement result data from the distributed probes and stores it in the results database.

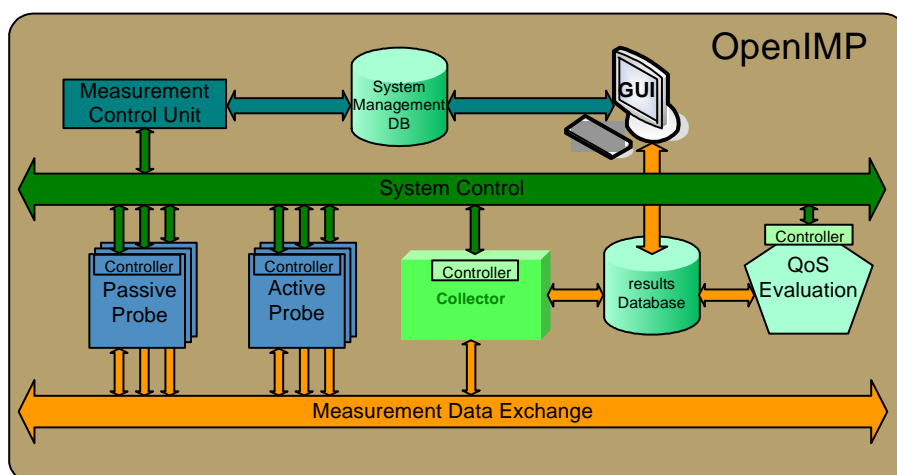


Figure 3.4: OpenIMP architecture.

QoS evaluation - it calculates IP QoS metrics like one-way-delay, jitter and packet loss from the measurement data stored in the results database. The metrics results are stored on the same database.

Measurement control unit - it manages the measurement system, pushes measurement tasks to the distributed probes and it executes complex measurement tasks.

The components can be accessed via a common control interface which offers a text based command line interpreter. Unfortunately, the OpenIMP platform does not provide an interface to outside components to manage measurement tasks. The platform has a web interface where it is possible to manage measurement tasks and evaluate collected measurement results. Moreover, it does not provide all the required metrics.

3.2.2 CMS architecture

In Figure 3.5, it is presented CMS interfaces towards the other DAIDALOS components and the attached probes, designated by NMEs. Measurements results are pushed by NMEs in regular interval, using IPFIX, and stored in the CMS database, i.e. OpenIMP results database. The other DAIDALOS components can receive the measurement results by polling or querying this database or by configuring the CMS using a SOAP API, to re-export the results to this component.

The physical export of results will be usually done from the probes (NMEs) to the CMS and then further from the CMS to the final recipients in order to ensure that the receivers of the result data will get it from a central trusted component. The NMEs, on the other hand, need a security association (e.g. key pair) to the CMS.

The CMS interfaces with the components shown in Figure 3.6. For the AG, only accounting information related to user traffic is exported. This component assumes that there is a default policy to account the traffic from specific user sessions. It also interfaces

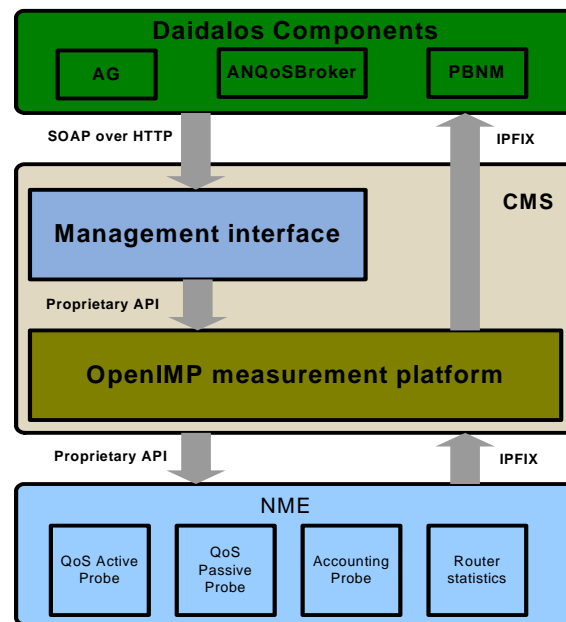


Figure 3.5: CMS - Architecture and interfaces.

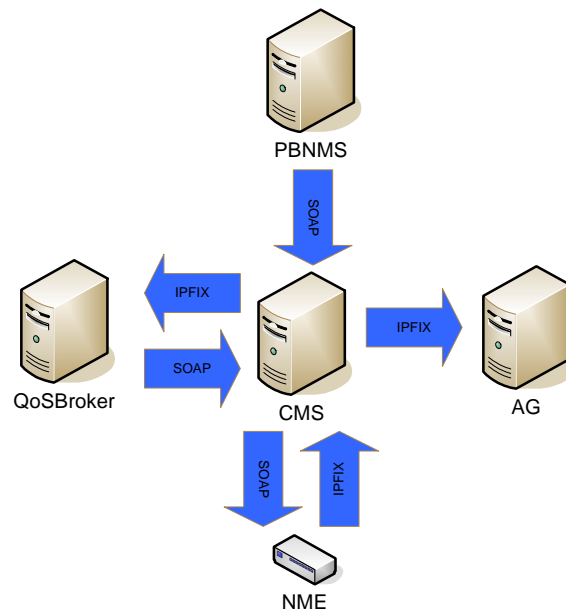


Figure 3.6: DAIDALOS components interfacing with CMS.

with the ANQoS Broker and the CNQoS Broker for configuring on-the-fly QoS measurements tasks and exports the collected measurements back to these components using IPFIX. PBNMS can configure default network measurement policies and it can monitor the status of NMEs. The NME is the other component that interacts with CMS. NMEs can be configured by CMS, using a proprietary interface, based on TCP strings. Moreover, the NME can export the collected measurements and also report their status. On Table 3.1, the exchanged information between CMS and the other components is summarized.

Table 3.1: Types of Data exchanged with the CMS.

Entity	Type of exchanged data
AG - Flow Accounting	Measurement Results (IPFIX)
QoS Broker	Measurements (IPFIX) Session termination information (IPFIX) Configuration information : Monitoring Tasks (SOAP)
PBNMS	Policies and Configuration Data (SOAP) NME Status Data (SOAP)
NME (probes)	Configuration information : Monitoring Tasks (TCP) Measurement Results (IPFIX) NME Status (TCP)

The information's model and the functionality of the main CMS modules are presented in the CMS composite structure diagram in Figure 3.7. Three components are easily identified:

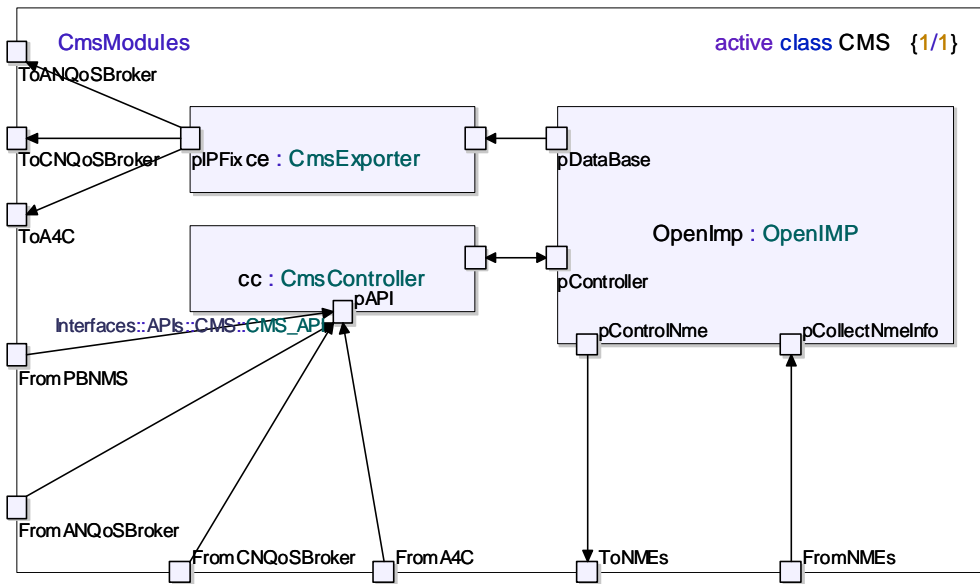


Figure 3.7: CMS composite structure diagram.

cmsController - it provides an interface to manage the addition and removal of measurements tasks and monitor the NMEs' status.

cmsExporter - it is used to export QoS measurements and accounting data to external components.

OpenIMP - it represents the database which holds NMEs and the running measurement tasks information.

The **CMS** controller exhibits a SOAP-based control interface which lets external components access its functions in order to start, schedule and terminate measurement tasks, as well as to fetch the **NMEs** tasks status. It is the central access point for all **CMS**-external applications, i.e. there is no direct communication between external components and **NMEs**, collector, or the database. The **CMS** controller receives the monitoring tasks via its SOAP control interface. The controller generates and sends the related commands to the OpenIMP measurement system. The OpenIMP controller then transfers the related measurement tasks to one of the measurement **NMEs**, depending on the type of task and the functionality requested.

NMEs send the measurement result data to the OpenIMP collector which stores the data in its central measurement result database. If some post analysis of the measurement data is needed then the analysis component is invoked. The **CMS** Exporter collects the data from the database and exports it to the specified destinations. This export happens at regular intervals or at the end of a measurement.

In specific cases where no post processing is needed but result data shall be delivered with low delay to the final receiving component (e.g. **AG** or **ANQoSB**), a **CMS** task may contain the directive to export results directly from the **NME** to the recipient without invocation of OpenIMP collector, database, and exporter. In this case, a security token on each message exchanged between the DAIDALOS component requesting the measurement, **CMS** and **NME** must be sent in order to ensure the measurement information authenticity.

The Figure 3.8 exhibits the class diagram of CMS SOAP API and Tables 3.2, 3.3 summarize the most important methods for the developed work. Two different interface classes were defined : cmsMeasurementIf and cmsManagementIf.

The cmsMeasurementIf is used to add, remove, and list measurement tasks. In order to add a measurement task, several parameters must be specified, like measurement function to use, task duration, export interval, collector address, etc. The removal only needs the taskId parameter. Moreover, all the methods from this class have a authorization token parameter, which is obtained when the user authenticates into CMS.

The cmsManagementIf manages the NMEs addition and removal. By having a list of all the NMEs available on the network, it is easier to add measurement tasks, specially when we also know their current status. There are other methods related to the user management but they are not important for the implementation.

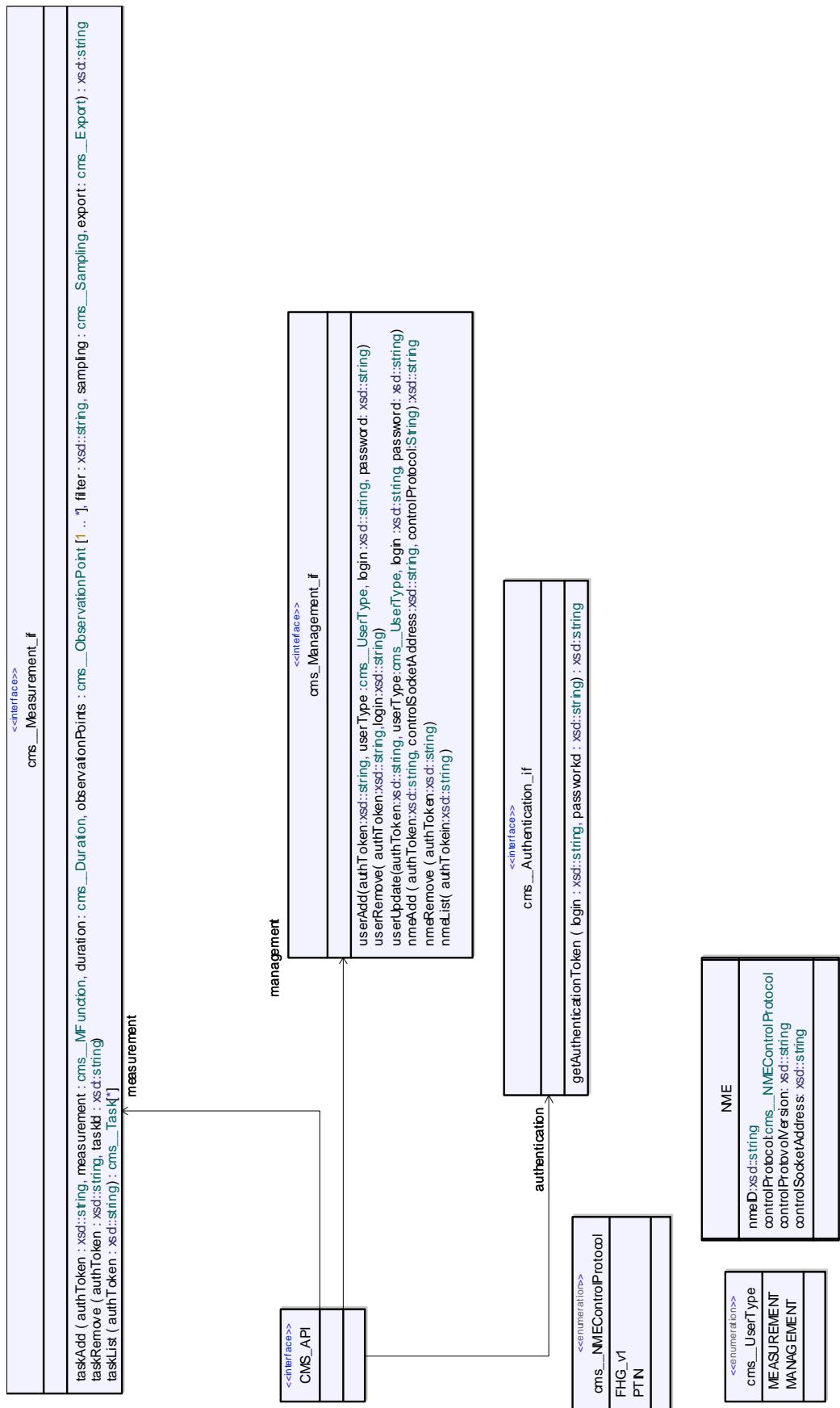


Figure 3.8: Class Diagram of CMS SOAP API.

Table 3.2: SOAP measurement interface overview.

Function	Parameters	Returns	Description
TaskAdd	Filter (srcaddr, dstaddr, protocol,) Function name, Function parameters measurement param. (start time, stop time, report interval) export parameters (collector address,)	String TaskId Ok/error	Add new task to CMS
TaskRemove	String taskID	Ok/error	Remove named task from CMS
TaskList	None	List of tasks Ok/error	Get list of currently configured tasks

Table 3.3: SOAP NME management interface overview.

Function	Parameters	Returns	Description
NmeAdd	Name, ipaddr, ctrlport, funclist, capabilities,	Ok/error	Add new NME to CMS
NmeRemove	String Name	Ok/error	Remove named NME from CMS
NmeList	None	Array of NMEs, Ok/error	Get list of all NMEs

3.2.3 NME architecture

NMEs are monitoring probes located in routers implemented by software. The measurement tasks for these probes have to be carefully setup. For example, in order to monitor the QoS metric one-way-delay, the packets should be captured on the wireless network interface, i.e. before they are sent to the wireless access point or the WiFi card, so that they also capture the time needed for routing the packets in the AR. This is especially important in the case in which measurement results from different monitoring tasks are combined.

As presented in Figure 3.9, the NME exposes two different interfaces. One is for the management of measurement tasks or to get the NME status, and the other interface is devoted to the export of measurements from the NME to the CMS collector. There are other interfaces presented in the figure that are not used in this implementation but that are intended to extend the NME functionalities. For example, to add support to inter-domain measurements.

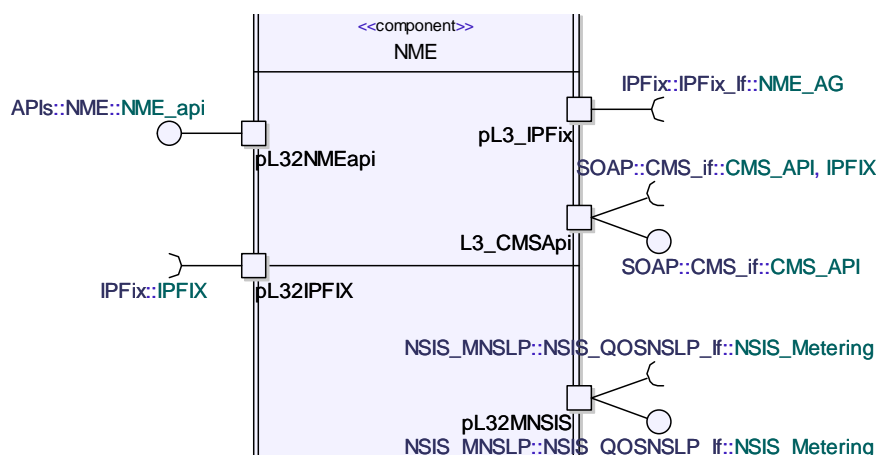


Figure 3.9: NME component diagram.

The Figure 3.10 shows a more elaborate diagram where the most important components in the NME architecture are depicted. The base components are the scheduler, the controller, and the exporter. These are responsible for the management of measurement tasks and the export of the results. In the middle of the figure, the functions that perform all the work are presented: VolumeAccount; ActiveMeasure; QoSMeasure. Each of these functions measures specific network characteristics, either by passively monitoring the network or by injecting packets into it.

In table 3.4, it is provided an overview of the NMEs controller interface. This interface is used by CMS to configure measurements tasks on the NMEs. It is also possible to fetch the information related to a measurement task or to get the current status of the NME. To add a measurement task to a NME, CMS can invoke the add_task method with the correct parameters, like taskId, filter, measurement function to run, task duration,

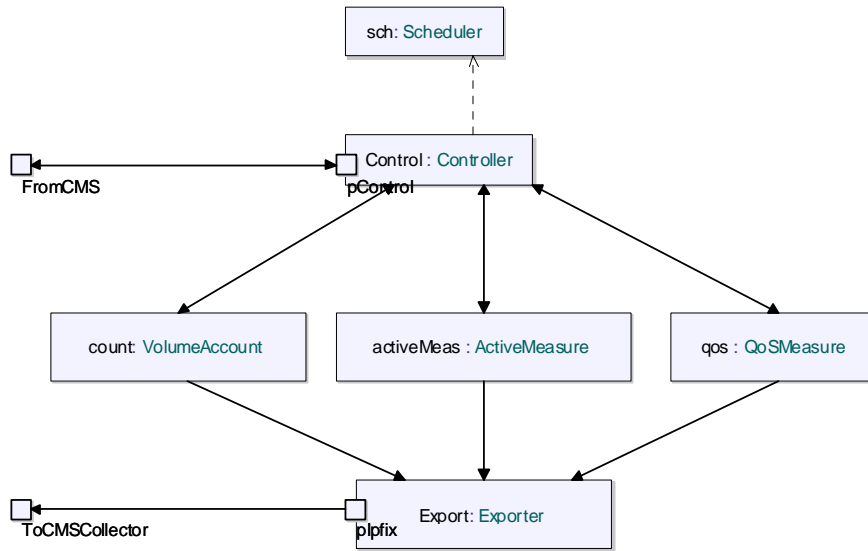


Figure 3.10: NME Composite Structure Diagram.

collector address, etc. The measurement function to be used is selected using the option “-a”, and it is possible to transfer extra parameters related to this function.

Table 3.4: NME interface overview.

Function	Parameters	Returns	Description
add_task	<code>< task_id > [-wait]</code> <code>-r <filter></code> <code>-a <action><actionopts></code> <code>[-m <moptlist>]</code> <code>-e < ropts >parameters</code>	Ok/error	Add new task to the NME. The action parameter specifies the function.
rm_task	<code><task_id></code> <code>[, <task.id>,..]</code>	Ok/error	Remove task from the NME
get_info	<code>[tasklist task=<task.id>]</code>	String	get task information
status		String	get NME status

3.3 Implementation of NME functions

3.3.1 ActiveMeasure

The function was implemented using a third party tool known as Multi-Generator (MGEN). This corresponds to an open source software developed by the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) Research Group. The tool can be used to perform IP Multicast performance tests using UDP traffic. From the tests, performance statistics can be calculated, like throughput, packet loss rates, one-way-delay, jitter. To configure the tests, a script is generated with the right parameters.

The tests are then logged into a log file that is read by this function and translated into an IPFIX template.

The sender and receiver sides should be configured differently. The sender should send multicast traffic to all the receivers on that AN. One flow per class of service, defined in Section 2.6.2, is generated to test the available resources for each class. Below, the automatically generated script on the sender side is presented. From the script, four different flows can be identified, with a multicast IPv6 address, using a site-local scope, so that all routers from that AN can receive it. Since permanently assigned multicast addresses are used, to join a multicast group is not required. The test traffic was modeled by a Poisson stream with constant payload size (of 1000 bytes) because it represents the worst case when several CBR streams are aggregated [30]. The last line is used to receive test packets from the receivers and tests if the route between the sender and the receiver is symmetric or asymmetric.

#Multicast Flows

```
0.0 ON 1 UDP SRC 5002 DST FF05::2/5001 POISSON [1000 1000] LABEL 0x1000000
0.0 ON 2 UDP SRC 5002 DST FF05::2/5002 POISSON [1000 1000] LABEL 0x2000000
0.0 ON 3 UDP SRC 5002 DST FF05::2/5003 POISSON [1000 1000] LABEL 0x3000000
0.0 ON 4 UDP SRC 5002 DST FF05::2/5004 POISSON [1000 1000] LABEL 0x4000000
```

#Receive Unicast Test

```
0.0 LISTEN UDP 5100-5109
```

In each receiver, a script to listen the multicast flows generated by the sender is generated. Below, four unicast flows are presented and, just like before, they are used to test the reverse path. The "LABEL" option is used to mark the packets with a DSCP value in order to classify them into four different classes.

File receiv.mgn

#Receive Multicast Test

```
0.0 LISTEN UDP 5000-5009
```

#Unicast Flows

```
0.0 ON 1 UDP SRC 5102 DST 2001::2/5101 POISSON [1000 1000] LABEL 0x1000000
0.0 ON 2 UDP SRC 5102 DST 2001::2/5102 POISSON [1000 1000] LABEL 0x2000000
0.0 ON 3 UDP SRC 5102 DST 2001::2/5103 POISSON [1000 1000] LABEL 0x3000000
0.0 ON 4 UDP SRC 5102 DST 2001::2/5104 POISSON [1000 1000] LABEL 0x4000000
```

Most of the parameters that appear in the script can be configured by CMS using the NME interface. The activeMeasure function is managed by the controller component, presented in Section 3.2.3. When the activeMeasure function is selected as measurement action, a specification for the sender or the receiver is required with the following parameters:

-a activemeasure send < srcport > < dstip > < dstport > < pktlen > < pktps >

-a activemeasure receive < dstport >

The DSCP is not a parameter of the function so, by default, four flows are created, one for each predefined class. In order to configure the unicast measurements, the same commands must be run again in order to create the sender and receiver of the unicast flows.

The QoS metrics owd_{min} , owd_{avg} , owd_{max} , $jitter_{min}$, $jitter_{avg}$, $jitter_{max}$, $packet_loss_{avg}$, $packet_loss_{max}$ are computed on each receiver. The metrics are then exported using the template specified below.

```
export_fields_t sla_fields[] = {
{IPFIX_ENO_DAIDALOS, IPFIX_FT_METERSTARTTIME, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_METERSTOPTIME, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_SOURCEADDRESSV6, 16},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_DESTINATIONADDRESSV6, 16},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OWDMINDELAY, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OWDAVGDELAY, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OWDMAXDELAY, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_MINJITTER, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_AVGJITTER, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_MAXJITTER, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_AVGPACKETLOSS, 2 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_MAXPACKETLOSS, 2 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_THROUGHPUT, 4 }
};
```

The exported template has all the required measurements between two end-points; in this case, the sender and the receiver of the active test. The CMS can later aggregate the information from the multicast and unicast tests and send it to the final recipient, i.e. ANQoSB.

3.3.2 QoSMeasure

The implementation of the QoSMeasure function, that passively measures the multicast traffic, is described next. The main objective of this function is to capture all the multicast flows between the [ER](#)-and-[AR](#) and, for each packet, calculate a unique identifier. The flow

records are then exported to [CMS](#) collector using [IPFIX](#) protocol. The Packet Capture library (*libpcap*) was used to capture the packets because it provides a high level interface. The NME interface has a parameter called filter that can be used by this function to capture specific traffic. For instance, there may be times when it is needed to capture multicast sessions with destination port=1234. The chosen library also supports filters, so the original [NME](#) parameter can be forwarded to *libpcap*.

The procedure is quite simple: when a packet is captured by *libpcap*, a callback function is called to process the packet. This function will perform the packet selection and it will also store all relevant fields in a flow record. This means that the function keeps track of all the flows. The main problem is that the required metrics imply multi-point measurements, i.e. the same packet must be univocally identified in more than one observation points. The solution mechanism devised to solve this problem is message digesting, which consists in hashing the packet with an algorithm, like SHA1 (Secure Hash Algorithm) or MD5, resulting in a small length hash. This can be considered like a packet signature, even though ciphering is not employed. The hash univocally identifies the packet among all the others, with a certain probability of error. However, there is an important issue that must be considered before computing the hash, the packet content used to compute the hash should be the same on both observation points. Since the observation points are located in different routers, there is at least one IPv6 header field that is modified, the Hop Limit (this field is decreased each hop). In order to get the same hash in different observation points, this field should not be considered to the hash function. Each time a packet arrives, a timestamp for the arrival time must be stored. For computing some metrics, time synchronization between observation points must be ensured; for example, NTP protocol can be used to synchronize all observations points. Moreover, a new flow record is created when a packet arrives and does not belong to any of the stored flow records.

As mentioned in Section [2.4.4](#), it is impossible to capture all the packets, process them and export measurement data to CMS. To overcome this problem three sampling schemes were tested:

- time-based;
- count-based;
- hash-based;

These sampling schemes are applied on per flow basis. This can be identified as stratified sampling method, because it implies a first selection of the population and, after this, a sampling scheme is used. In [\[31\]](#), several methods were compared and the stratified sampling method was characterized as being the more accurate. For the time-based and count-based sampling schemes, parameters were used to characterize the sampling, the interval and the spacing. The interval can be spatial or time variable and it is used by the

sampling method to identify the duration of the capture. Spacing is variable like interval but it identifies the amount of time, or the number of packets, not captured between each interval.

The time-based sampling is implemented using a timer function, i.e. for each flow two different timers, one for interval and the other with spacing time, were developed. When a packet arrives, and the flow record is already created, the function retrieves the time of arrival of the last packet received. Using this information, and with the current time, it is possible to classify the packet as to be inserted into the flow record or to discard it.

The count-based sampling is more simple; for each flow the total number of captured packets on the flow record is maintained. Each time a packet is captured and belongs to flow *i*, the `packet_count` is retrieved for that flow. If the conditions presented below are met, the packet is discarded, if not, the relevant packet information is inserted in the flow record.

```

    If ((flow_rec->packet_count[i] % (opts->sinterval + opts->:sspacing))
        < (opts->sinterval + opts->:sspacing))
    && ((flow_rec->packet_count[i] % (opts->sinterval + opts->:sspacing))
    >= (opts->sinterval))) {

mlogf (2, "Count-Based sampling - ignored packet\n");
}
else {
mlogf (2, "Count-Based sampling - insert packet\n");
}

```

Hash based filter is the other scheme implemented. The scheme requires more processing effort because it implies, before applying the sampling method, to compute a hash based on the packet content. This process was already described before, since the hash is always necessary in order to compute some metrics. In fact, in some situations, we can consider a different identifier extracted from the header information. For instance, if we consider RTP traffic, the sequence number can be used to identify a packet. Although, this scheme comes from the filtering of specific packets based on the computed hash, it can be guaranteed that this is a random process, because of the SHA1 algorithm used to generate the hash. A rate is specified and a mask is generated according to this parameter. Each time a packet arrives, the hash is computed and a logic operation is executed between the mask and the hash number. If the result is 1, the packet is selected and inserted into the flow record.

The sampling parameters are configured by CMS using the NME interface. The QoS function is managed by the controller component, presented in Section 3.2.3, that performs the interface with CMS. When it is selected, the QoSMeasure function as measurement

action can specify the sampling parameters:

-a qosmeasure < sampling - method > < interval > < spacing > < rate >

The sampling-method and its parameters can be configured. If the time-based or count-based sampling scheme is chosen, it is mandatory to also add the interval and the spacing parameter. If the hash-based sampling scheme is selected, it is only needed to specify the sampling rate.

There is another important interface that is used to export the measurements to the CMS collector. It is required to specify two export templates that are related. The first one, depicted below, describes a flow, i.e. 5 tuple source and destination IP addresses, source and destination ports and protocol. Besides this information, the samplerId, or IP address of the probe, is also provided in order to identify the observation point for further processing. Generic flow statistics, like total number of packets and average throughput and the DiffServ code point of the class of service, are also exported.

```
export_fields_t qosMeasure_fields[] = {
{IPFIX_ENO_FOKUS, IPFIX_FT_TASKID, 4},
{IPFIX_ENO_FOKUS, IPFIX_FT_FLOWID, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_SAMPLERID, 20},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_SOURCEADDRESSV6, 16},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_DESTINATIONADDRESSV6, 16},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_TRANSPORTSOURCEPORT, 2},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_TRANSPORTDESTINATIONPORT, 2},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_IPPROTOCOL, 1},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_DSCP, 1},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_NUMPKTS, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_THROUGHPUT, 4}
};
```

To each flow record, a captured packet information is associated and presented below. The mapping between both templates is done with the IPFIX_FT_TASKID, IPFIX_FT_FLOWID and IPFIX_FT_OID. Moreover, the hash number represented by NSEQ (in the template IPFIX_FT_NSEQ) is sent together with the time of the arrival of the packet, in order to compute metrics like one-way-delay or jitter.

```
export_fields_t qosMeasure_flow_fields[] = {
{IPFIX_ENO_FOKUS, IPFIX_FT_TASKID, 4},
{IPFIX_ENO_FOKUS, IPFIX_FT_FLOWID, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OID, 20},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_NSEQ, 4},
{IPFIX_ENO_FOKUS, IPFIX_FT_TSTAMP_SEC, 4},
```

```
{IPFIX_ENO_FOKUS, IPFIX_FT_TSTAMP_NSEC, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_PL_SIZE, 2},
};
```

The exporter is invoked at each exporting interval, which is also specified by [CMS](#) to the controller component. When the current measurements must be exported, the exporter component accesses the flow records and sends the information using the template presented above. To avoid conflicts between the exporter and the QoSMeasure function that inserts information into flow records (because flow records are kept in shared memory), a mutex was implemented. After the export, all records are cleared.

3.3.3 QoS Calculator

The QoS calculator was a function implemented to perform some computation of the QoS metrics presented in Section 2.3. It can also be considered as a function that runs inside the [NME](#), like the ones presented in Figure 3.10, but it is not mandatory. This function can run inside [CMS](#) in order to compute the desired metrics using the measurements collected and stored in the results database. This function works in conjunction with the ones described before, the first ones collect the measurement from specific observation points, either passively or actively, and the QoS calculator computes the required metrics and aggregates the measurements to be exported to the ANQoS component.

The task seems to be quite simple but it can be very complex because of the amount of measurements stored in the result database. To compute One-Way-Delay, packets from different observation points (or [NMEs](#)) must be matched and the time of arrival from these packets has to be subtracted. In order to simplify the matching function, the common flows between observation points are identified. For each flow common to more than one observation point, packets can be matched using the IPFIX field denominated as IPFIX_FT_NSEQ. When a packet is matched, the one-way-delay between the observation points can be calculated by subtracting both packets timestamps, and this metric is then stored in a array associated to the flow. The observation points pairs are also stored for further processing.

The delay variation is calculated like in Equation 2.2, and two consecutive delay values are at least necessary. Packet loss is calculated by identifying, for identical flows transversing more than one observation point, the difference between the packets sent and received for each class of service. The final required metrics are: owd_{min} , owd_{avg} , owd_{max} , $jitter_{min}$, $jitter_{avg}$, $jitter_{max}$, $packet_loss_{avg}$, $packet_loss_{max}$. To obtain these metrics, the *gsl* library is used to calculate for each metric the minimum, the average and the maximum values. Taking into consideration the error of some metrics, a percentile is used to eliminate the most diverse values.

Before, exporting results to the ANQoS component, measurement values must be aggregated by class of service, by multicast group and also by observation points. The

idea is to join all the different flows arrays and calculate the metrics considering the aggregation parameters mentioned before.

Following the aggregation, the metrics are exported using the template presented below:

```
export_fields_t mcast_fields[] = {
{IPFIX_ENO_DAIDALOS, IPFIX_FT_METERSTARTTIME, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_METERSTOPTIME, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OID1, 20},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OID2, 20},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_MULTICASTGROUP, 16},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_TRAFFICDIRECTION, 1},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_DSCP, 1},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OWDMINDELAY, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OWDAVGDELAY, 4},
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OWDMAXDELAY, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_MINJITTER, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_AVGJITTER, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_MAXJITTER, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_AVGPACKETLOSS, 2 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_MAXPACKETLOSS, 2 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_INTHROUGHPUT, 4 },
{IPFIX_ENO_DAIDALOS, IPFIX_FT_OUTTHROUGHPUT, 4 }
};
```

The QoS Calculator function supports a list of parameters, allowing more flexibility.

```
"options:\n"
"  -h                this help\n"
"  -v                verbose level\n"
"  --maxdelay <s>    to determine packet loss\n"
"  --stat <outfile> write statistic data to <outfile>\n"
"  --collector <host:port> collector to send results to\n"
"  --startts <val>   evaluation start time\n"
"  --stopts <val>    evalutatin stop time\n"
"  --tid-res <val>   task id\n"
"  --tid <val>,...<val> tasks id\n"
```

Besides the collector address (in this case is the [ANQoS](#)), the start and stop time of the evaluation can be specified. The other parameters "tid*" are mandatory because the taskId that has to be evaluated must be specified. The "tid-res" parameter can be used to identify the taskId of the calculation function so that CMS can keep track of the measurement results.

3.4 Integration

The interfaces were already defined so that integration would be an easy step to accomplish. The NME component works like a wrapper for the implemented measurement functions. As mentioned in Section 3.2.3, the controlling functions were already defined, the specific parameters from each function were the only thing left to test. The IPFIX export was also very peacefully integrated since the platform already provided an IPFIX library which only needed to be extended with some new fields. The OpenIMP platform provided a PHP interface through a web server. This GUI was extended in order to allow the testing of the new functions. This way, it was possible to configure graphically and to visualize the measurement results. Figure 3.11 depicts the GUI for configuring the function QoSMeasure, i.e. passive test with sampling methods. As it can be observed, the calculator is also selected in this interface, simplifying the configuration. In the Figure 3.11, the interface to configure the function ActiveMeasure is also shown.

new traffic sampling	
Measurement Name	mcast
Description	Multicast Passive Measurement
Start Time	2008 - 05 - 26 15 : 42
Duration	1 min
Filter	nofilter no filter
Interval (s)	300
Sampling Method	count-based
Sampling interval	1
Sampling Spacing	1
No. bytes to capture per stream	1
Hash Based Selection Ratio(%)	25
Servers	no process
Calculator	skyrack - Calculator@skyrack
Collector (hostname:port)	

Figure 3.11: Graphical user interface for configuring the function QoSMeasure.

The functions can also be configured using the command line interface provided with the OpenIMP platform. This type of configuration requires a connection with each of the NMEs that are needed to configure. On the other hand, the GUI allows a more simple and quick interface, for instance, it is possible for the QoSMeasure function to select several NMEs and also the QoS calculator. The NMEs appear in the GUI as drop down lists because they are initially added into the CMS. This one maintains the current status of all NMEs and their capabilities, i.e. the function they support.

new active one-way delay measurement	
Measurement Name	<input type="text"/>
Description	<input type="text"/>
Start Time	2008 - 04 - 26 13 : 53
Duration	1 min
create RRDB	<input type="radio"/> yes, <input checked="" type="radio"/> no
Interval (s)	300
Traffic Generator	no process
IP Version	<input type="radio"/> ipv4 <input type="radio"/> ipv6 <input checked="" type="radio"/> auto
Source Port	5000
Packet Rate (pkt/ps)	1
Payload Length (bytes)	100
Traffic Monitor	no process
Destination Port	5001
<input type="button" value="submit"/>	

Figure 3.12: Graphical user interface for configuring the function ActiveMeasure.

Chapter 4

Measurements results analysis

A test scenario, based on the DAIDALOS Testbed, is defined in this chapter. Using this scenario, several experiments were executed in order to evaluate the functions implemented.

4.1 Test Scenario

A small segment of the DAIDALOS testbed is depicted in Figure 4.1. The full scenario is presented in the Annex B. The test scenario is elaborated in order to evaluate the measured metrics before exporting them to the ANQoS.

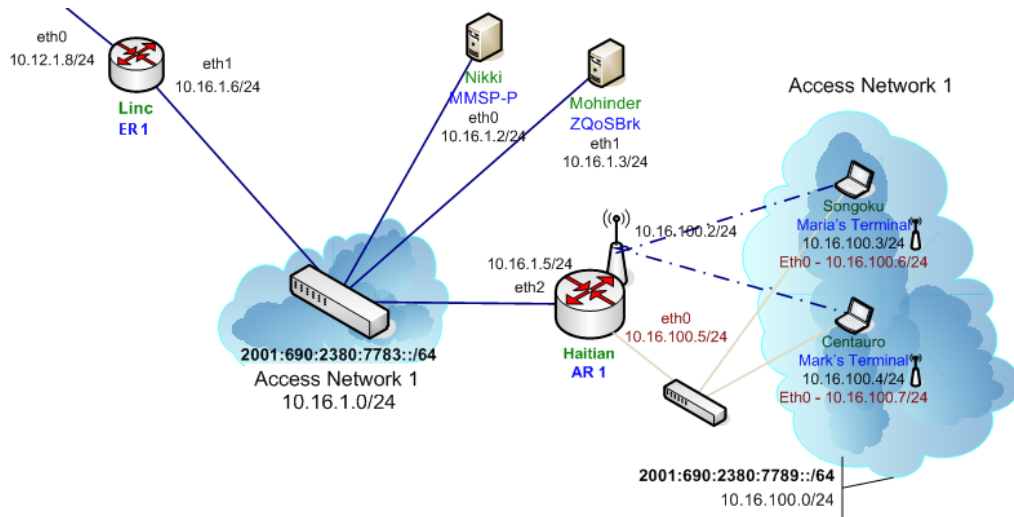


Figure 4.1: Test Scenario.

The ANQoS can configure the required measurements using the SOAP API. CMS configures the measurements on the NMEs, depending on the requested functionality. The NMEs perform the required measurements and export the results to CMS. Some of the measurements results, i.e. passive measurements, need to be post-processed by a second NME, using the QoS calculator function. The tests consisted in the evaluation of the

intrusiveness of the activeMeasure function and the accuracy of resulting QoS metrics. A second set of tests were executed to the qosMeasure function and the associated QoS calculator. Sampling methods were tested and the QoS calculator processing effort was estimated.

The NMEs were configured to run on the ER₁ and AR₁. To test the activeMeasure function, a measurement task was configured between both routers, and the results were exported to the CMS collector. The traffic was injected using a Poisson distribution. Four different flows were used to collect QoS performance metrics for each of the QoS classes previously defined.

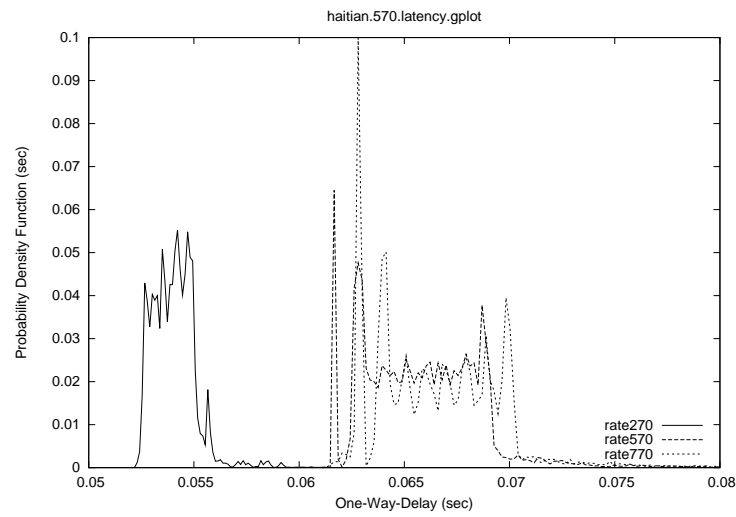
The function qosMeasure was tested with the help of mgen measurement tool. Test traffic was injected from the core network (which is located upstream ER₁), to the terminals Songoku and Centauro. The NMEs, on the routers ER1 and AR1, were configured to collect the multicast flow information, and export it to CMS. The QoS calculator then used this information to compute the required QoS metrics.

4.2 Results

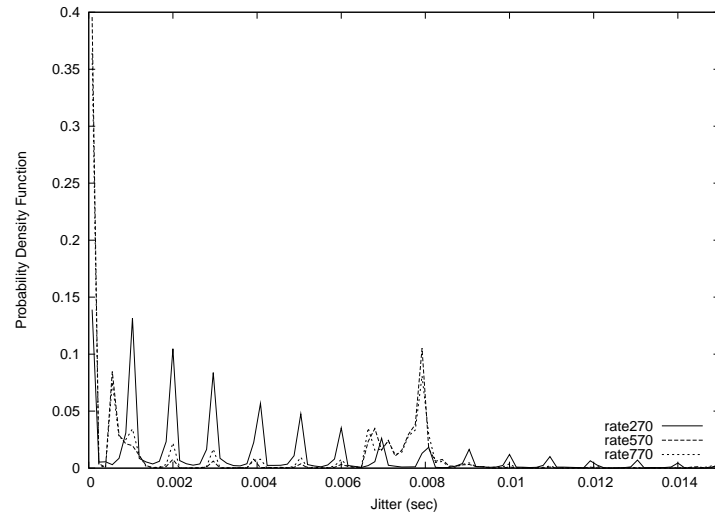
The scenario presented before was used to collect the network performance metrics, either by the executing active tests, with the activeMeasure function, or passively monitoring the network with the qosMeasure function.

The first scenario was the evaluation of the activeMeasure function and of the exported metrics, i.e. One-Way-Delay, jitter, and packet loss ratio. The collected results are depicted in Figure 4.2. This one presents the calculated metrics on the Maria's Terminal for a user session. The intrusiveness of the measurements are evaluated by injecting four multicast flows from the ER₁ to the AR₁ and monitoring how the user session is affected. The test active traffic is injected at different average bit rates, but with a constant payload size of 1370 bytes. The Figure shows that for the higher bit rates the One-Way-Delay and Jitter increased both their average values and maximum values. The average packet loss ratio for the different scenarios is considered 0. Moreover, the network performance was not affected, since the bottleneck of the scenario is located on the wireless technology, but active measurements are only executed till the AR

The passive measurements rely in the capture of network flows. If all the packets are captured, the QoS calculator has to process the information and to compute the required metrics. This can affect the responsiveness of this function, since for the same interval of time, more data must be selected from the database and processed. To overcome this problem, sampling methods were used. One of them, the random sampling method, was selected to calculate a hash, in order to synchronize the sampling processes. The other sampling methods (i.e. count-base and time-base) would required a sequence number so that the same information, collected in different observation points, would matched together. These methods can be used when RTP flows are captured because the header of

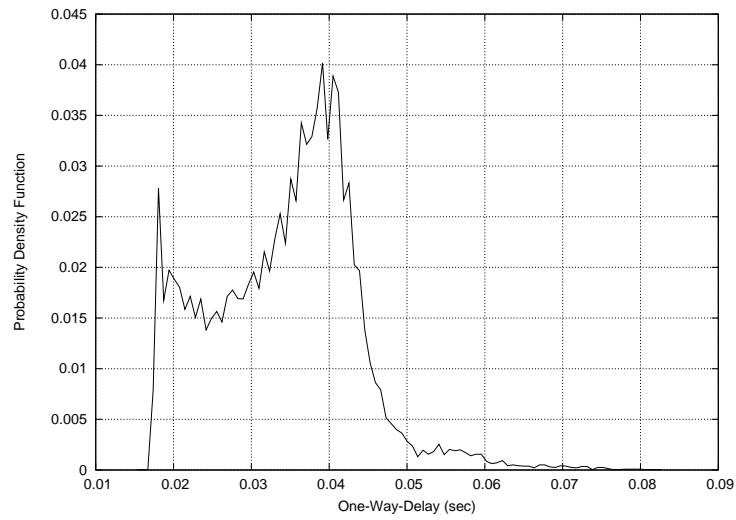


(a) One-Way-Delay Probability Density Function.

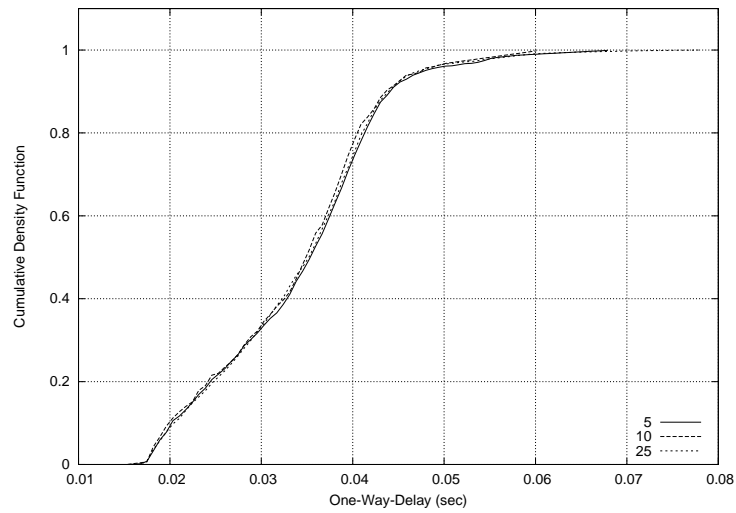


(b) Jitter Probability Density Function.

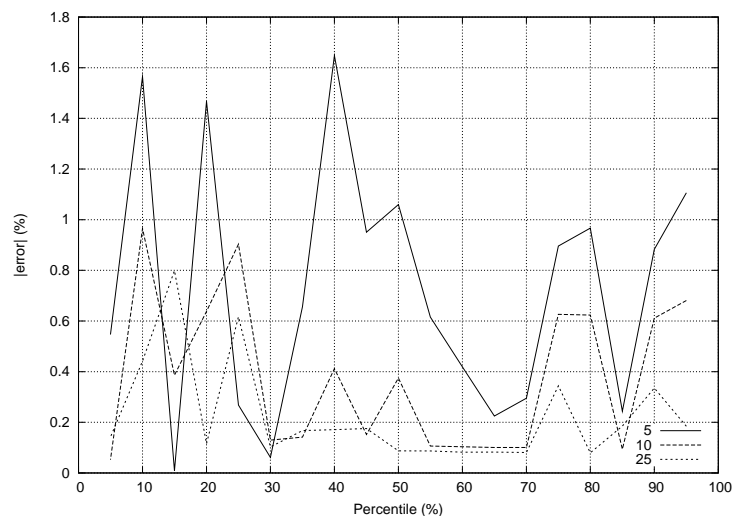
Figure 4.2: Active measurement results



(a) Probability Density Function.

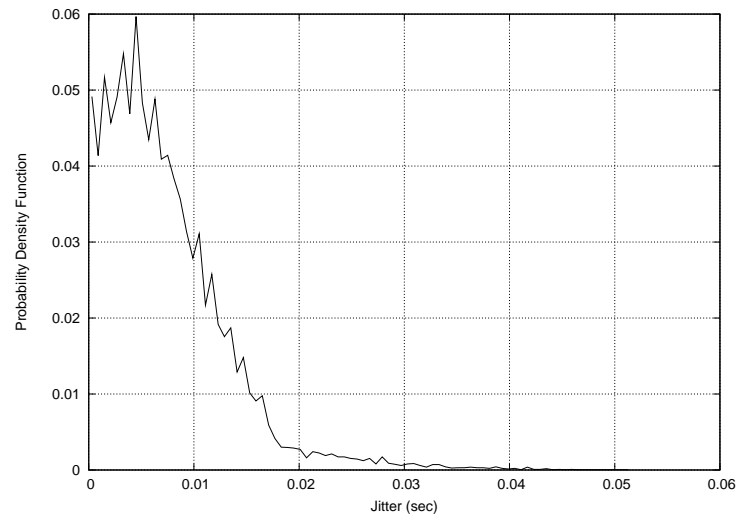


(b) Cumulative Density Function.

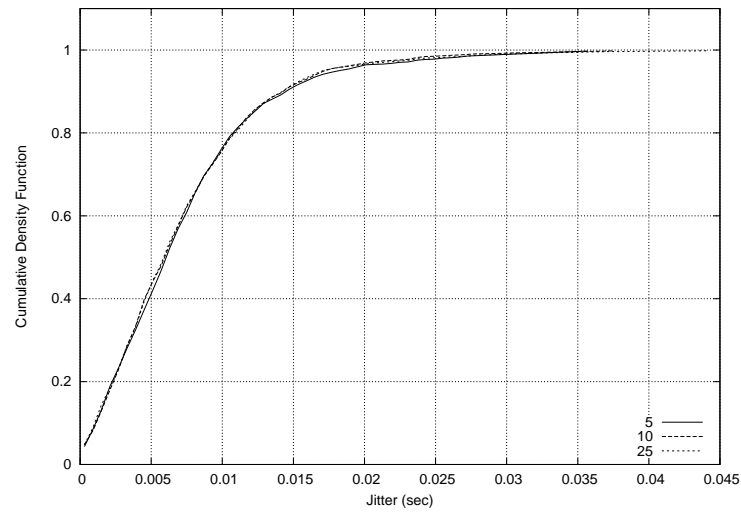


(c) Sampling error.

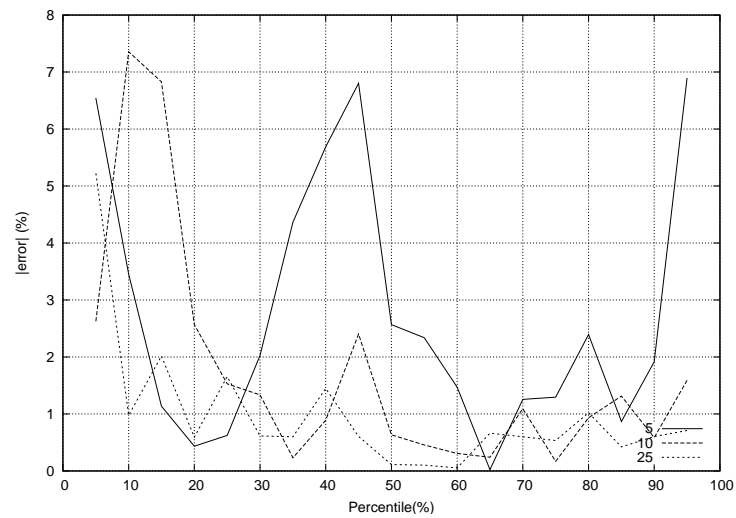
Figure 4.3: Passive measurement results - One-Way-Delay.



(a) Probability Density Function.



(b) Cumulative Density Function.



(c) Sampling error.

Figure 4.4: Passive measurement results - Jitter

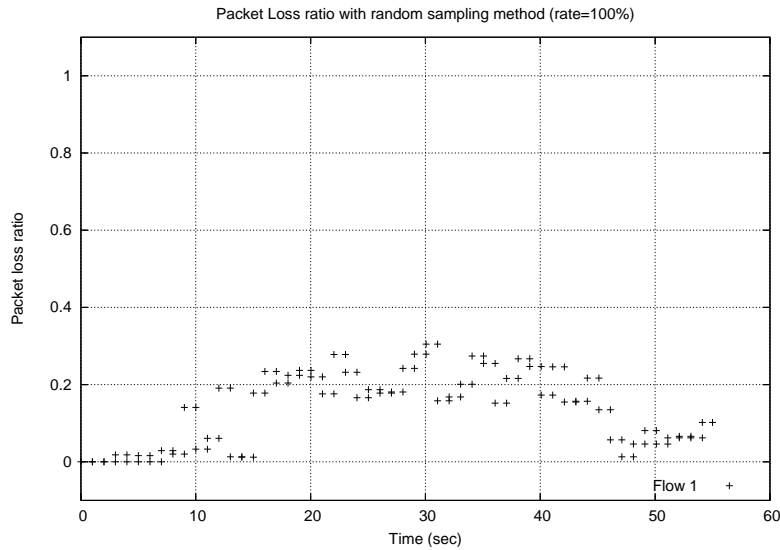


Figure 4.5: Passive measurement results - Packet Loss

this protocol carries a sequence number. In the scenario, only UDP flows were considered, so a hash had to be calculated per captured packet.

The Figure 4.3 shows the One-Way-Delay metric error for the different sampling rates and, for a sampling rate of 10%, the error is considered acceptable. The other metrics are also depicted in Figure 4.4 and Figure 4.5. The probability density function helps to identify how the values (One-Way-Delay and Jitter) are distributed. The cumulative density function depicts the differences between the selected sampling rates. The sampling error shows that for a sampling rate of 10% the required Percentils (5%, 50% and 95%) have a small error.

4.3 Evaluation

The enormous amount of measurement data that can be collected from the network is resource consuming; but sampling provides adequate techniques to overcome this problem. The computation of QoS metrics (e.g. delay, jitter, packet loss) can be one of the main problems, since it can affect the response time of the network monitoring system. We will present the results of sampling methods and their accuracy, taking into account the elapsed time of the QoS calculator function. The process of measuring data traffic metrics consists of the following steps:

- capturing packets from the wire;
- filtering packets;
- packet classification;
- processing the packet and its header fields to compute the metric of interest;

- exporting the results to a central collector entity for further processing;
- computation of metrics and aggregate measurement per class of service and observation points pairs;
- exporting the results to [ANQoSB](#).

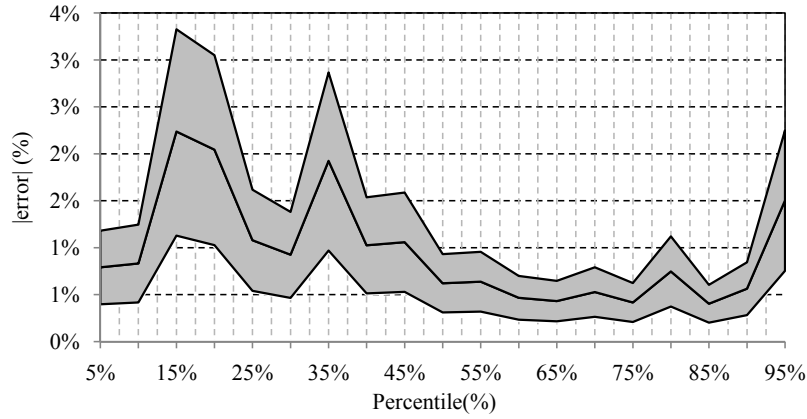


Figure 4.6: One-Way-Delay Error

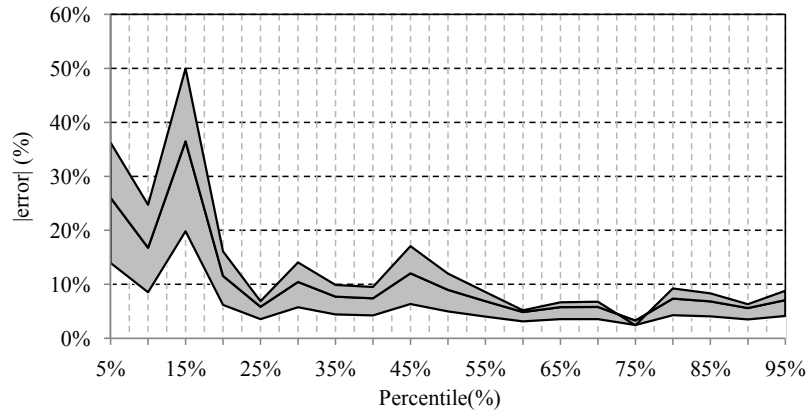


Figure 4.7: Jitter Error

Multipoint measurements were implemented. The sampling method needs to guarantee that the same packets were captured at all the involved measurement points so that metrics such as delay can be obtained. In order to perform this task, a hash-based method was selected to synchronize the sampling processes. This method is based on a deterministic function, based on the packet content; it is a special form of filtering that provides a pseudo random selection. Several tests were executed with different flow types to analyze the error related to the implementation of the hash-based sampling and the performance of the computation algorithm.

The tool mgen was used to generate different types of flows, following a Poisson traffic distribution. CMS was configured to monitor these flows using different sampling fractions

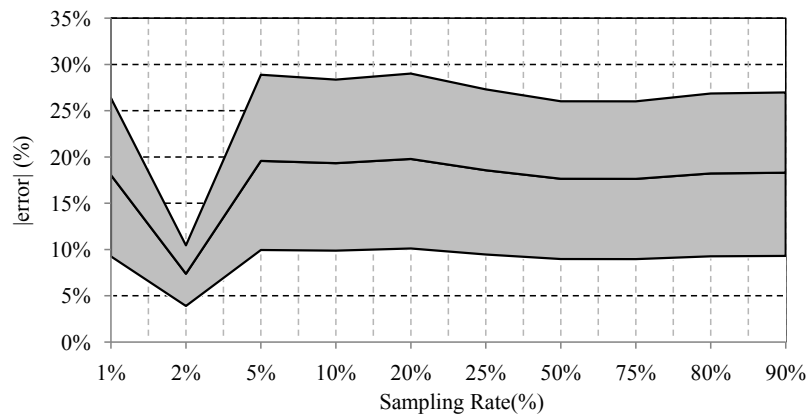


Figure 4.8: Packet loss sampling error

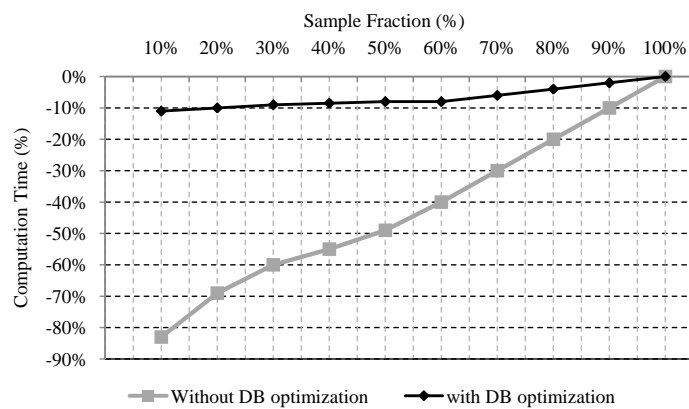


Figure 4.9: Metrics Computation Time.

and to compute the QoS metrics for each of them. The metrics selected were those used by the call admission control algorithm.

Figure 4.6 and 4.7 present the evaluation performed to the sampling method mentioned above. The Y axis represents the Percentile of the calculated metrics relative to the measured error caused by sampling. The tests were realized several times in order to calculate the mean sample values and the respective confidence interval for a sampling rate of 10%. This sampling rate achieved good results with a low measured error for the One-Way-Delay and Jitter calculation. The One-Way-Delay has a small error for all the Percentil's, but jitter has a high error for the lowest Percentils. Since the median and the 95% have errors below 10%, this can be considered a possible sampling rate to be adopted. The Figure 4.8 depicts the packet loss ratio error, the high error is justified by a small number of samples and a high packet loss ratio.

Besides the accuracy, the resource consumption of the implemented algorithm was also measured. The Figure 4.9 shows the percentual computation time relative to the sample fraction 100%. The measurements are stored in the results database and the structure of this database will be reflected on the performance of the metric computation algorithm. Several tests were made without any optimization of the database but the results were poor. This was caused by the lack of an index on the database. A database index is a data structure that improves the speed of operations in a table. They can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records. It was detected that some queries were performed using the flowId and taskId columns as conditions. After altering the tables and adding the flowId and taskId as index, we decreased the computation 4 times (below 1 second). As the sample fraction decreases, less time is required to compute the QoS metrics, because we are using fewer measurements to calculate the same metrics. Based on the evaluation performed we concluded that the metrics computation time can be decreased by implementing a hash-based sampling scheme, still obtaining metrics with a good accuracy. The computation time can be easily decreased if the database table index is modified. This can be seen as a requirement for the IPFIX protocol, where it is required to specify a field's template to be the index or primary key of the database table.

Chapter 5

Conclusions

5.1 Summary

This thesis presents a network monitoring solution for a multicast Call Admission Control (CAC) algorithm, developed for the DAIDALOS project. First, the adopted technologies, like IPv6, Multicast, and QoS, are briefly presented. Multicast monitoring tools are also studied but only a few of them fulfill the requirements. Multicast Quality Monitor is one of the possible tools that can be integrated into the monitoring system. Although it can manage measurement tasks and export measurements results to a central entity, the export is not compliant with the standards.

The admission of new multicast flows in the network requires information about the available resources. This can be obtained by a network monitoring system which coordinates the measurements tasks and collects the measurements results. The network monitoring system is compliant with the IPFIX standard and proves to be scalable and interoperable. System requirements were specified, based on the ANQoS call admission control algorithm. An architecture was designed in order to fulfill the requirements and to interface with the other DAIDALOS components, using the current standards, like IPFIX.

Multipoint measurements and sampling methods were implemented. The selected sampling methods ensure that the same packets are captured at different measurement observation points. The synchronization of all the sampling processes can be guaranteed by a hash based method. The packets are selected based on their content, providing a pseudo random selection mechanism.

Active measurements were implemented based on an open source tool and the performed evaluation proved that, for this application scenario, the measurements are not very intrusive. Nevertheless, a feedback mechanism, based on the collected passive measurements, can be implemented to avoid the disruption of users' sessions. CMS can configure a bandwidth threshold per interface on each NME that sends an IPFIX report when the bandwidth reaches the specified threshold. CMS can either remove the active measurement, or just change the average bit rate of each active test.

5.2 Achievements

The main achievements of this work are the solution developed for multicast network monitoring used for call admission control, and the performed evaluation of the sampling methods and active measurements. Three measurement functions were developed and integrated into the network monitoring system. One of them is used to passively measure the network QoS metrics. Sampling methods were implemented and integrated into this function. A hash-based method was used in order to synchronize the sampling processes and facilitate the QoS metrics calculation. A template for exporting the measurements, based on flows, was defined. The second function was implemented in order to calculate QoS metrics from the collected measurements of the previous function. Scalability and performance concerns were considered, not only related on the way data is fetched from the database, but also on the way packet matching process is performed. The calculated QoS metrics are then exported using a defined IPFIX template. The third function executes active tests on the network using a predefined multicast group. In order to test route asymmetry, unicast tests were performed on the reverse path. This function does not require any additional computation probe, because the metrics are calculated on the receiver side. Since the tests can be controlled, it is possible for a probe running on a router to perform the calculation of the QoS metric. These metrics are then exported to a central database and aggregated using a defined IPFIX template.

Sampling methods were evaluated in terms of accuracy for the selected QoS metrics. The main objective was to understand the real network gain in usage and processing effort for the QoS calculator by having sampling methods implemented. Sampling is very important because it can reduce the amount of measurements to be exported to a collector, and thus reduces the amount of network usage. Moreover, in order to calculate QoS metrics for multi-point measurements it requires to match the information from both points and calculate the QoS Metrics using all the collected information. This can be a burden for the QoS calculator, and it can require more time to calculate the QoS metrics. When a sampling method is used, the processing effort and the network traffic for exporting the measurements can be diminished. The problem is that sampling inserts an error in the QoS metrics estimation. So it requires a compromise between the sampling rate and the processing effort, so that the system would be able to respond in real-time.

5.3 Future Work

For future work, we can think about many different topics. One idea is to dynamically adjust passive and active measurements parameters. When the network load is high, measurements can lead to the disruption of user sessions or high packet drops. To prevent this scenario, the passive functions could monitor the routers queue, and signal the active measurement functions each time the router is dropping a lot of packets. In this way, the

active measurement functions can diminish the packet rate or even stop the tests in order to avoid user sessions' disruption. For passive measurement functions a similar mechanism can also be applied. When the network's usage is high, the passive measurement function could adjust the sampling rate in order to capture less packets. A smaller set can also give good results, but with less processing effort or even exported information traffic.

Another topic of study can be a decentralized approach for the deployment of network monitoring. For now, the major functionalities are provided by CMS, like measurement tasks management, NMEs management, and results databases. To start, the architecture can support more than one collector. Using this concept, the same collected measurements can be reused by applications to calculate different metrics, reducing the amount of monitored information. The second step should be decentralize the management of measurement tasks. This is accomplished by the adoption of novel signaling protocols like NSIS. This protocol has a specific application devoted to the metering configuration, M-NSLP. By using path-coupled or path-decoupled messages, it is possible to configure several probes.

Inter-domain measurements is also another topic that can be further developed. How can measurements be exchanged or performed between two different administrative domains, separated by several transit domains that can or cannot support measurements? There are several possibilities. One is assuming that domains can cooperate and, in this case, each domain executes measurements between its boundaries and exchanges measurements with the neighbor domains. In an other scenario, where domains do not cooperate and it is required to have end-to-end measurements, a negotiation process must exist. A solution can be the adoption of the signaling protocol defined before, M-NSLP. This protocol negotiates only with the domains capable of understanding it and which can support the desired measurement functions. There is a third alternative, where a trusted third-party domain is responsible for performing and collecting the measurements on the other domains, by using a well defined equipment.

Appendix A

Monitoring platforms evaluation

This section presents a comparison of the main characteristics and features of the available monitoring and measurement platforms. The evaluated platforms were:

- Open Internet Measurement Project (OpenIMP);
- IP Probes;
- Monitoring Platform for Mobile Flows (MPMF);
- NeTraMet.

A.1 Platform Description

A.1.1 Open Internet Measurement Project (OpenIMP)

The Open Internet Measurement Platform (OpenIMP) has been designed to distribute IP traffic and quality of service measurements. The range of measurements supported includes volume, one-way-delay, round-trip-delay, jitter, and packet loss. IMP includes passive and active measurement components, and it has a modular design that can be adapted to several different kinds of measurements. The results can be used to feed other platforms, for example, for usage-based accounting, SLA validation, intrusion detection and traffic profiling. The time synchronisation is achieved either via a GPS signal or via NTP. The OpenIMP main components are:

- Active meters;
- Passive meters;
- Data collector;
- Evaluation server - Results post-processing and visualisation;
- Measurement control unit with Web interface.

For the data export from the probes, the IPFIX protocol is currently used.

A.1.2 IP Probes

The IP Probes were designed for distributed IP traffic and quality of service measurements. The current version of the IP Probes platform performs only active measurements. The range of measurements supported are one and two way delays (end-to-end), one and two way delays with route specification, jitter and packet loss. At a higher layer, the platform is integrated with another one, able to verify SLA, generate alarms, etc. The synchronisation is achieved either via embedded GPS capabilities in the distributed active probes or via NTP (less accuracy). The IP Probes main components are:

- Active meters;
- Element of operation, results visualisation and interface with external entities, accessible via a Web interface;
- Data Base for pre and pos processed results.

A.1.3 Monitoring Platform for Mobile Flows (MPMF)

The platform has been designed to evaluate IP traffic quality of service measurements, with a special focus on IPv6 mobility scenarios. The MPMF is an operational distributed passive monitoring system for a MIPv6 based access network. Its features are:

- Unequivocal identification of mobile IPv6 flows by the samplers;
- Graphical user interface (through which multiple performance information is made available to the user);
- Assessment of individual flow identification parameters;
- Packet and time statistics, resourceful flow plot implementation including a high precision zoom;
- Real-time network topology view and flow path display;
- Four network traffic sampling techniques for UDP and TCP protocols;
- The host, network, port and transport protocol oriented filtering of captured/sampled traffic;
- Implementation of logging for saving and posterior loading of monitored network traffic.

The MPMF main components are the sampler and collector.

Sampler:

- Hosted in the network elements;

- Captures the information in the network interfaces;
- Samples and filters the captured packets;
- Process IPv6 packets;
- Regularly sends the information to the collector.

Collector:

- Hosted in a server (dedicated machine);
- Receives all the information from the samplers;
- Process all the information received;
- Identifies and classifies flows;
- Controls the networks samplers (sampling and filtering).

The NetFlow9 protocol, defined by Cisco Systems, is used for the communication between the sampler and the collector.

A.1.4 NeTraMet

NeTraMet is a meter for network traffic flows. It performs passive network monitoring. NeTraMet measures one-way delays, packet losses and throughput. Synchronization can be achieved either by a GPS signal (not embedded) or NTP. It has three components:

- Meters;
- Meter readers;
- Managers.

Although several modifications to this platform may be available, the default version will be the one here analysed.

A.2 Evaluation

The table presented depicts a comparison between the monitoring platforms described.

Table A.1: Comparison between the monitoring platforms

General Features	OpenIMP	IP Probes	MPMF	NeTraMet
IPv4	yes	yes	no	yes
IPv6	yes	yes	yes	no
Embedded GPS synchronisation	yes	yes	no	no
Accuracy of the packet marking (using GPS)	$\approx 100 \mu\text{sec}$	$\leq 100 \mu\text{sec}$	no	no
NTP synchronisation	yes	yes	yes	yes
OS	BSD/Linux	BSD/Linux	Linux	BSD/Linux
Interface to external elements (specify)	Web interface for control + results GUI	Web services	no	no
Ethernet/Fast Ethernet interface	yes	yes	yes	yes
Wireless (802.11b) interface	yes	yes	yes	yes
Wireless (802.11g) interface	yes	yes	yes	yes
Bluetooth interface	yes	yes	no	yes
Self hardware platform	yes	yes	no	no
Dedicated Hardware Network Processor	yes	yes	-	-
Dedicated hardware platform able to integrate GPRS/UMTS and GSM measurements	no	yes	-	-
Integrated 1xRTT CDMA 2000 interface	no	yes	-	-
Active Measurements	yes	yes	no	no
Packet Loss	yes	yes	-	-
One way Delay	yes	yes	-	-
Round Trip delay	yes	yes	-	-
One way jitter	yes	yes	-	-

Continues on Next Page...

General Features	OpenIMP	IP Probes	MPMF	NeTraMet
Round Trip Jitter	no	yes	-	-
Throughput	yes	yes	very intrusive	
Route Specification for each measure	no	yes		
Configurable packet distributions	no	yes		
Configurable interpacket time distribution for packet sending	yes	yes		
Configurable TCP/IP layers	5	4	-	-
DNS, HTTP performance analysis	yes	no	-	-
Total packets	yes	yes	-	-
Total bytes	yes	yes	-	-
Passive Measures	yes	no	yes	yes
Packet Loss	yes	-	yes	yes
Delay	yes	-	yes	yes
Jitter	yes	-	yes	yes
Instantaneous flow Bit rate (for real time visualisation)	no	-	yes	yes
Flow bit rate (average)	yes	-	yes	no
Aggregate bit rate (aggregate and view flows with similar characteristics)	yes	-	yes	no
Path Discovery	no	-	yes	no
IPv6 Mobility aware	no	-	yes	no
Filters configurable until the TCP/IP layer n.	5	-	4	no
Retransmitted packets	yes	-	yes	no
RTP loss	yes	-	no	no

Continues on Next Page...

General Features	OpenIMP	IP Probes	MPMF	NeTraMet
Tariff Formula language to express charging schemes	yes	-	no	no
Display of charges per flow	yes	-	no	no
Centralised Management	yes	yes	yes	yes
Web Interface	yes	yes	no	no
Control interface with specific software	yes	no	yes	no
Command line control interface	yes	no	no	yes
Data Collector	yes	yes	yes	yes
Operation and Control Element	yes	yes	yes	yes
Interface towards a management platform	no	yes	no	no
Integrated Result Visualization	yes	yes	yes	no
Web based	yes	yes	no	-
Own visualization system	yes	no	yes	-
Zoom tool (zoom in and out on the graphics with the results)	yes	no	yes	-
Integrated network topology visualisation	no	no	yes	-
Test Execution and Reporting				
Possibility to view the test results in real time	no	yes	yes	yes
Test Scheduler	yes	yes	no	no
Test Results Storage/Centralized	yes/yes	yes/yes	yes/yes	yes/yes
Interactive test mode (real time test set-up)	yes	yes	no	no

Continues on Next Page...

General Features	OpenIMP	IP Probes	MPMF	NeTraMet
Macros (templates) for test specific ations	yes	yes	no	no
SLA conformance verification	yes	yes	no	no
Report generation to external systems	no	no	no	no
Storage of complete test results	yes	yes	yes	yes
Storage of filtered test results	yes	yes	no	no
Other Features				
Standards compliance IPPM, IPFIX, IPSAMP	IPFIX export im- plementation in progress	no	yes	no
Transactions capture	no	no	no	no

A.3 Licensing and availability

A.3.1 OpenIMP

The IMP platform is a free software; it can be redistributed and/or modified under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (optionally) any later version. Some metric computation components can be restricted in their redistribution due to a separate copyright.

A.3.2 IP Probes

IP Probes can be provided free of charge for the DAIDALOS project.

A.3.3 MPMF

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (optionally) any later version.

A.3.4 NeTraMet

NeTraMet is free software, distributed under the terms of the GNU General Public License. A copy of this is provided with the NeTraMet software distribution files.

A.4 Conclusions

The main conclusion of this evaluation is that none of the available platforms fully comply with all the necessary requisites. With the exception of NeTraMet, all the others present a user friendly environment for test configuration and reporting. The OS in which these platforms operate are indicated to the DAIDALOS environment.

OpenIMP - it is the broadest platform in the sense that it supports both IPv4 and IPv6, both active and passive measurements and it goes in the analysis up to the application layer. However, there are some features that make it incomplete and some further evaluation must be done.

IP Probes - this platform lacks the passive monitoring features. In terms of active monitoring, the measurement of application layer characteristics is still missing. The existent interfaces with a network management system and with a QoS Broker are good features. The possibility to have dedicated hardware to perform certain functions may be a value added feature. Licensing and availability must be further evaluated.

MPMF - this tool is limited to IPv6 passive monitoring. However, in this field, it implements a set of very useful features, especially those related to IPv6 mobility and route path discovery. Licensing and availability must be further evaluated.

NeTraMet - the default version of this tool operates in IPv4 only. Additionally, it supports passive monitoring only. When compared to the other available platforms in this segment (passive monitoring), the characteristics of the others are more suitable for the DAIDALOS environment. However, the licensing and availability within the consortium may be a determinant factor, due to the "free" nature of this platform.

Appendix B

DAIDALOS testbed

B.1 Accident and University scenario

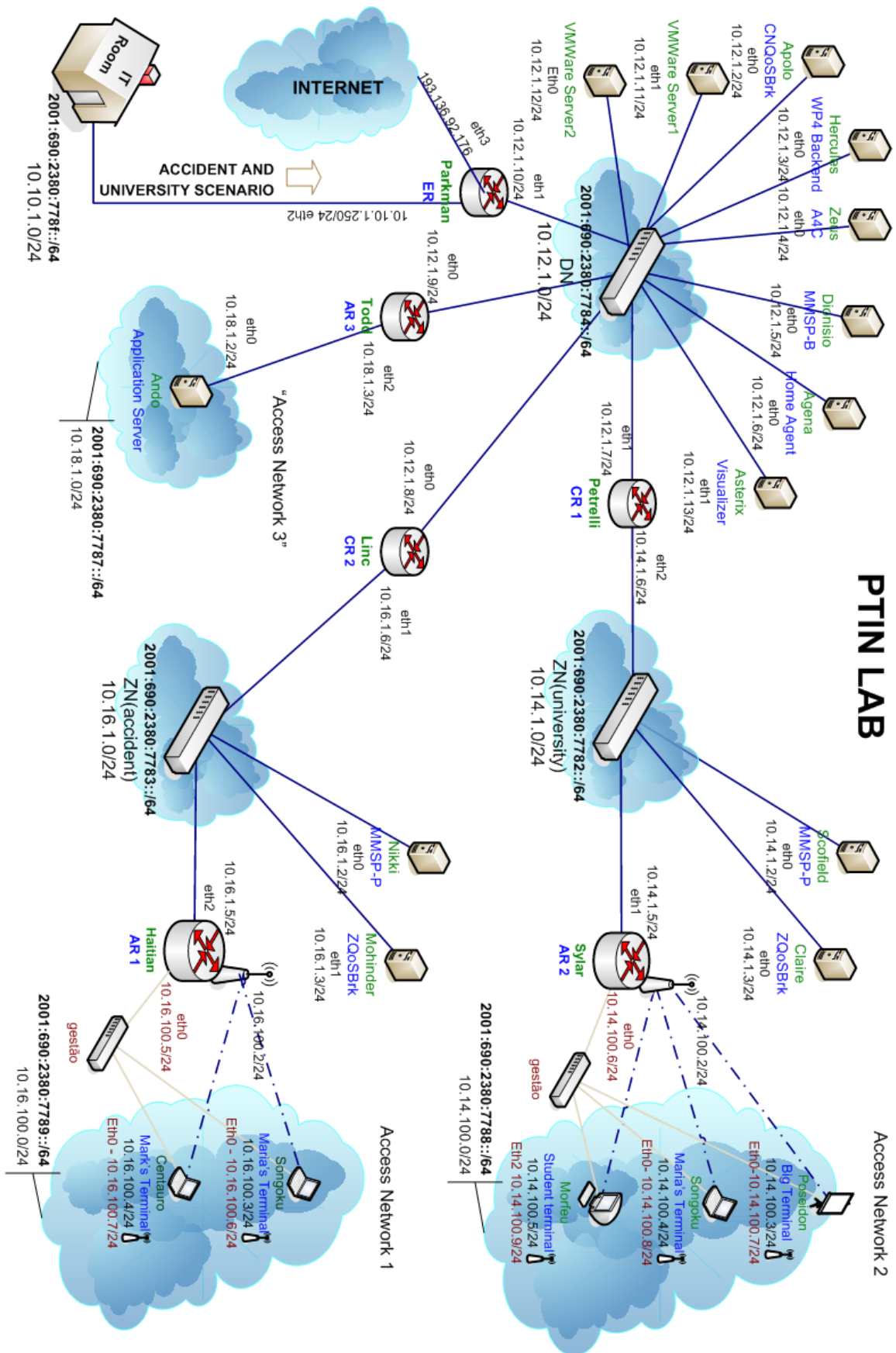


Figure B.1: DAIDALOS testbed

Bibliography

- [1] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.
- [2] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFC 5095.
- [3] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. RFC 3095 (Proposed Standard), July 2001. Updated by RFCs 3759, 4815.
- [4] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 2373 (Proposed Standard), July 1998. Obsoleted by RFC 3513.
- [5] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), February 2006.
- [6] R. Vida and L. Costa. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810 (Proposed Standard), June 2004. Updated by RFC 4604.
- [7] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601 (Proposed Standard), August 2006. Updated by RFC 5059.
- [8] S. Deering, W. Fenner, and B. Haberman. Multicast Listener Discovery (MLD) for IPv6. RFC 2710 (Proposed Standard), October 1999. Updated by RFCs 3590, 3810.
- [9] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard), October 2002. Updated by RFC 4604.
- [10] A. Adams, J. Nicholas, and W. Siadak. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised). RFC 3973 (Experimental), January 2005.
- [11] M. Handley, I. Kouvelas, T. Speakman, and L. Vicisano. Bidirectional Protocol Independent Multicast (BIDIR-PIM). RFC 5015 (Proposed Standard), October 2007.
- [12] S. Bhattacharyya. An Overview of Source-Specific Multicast (SSM). RFC 3569 (Informational), July 2003.

- [13] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard), December 1998. Updated by RFCs 3168, 3260.
- [14] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633 (Informational), June 1994.
- [15] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246 (Proposed Standard), March 2002.
- [16] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC 2597 (Proposed Standard), June 1999. Updated by RFC 3260.
- [17] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205 (Proposed Standard), September 1997. Updated by RFCs 2750, 3936, 4495.
- [18] B. Claise. Packet Delay Variation Applicability Statement. Internet-Draft draft-ietf-ippm-delay-var-as-00, Internet Engineering Task Force, February 2008. Work in progress.
- [19] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Proposed Standard), January 1996. Obsoleted by RFC 3550.
- [20] G. Sadasivan. Architecture for IP Flow Information Export. Internet-Draft draft-ietf-ipfix-architecture-12, Internet Engineering Task Force, September 2006. Work in progress.
- [21] P.D. Amer and L.N. Cassel. Management of sampled real-time network measurements. In *Local Computer Networks, 1989., Proceedings 14th Conference on*, pages 62–68, 1989.
- [22] N. Duffield. A Framework for Packet Selection and Reporting. Internet-Draft draft-ietf-psamp-framework-12, Internet Engineering Task Force, June 2007. Work in progress.
- [23] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer. Information Model for IP Flow Information Export. RFC 5102 (Proposed Standard), January 2008.
- [24] K. Almeroth, K. Sarac, and L. Wei. Supporting multicast management using the multicast reachability monitor. 2000.
- [25] Falko Dressler. *Scalable QoS Measurements in Multicast Environments*. May 2003. Published: Poster.
- [26] S. Sargento, V. Jesus, F. Sousa, F. Mitrano, T. Strauf, C. Schmoll, J. Gozdecki, G. Lemos, M. Almeida, and D. A10 Corujo Corujo. Context-aware end-to-end qos architecture in multi-technology, multi-interface environments. In *Mobile and Wireless Communications Summit, 2007. 16th IST*, pages 1–6, 2007.

- [27] B. Claise. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Proposed Standard), January 2008.
- [28] E. Stephan. IP Performance Metrics (IPPM) Metrics Registry. RFC 4148 (Best Current Practice), August 2005.
- [29] B. Claise. Packet Sampling (PSAMP) Protocol Specifications. Internet-Draft draft-ietf-psamp-protocol-09, Internet Engineering Task Force, December 2007. Work in progress.
- [30] A. Bak, A. Beben, W. Burakowski, M. Dabrowski, M. Fudala, Z. Kopertowski, and H. Tarasiuk. On handling streaming and elastic traffic in ip based aquila network: measurement results.
- [31] T. Zseby. Deployment of sampling methods for sla validation with non-intrusive measurements. *Proceedings of Passive and Active Measurement Workshop*, 2001.